Final-exam Solution

1 OpenVPN (20%) ★★★

Follow this tutorial and you can setup an OpenVPN server.

2 DHCP (20%) ★

Since the choices of packages is not restricted, there are many solutions. Here are some of the tutorials:

1. CentOS
2. pfSense

3 Plan H (20%, 2 subtasks) ★★

1. (10%) Just follow what we do in our dns lab. Don't forget to modify which dns server your OS uses.
2. (10%) DNS masterslave tutorial

4 Network Debugging (20%) ★★★☆

1. he can SSH to CSIE workstations from his machine without remembering IP addresses (and resolving domain names should be done in an instant without lagging),
2. for new TCP connections to his machine, only those to port 22 will be accepted,
3. the changes you made should be persistent across reboot.

5 Kawaii Cipher カワイイサイファ (20%) ★☆

1. The cipher uses current time as random seed, which is often used. However, the random function is predictable. So, all we have to do is brute force the random seed.
2. Sample solution can be found here.
3. **Always use cryptographically secure RNG.**

6 Custom Arch Repository (20%) ★★★

We use aurutils and apache to help us finish this task.

1. Create two Arch Linux instance in VirtualBox, and add a new NAT network, and connect these two machine via it.
2. Under `repo`

(a) Run `echo repo > /etc/host`

(b) Install `aurutils-git` from AUR, and install `apache` from official repository.

(c) Run `sudo systemctl start httpd`

(d) Add the following setting to the bottom of `/etc/pacman.conf`

```
[options]
CacheDir = /var/cache/pacman/pkg
CacheDir = /srv/http/nasa-final
CleanMethod = KeepCurrent

[nasa-final]
SigLevel = Optional TrustAll
Server = file:///srv/http/nasa-final
```

(e) Run `aur sync 2048-rs kakoune-git nnn`

3. Under `client`

(a) Run `echo client > /etc/host`

(b) Add the following setting to the bottom of `/etc/pacman.conf`

```
[nasa-final]
SigLevel = Optional TrustAll
Server = http://<ip address of repo>/nasa-final/
```

(c) Run `sudo pacman -Syu 2048-rs kakoune-git nnn`

7 Simple NFS (20%, 2 subtasks) ★☆

1. (8%) Fulfill the following requirements.

```
$ pvcreate /dev/sd{b,c}
$ vgcreate storage /dev/sd{b,c}
$ lvcreate -n script -L 50Gib storage
$ lvcreate -n backup -l 100%FREE storage
$ mkfs.ext4 /dev/storage/script
$ mkfs.ext4 /dev/storage/backup
$ mkdir -p /share/script /share/backup
$ mount /dev/storage/script /share/script
$ mount /dev/storage/backup /share/backup
```

2. (12%) On the first VM (the same as in Part 1), create an NFS server and allow only the second VM's IP to access both `/share/script` and `/share/backup` with read and write permissions.

VM1:
First, installed required packages.

```
$ yum install -y nfs-utils rpcbind
```

Edit /etc/exports, add the following lines.

```
"/share/script" <SECOND VM IP>(rw,no_root_squash)
"/share/backup" <SECOND VM IP>(rw,no_root_squash)
```

Add firewall rules.

```
$ firewall-cmd --permanent --add-service=rpc-bind
$ firewall-cmd --permanent --add-service=mountd
$ firewall-cmd --permanent --add-service=nfs
$ firewall-cmd --reload
```

Start rpcbind and nfs daemons.

```
$ systemctl start rpcbind
$ systemctl start nfs
```

VM2:
First, installed required packages.

```
$ yum install -y nfs-utils
```

Then mount them on VM2.

```
$ mkdir /script /backup
$ mount -t nfs <FIRST VM IP>:/share/script /script
$ mount -t nfs <FIRST VM IP>:/share/backup /backup
```

8 Codera's Editor (20%, 3 subtasks) ★★★★

Please find sample answer here.

9 GRRR... RUB (20%) ★★

Briefly, there are two issues in this VM: The GRUB configuration file and the initramfs.
For the initramfs, you have to download the Arch image, and boot the VM with it. What you should do is to `chroot` into the VM's (disk) root, and then fix the initramfs like below:

```
$ mount /dev/sda1 /mnt
$ arch-chroot /mnt
# Check /usr/lib/modules to find the right version of kernel to specify
$ mkinitcpio -g /boot/initramfs-linux.img -k 4.17.2-1-ARCH
```

After this, you can directly change the image path in `/boot/grub/grub.cfg`, or you can use `grub-mkconfig` to generate new configuration file, and that's it! Easy, huh?

10 Web Server (20%, 3 subtasks) ★★☆

there are many ways to congifure the network

```
1. choose internal interface and set ip manually
RP: Bridge (or NAT + Host-Only) + internal /  WS: internal
2. choose NAT network interface
RP: NAT network + Host-Only / WS: NAT network
3. choose Bridge interface and add some rule so your host cannot ping WS
RP: Bridge / WS: Bridge
```

some preparation

```
nmtui  ## activate enp0s3 interface
dhclient en0s3  ##  another way
yum -y install httpd
systemctl start httpd
systemctl enable httpd ## automatically start on boot
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
```

now copy RP as WS (initialize Mac address)

1. (10%) Run a simple website on `WS`.

   ```
   ## you can simply add VirtualHost in /etc/httpd/conf/httpd.conf
   ## remember to reload
   ( On RP )
   ProxyPass /web-app1/ http://[ip of WS]/
   ProxyPassReverse /web-app1/ http://[ip of WS]/
   ( On WS )
   $ cp web1.html /var/www/html/index.html
   DocumentRoot /var/www/html
   ```

2. (5%) Multiple websites & hosts

   ```
   ( /etc/hosts )
   [ip of RP] web-app1 web-app2
   ( On RP )
   ServerName web-app1
   ProxyPreserveHost on
   ProxyPass  / http://[ip of WS]/
   ProxyPassReverse / http://[ip of WS]/
   ( On WS )
   $ cp web1.html /var/www/html/web-app1/index.html
   ServerName web-app1
   DocumentRoot /var/www/html/web-app1
   ```

3. (5%) HTTP

   ```
   ( On RP )
   ## check conf.d/ssl.conf,  if you can't find it, it means there is
   ## no mod_ssl, install it ( it is why you might see something
   ## like <ifModule mod_ssl.c> )
   $ cp ca.crt /etc/pki/tls/certs/localhost.crt
   ```

```
$ cp ca.key /etc/pki/tls/private/localhost.key
ServerName web1
redirect / https://web1
<VirtualHost *:443>
        ServerName web1
        DocumentRoot /var/www/html/
        SSLEngine on
        ## can be omitted if you use the path in conf.d/ssl.conf
        SSLCertificateFile /etc/pki/tls/certs/localhost.crt
        SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
</VirtualHost>
$ vim /var/www/html/index.html ## or cp a html file
$ firewall-cmd --add-service=https --permanent
$ firewall-cmd --reload
```