ComS 252 Homework 9: Web server

Group assignment (check syllabus for group penalty details)

Due November 14, 2023

1 Objectives

In this assignment, you will configure network settings and build a web server, in Linux.

2 Build a virtual machine

1. Download the ISO file to initialize the virtual machine for homework.
2. In VirtualBox, create a new virtual machine for this assignment. The default disk size (a few GB) should be sufficient.
3. Set the ISO file as the optical disk, and boot up the VM.
4. Select "Build Hw09 virtual machine" from the boot menu.
5. After installation completes, remove the homework initialization ISO from the Optical Drive. You are encouraged to take a snapshot of the VM at this point, in case you need to roll back to a fresh install.
6. At first boot, the VM initializes itself by fetching and running a script from the server. This requires Internet access, and VPN access if you are off campus. The script will, among other things, create a user account with your ISU username. All user accounts will initially have passwords that are the account name, followed by "`pw`".
7. When the VM shuts down after initialization, you are again encouraged to take a snapshot of the VM. That way, if you make a mistake and accidentally trash the user files, you can easily roll back to a freshly initialized VM.

3 Set up host to guest networking

3.1 If you are using VirtualBox

1. Build a host–only network, to allow communication between your host machine and virtual machines. 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. Make sure "DHCP server" is *enabled*, and to make your life easier, you can restrict the range of addresses to a single address. For example, Figure 1 shows a host–only network named "vboxnet0", that will give out IP addresses in the range `192.168.56.56` through `192.168.56.56`.
2. Under the settings for your VM, under "Network", you should already have Adapter 1 enabled and attached to a "NAT" network. Now, enable Adapter 2, attach it to a "Host–only Adapter", and select your host–only network name.
3. Boot up the VM, login, and run `nmcli` to make sure both network devices are connected and have obtained IP addresses (under `inet4`). Make a note of the second adapter IP address. This will be referred to as the *guest address*. Make sure this address falls within the range of your host–only DHCP server settings.
4. On your *host* machine, open a terminal window, and try to `ping` the guest address. If packets can get through, then you are ready to configure the web server.

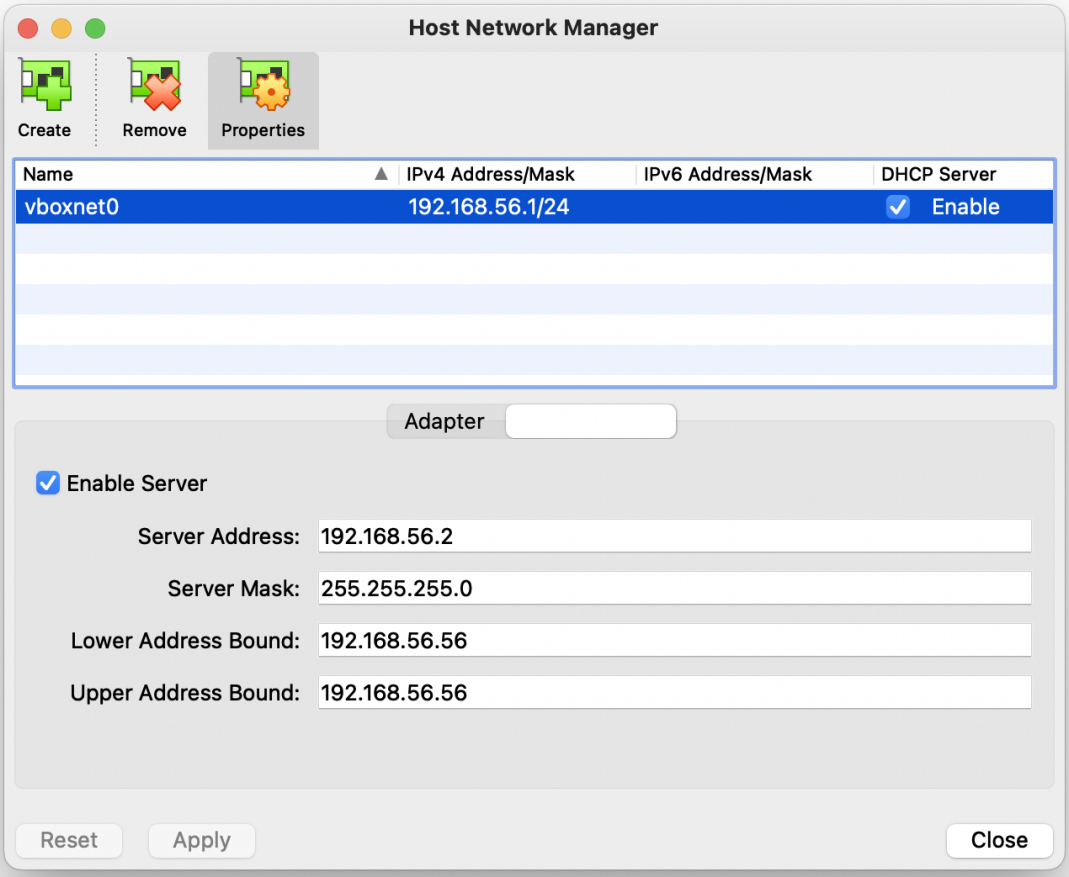

Figure 1: Host–only network settings

3.2 If you are using UTM

1. You should be able to complete this assignment with a single network adapter; make sure the "Network Mode" is set to "Shared Network".

2. Boot up your VM, login, and obtain the IP address using `nmcli`. It should be under "inet4" and will be something like `192.168.64.3`. This will be referred to as the *guest address*.

3. On your *host* machine, open a terminal window, and try to `ping` the guest address. If packets can get through, then you are ready to configure the web server.

4 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!)

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://`*guest-address*, 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.

7. Reboot the VM to make sure your service and firewall settings remain in effect.

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.

5 Static content

The Apache configuration page will indicate the "DocumentRoot", where static content should be placed. Usually this is directory `/var/www/html`. Create an example HTML file here, with name `index.html`. This should be visible at URL `http://`*guest-address*. Make sure the HTML file contains the **ISU netid** of the primary user account on the VM, somewhere in the file. 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>
<p>
Website by asminer.
</p>
</body> </html>
```

Your HTML content can be very simple, but it must contain at least one header tag (`<h1>`, `<h2>`, etc.) and one paragraph tag.

6 Dynamic content

Web pages can be dynamically created by the server. One way to do this is to write a bash[1] 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.

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

- `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!"
   ```

   and an HTML version of the script:

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

   Save the file as `hello.cgi` and turn on execute permission. You should then be able to view it at URL `http://`*guest-address*`/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. The time should have format `HH:MM:SS`. 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 and should retain the column format as output by `ps`. Check the script in a browser.

4. Create a CGI script named `env.cgi` that displays a list of all environment variables (using `env`). This script should produce plain text, and should retain the format as output by `env`.

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 a text file to save the value of the counter; this should be placed in the same directory as `counter.cgi` and have name `.counter.txt`. You will need to do the following for `.counter.txt`:

   - Set the file owner/group/permissions appropriately, so that the CGI script is able to modify the file. For full credit, the file should *not* be world-writable.
   - Tell SELinux that the file is allowed to be updated, by running

     ```
     chcon -t httpd_sys_rw_content_t .counter.txt
     ```

     within the directory containing the text file.

   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.

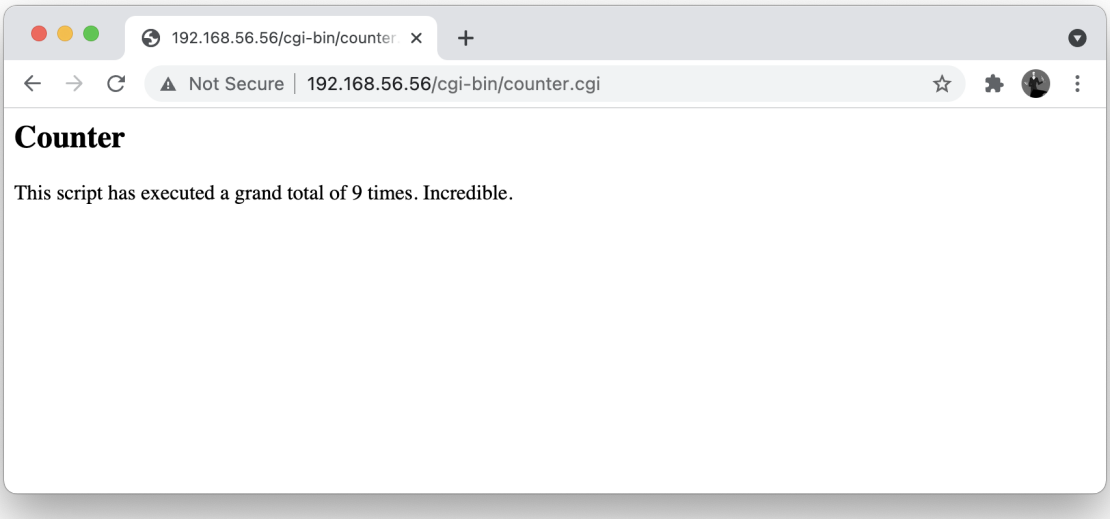

Figure 2: Example of `counter.cgi` in a browser

7 Submitting your work

From your user account, run "`sudo Turnin`" to submit your work. Again, this requires Internet access (and VPN access, from off campus), as this will collect and upload your work to the homework server.

Feedback on your submission is collected in a text file, that you can view later using "`cat submit.log`" or "`less submit.log`".

To shutdown the VM cleanly, run "`poweroff`".