Securing Web Applications Lab

35 Points

Due Date: Start of Lesson 14

1 Overview

Previously we explored interactions between network configurations, installation of domain services like DNS, along with the vulnerabilities each faces. Now it is time to investigate how to secure applications and services that would be deployed on the network. The main challenge with these applications is that they allow access to view and manipulate content on a server by remote users. A system administrator needs to not only protect the integrity and data of hosted applications from malicious users, but also protect the host system itself from exploitation.

In this lab we will focus on securing common applications such as web and database services.

2 Lab Setup

For this lab you will need two Ubuntu VMs, each configured with an IPaddress and able to ping each other. PAY CLOSE ATTENTION TO THE CONFIGURATION! The Lab "victim" image is server image with multiple web front-ends and a MySql backend. It has been specially crafted to be exploitable. This lab documentation will refer to it as the 10.172.XXX.10 machine or VM2 Server/Victim. Use the "Observer" image for the client, referred to here as 10.172.XXX.11 and VM1 Client/Observer.

As always, if you make a snapshot of your VM at the beginning of the lab, you can use it to revert to a working version if you encounter problems. You should turn off any firewall configurations you may have created on these VMs (from previous labs: `sudo ufw disable`).

Figure 1: Required Lab Topology

This lab will use the preloaded MySQL Database server and Apache Web server. Make sure those services are actively running on VM2Server:

```
$ sudo service mysql status
(sudo service mysql start|restart)
$ sudo service apache2 status
(sudo service apache2 start|restart)
```

3 Lab Part 1: Database Servers

Databases can present challenges to security. Database implementations can be complex and involve interactions with many other services. Remember, complexity is the enemy of security (a violation of economy of mechanism). Most database systems include an internal set of roles, privileges, and security settings separate from those of the host operating system. These settings must be configured correctly to ensure confidentiality, integrity, and availability of the information the database contains.

Further, databases tend to contain high-value information that an adversary might want to steal, manipulate, or destroy. Examples include credit card information, medical patient records, or situational awareness tracking data for a military command and control system.

You will use the popular MySQL RDBMS (relational database management system) product to explore some of the issues with securing databases. SQL stands for Structured Query Language and is the most widely used database programming standard in use today. In addition to MySQL, the SQL standard is used by Oracle and Microsoft products. MySQL as a product is used commercially by many of the largest companies in the world to manage customer and product data with web front-ends. An unsecured MySQL server is a huge vulnerability.

This lab illustrates a few select database security concepts. In a production environment, you should deploy a fully secured database. For example, the National Vulnerability Database provides a checklist for server implementations: `http://www.nvd.nist.gov/view/ncp/repository`

The main MySQL website contains a good explanation of detailed operation, available at `http://www.mysql.com/`

3.1 Basic MySQL Configuration

For Part 1 of this lab, you will be configuring the MySQL database on VM2Server. MySQL has a command-line "shell" interface. MySQL is a database that stores all run-time information about the database (individual databases, user accounts, permissions) as yet another database within it.

- MySQL's configuration files can be found in `/etc/mysql`

3.1.1 Database Interaction

One change we want to make is to control external network access to MySQL. In a production environment, the database server would be installed on a computer on the internal network and access would be restricted such that a web server was the only externally-facing computer able to communicate with it. This would be accomplished with a combination of firewall settings, SSH, and/or IPsec. In this exercise, our web server is on the same machine as the database server, so we set up MySQL to only listen for network connections from the `localhost`, rejecting all others (see Figure **??**). Note: it is also possible to disable MySQL networking completely and use only *named pipes* for inter-process communication. However, some applications will not support this.

- In the MySQL configuration file, verify that the `bind-address` is the localhost address, `127.0.0.1`
- Using linux commands, verify that your VM is listening on `127.0.0.1`.
  (Hint: you learned how to do these things in Lab 1).

Question 1: Which file is the main MySQL configuration file? What port is MySQL listening on? How did you determine this? How should you filter access to this port, if at all?

3.1.2 Server user

Determine what userID runs the `mysqld` server process. Verify this by checking the running process. Note that we could specify a different user in the mysql configuration file, if needed. However, the default will work for now.

Question 2: Under what user name is `mysqld` running? What is the numeric user ID value for this account? Why would we NOT want to run `mysqld` as root?

3.1.3 Database Availability

Now that we have done some hardening on the host system, let's look inside MySQL. For most of the rest of this part of the lab, you'll be interacting direcly with MySQL inside the command shell. *You must end all commands to* `mysqld` *with a semicolon as shown below!*

- Open a terminal window and enter: `$mysql -u root -p` to connect to the local MySQL DB. Enter the password `'toor'` when prompted. (Note: you can use up-arrow to re-run commands in your mysql session history.)
- Check the status of the MySQL server:
  `mysql> status;`
- Check what databases are present on this server:
  `mysql> show databases;`
- The `mysql` database contains master control information for the database server. This includes a set of roles and privileges separate from those of the host system. You can see what is contained in this database by entering:
  `mysql> use mysql;` ...to make the `mysql` db active
  `mysql> show tables;` ...to see the tables in the `mysql` db

3.1.4 Database Users

Database services such as MySQL have their own internal set of user roles that control who can interact with the database and the actions they can perform (select, insert, drop, etc.) These accounts are not the same as the accounts on the underlying operating system. We typically refer to these roles with a `<username>@<hostname>` format. This allows for assigning specific privileges to users on the local system as well as users logging in from remote systems.

- You can see all the possible data fields for a database role with (you may want to maximize your bash terminal):
  `mysql> describe user;`
- Now examine the roles set up in MySQL:

```
mysql> select user, host, authentication_string  from user;
```

Question 3: What did this query return? Provide a screen shot.

Question 4: Examine the results in the password field – does MySQL use password salts? How can you tell?

- Sometimes there is a `null` entry in the user table. This permits anonymous access to MySQL. Unless we really want to allow this, we should purge this null account. The commands below will remove all roles other than `root@localhost`. BE EXTREMELY CAREFUL WITH THESE COMMANDS. Do not do this to a production DB unless you know you are not about to cripple the system. Enter:
```
mysql> delete from user where not (host="localhost" and user="root");
mysql> flush privileges;
```
- The database service typically has a `root` role with complete control over the server. We would normally want to rename `root`, possibly adding a crippled, decoy `root` account, similar to renaming Administrator on Windows systems. We could use the `root` role to interact with this database, but that would be a gross violation of least-privilege. Now create a few limited user accounts:
```
mysql> grant insert,select on phpmyadmin.* to cs482@localhost
identified by '284sc';
mysql> grant select on phpmyadmin.* to CadetSmith@localhost
identified by 'abc123!';
```
- Now enter:
```
mysql> select user,host, authentication_string from mysql.user;
```
- Enter:
```
mysql> show grants for cs482@localhost;
mysql> show grants for root@localhost;
```
- Compare the privileges granted to these two users. Note that these database services allow for very fine-grained control of the ways users can query and alter the database.

Question 5: Provide a screen shot of your work.

3.1.5 Individual Databases

While there is a single `mysql` process, it manages many databases, each with their own sets of users, tables, data, and individual permissions for those users.

- Now look closer at a sample database:
```
mysql> use xss_collabtive_db;
mysql> show tables;
```

You should now see a database schema for the `xss_collabtive_db` database.

Question 6: What tables are in the `xss_collabtive_db` database?

- Now select the database:
```
mysql> use xss_collabtive_db;
mysql> select * from user;
```

Question 7: Who are the users in this table? What conclusion could you possibly infere about `admin` and `alice`? (Hint: widen your terminal window to see the table formatting better.)

Note that these are not MySQL users nor are they user accounts on the Ubuntu OS, they are user accounts for the web application that references this `xss_collabtive_db` database. This is an additional level of things that would need to be secured.

3.1.6 Individual Users

Now examine what you can do with the `cs482` user.

- Close your `root` session with the database:
```
mysql> exit;
```
- Then reconnect to the database as cs482. When prompted, use the password you created for the account above:
```
$mysql -u cs482 -p
```
- Now look at the databases table:
```
mysql> show databases;
```

Note that `cs482` cannot even see the `mysql` database. This user does not have the authority administer the database service.

- Try to drop the `phpmyadmin` database:
```
mysql> drop database phpmyadmin;
```

Question 8: What is the significance of drop? What error number do you get? Why?

Log out of `mysql` to end this part of the lab.

4 Lab Part 2: Web Servers

Web servers are a common source of network vulnerabilities. A poorly configured web server can represent a gaping security hole. Moreover, it is not uncommon to find web servers that were installed to support an application or an outdated function that are left unmanaged or neglected.

For Part 2 of this lab, you'll be reconfiguring the Apache web server on VM2Server. You'll be testing various changes to this web server using Firefox on the VM2Server and also on VM1Client. Take a few minutes now to disable caching on your web browser so that every time you reload a web page, the browser contacts your web server for it instead of using a local version of the page.

- In the Firefox address bar, type `about:config` to get a page of various preferences that you can change. You will first be presented with a warning, click "I'll be careful, I promise!".

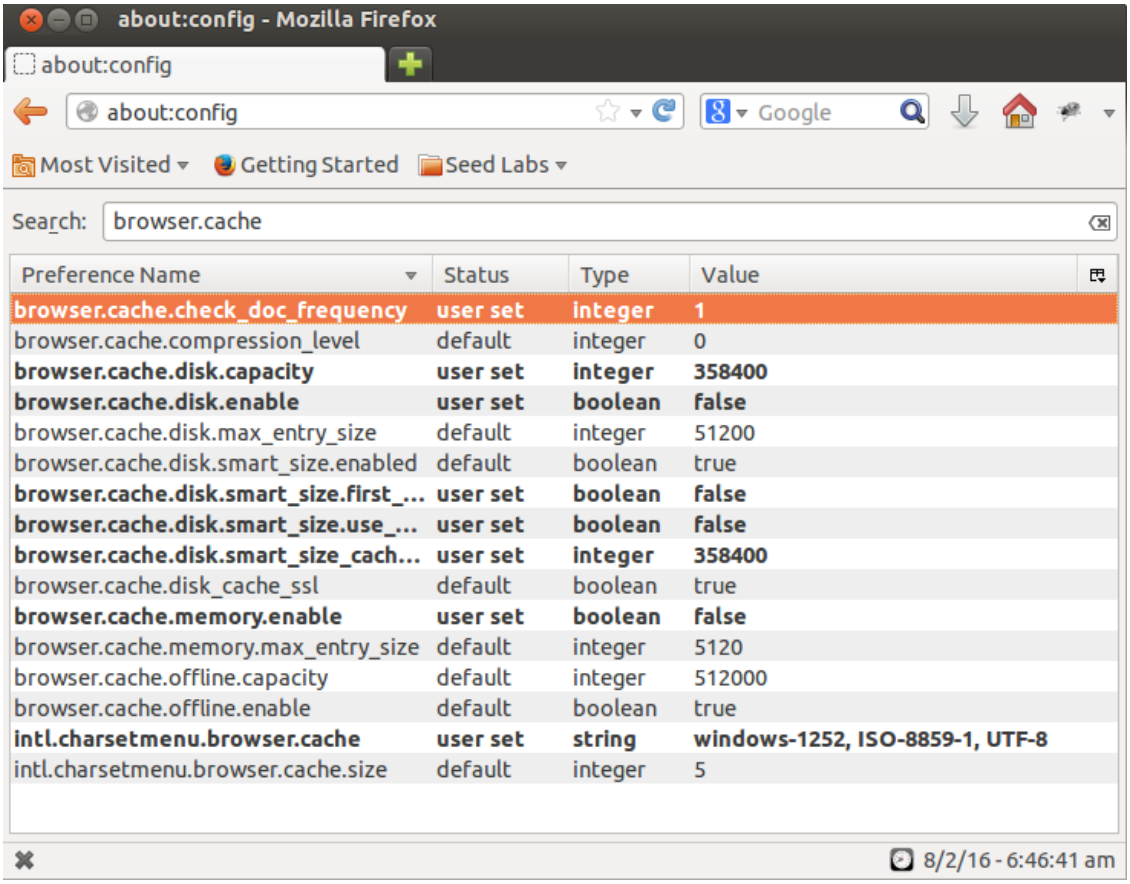

Figure 2: Configuring the Firefox browser cache

- In the "Search:" at the top, type `browser.cache` to filter for the cache options (see Figure 2).
- Locate and change entries as follows:

  1. `browser.cache.memory.enable` – double-click to set the Value to "false". This will turn off browser caching in memory.
  2. `browser.cache.disk.enable` – double-click to set the Value to "false". This will turn off browser caching on the disk drive.
  3. `browser.cache.check_doc_frequency` – double-click to open a dialog box to change the frequency, set this to 1 and click "OK". This will force the browser to verify a page each time you load it.

Make sure to do these changes on both of your VMs!

To ensure that you are only accessing the webserver on VM2Server, make sure to turn it off on VM1Client:
`sudo service apache2 stop`

4.1 Secure a Web Server.

Apache is a powerful, open-source web server produced by the Apache Software Foundation. Apache is available for a wide range of operating systems including *nix and Microsoft Windows, serving about 60%

of all web sites. We will work with Apache in this lab. The principles covered here also apply to Microsoft's Internet Information Server but the interface and details differ significantly. In any case, this lab illustrates a few key concepts; you should consult a detailed security configuration guide when configuring a production web server. Learn more at `http://httpd.apache.org/`

4.2 Basic Apache Configuration

The configuration files for Apache are located in `/etc/apache2`. In the canonical Apache installation, the master configuration file is `httpd.conf` (not present). The Lab Ubuntu VM has the Debian variant of Apache installed, which has expanded the configuration options. A missing or empty `httpd.conf` means several other files manage the configuration. This variant allows "name-based virtual hosting" which means that one web server can serve multiple websites when accessed with different URLs. Some configurations are global to all of the sites hosted by the web server. Some configurations are specific to the virtual host for a specific website. (Hint: the linux command `grep` is useful for searching for items in files. `man grep` for more details, particularly how to search through a subdirectory tree.)

These are the key configuration files:

- `apache2.conf` – contains the master configuration parameters common to all hosted sites
- `envvars` – contains some host-specific execution parameters for the `httpd` process, used in combination with apache2.conf
- `ports.conf` – the specific ports that the webserver listens on for various virtual hosts
- `/conf-available` – contains additional global configuration files for the webserver.
- `/mods-enabled` – this directory contains the various optional modules that add features to the webserver. Each of them has a module-specific configuration file.
- `/sites-available/000-default.conf` – this file describes the various virtual hosts supported by this webserver and specifies configuration and access control lists for each of them.

You may need to look in or modify several of these files to complete the tasks and answer the questions below.

4.2.1 Webserver availability

Locate the `Listen` option. It should be set to 80. The directive `"NameVirtualHost *"` instructs the web server to use all IP addresses and ports on the machine for the given site (some machines may have multiple IP addresses). Each website has a Virtual Host block in `/sites-enabled/000-default` that specifies the URL for the website and the directory in the file system that contains the source files for the website. The DNS on VM2Server is configured so that all of those website URLs point to `localhost`, so you should be able to load any of them in your browser on VM2Server.

Question 9: What file is the `Listen` option in? What does port 80 mean? How many websites are available through this port? What do you have to change to get `www.wtmobilestore.com` to respond on port 9000?

4.2.2 Server user and group

Locate the server `User` and `Group` options. These specify which system user and group owns and runs the webserver `httpd` process. It is an extremely bad idea to run a web server as `root`. Fortunately, as you can see, Apache is configured to run under another context.

Question 10: Under what user and group is Apache configured to run? Where did you find this?

4.2.3 Modules

The `/mods-enabled` directory specifies the modules Apache will use. These modules add features such as authentication, encryption, enhanced logging and support for languages such as php and Ruby to the Apache core functionality. One popular third-party module is modsecurity, provided by Trustwave SpiderLabs.

Question 11: What is modsecurity? Is it compatible with our Apache version?

4.2.4 Administration and naming

- Locate `ServerAdmin`. If you wish to have a point of contact e-mail address for the web server, it is set here. This should be set to minimize disclosure of sensitive information, since it may appear in various error messages by default. (Also, it may become a spam magnet...) Set the `ServerAdmin` address to: `cs482@mail.cs482lab.cdx` for this exercise.
- Locate the following parameters and configure them as follows:
  ```
  ServerSignature Off
  ServerTokens Prod
  ```
  These options suppress the server version number in page footers and HTTP responses, respectively. The server replies simply with 'Apache', making life a bit harder for a would-be attacker.
- Locate `ServerName`. This is the FQDN (fully-qualified domain name) the server uses to refer to itself. Change this to read: `www.cs482lab.cdx:80` (Note: you can also add this below `ServerAdmin`)
- Save your files and reboot Apache:
  ```
  sudo service apache2 restart
  ```

4.2.5 Content locations

Locate `DocumentRoot`. This specifies the base location for the web pages and other content. If you want to store the content in a different location it is important to update this setting.

Question 12: What file did you find it in? What is the `DocumentRoot` setting for this server? What are the websites configured to run on this webserver?

- Save your changes to the various configuration files and restart apache.
  ```
  sudo service apache2 restart
  ```

- On VM1Client, use Firefox and try to access your webserver using the VM2Server IP address to verify that it is working (`http://10.172.xxx.bbb`). If you do not get the "It works!" page on VM1Client, do not continue with the lab until your webserver is running properly.

4.2.6 Setting Site Access Policy

Apache allows for access controls on the directories that constitute the web site(s) served. It is possible to exercise very fine-grained control over access to the site content. Now look at a few settings to get a feel for this.

- Open the file that specifies `DocumentRoot` again and look for the option `< Directory />` This is the default policy for the `DocumentRoot` and everything below. We set this to deny everything by default, then add specific exceptions later. This helps prevent access to unlinked pages or objects and other items we do not actually intend to publish. The default policy is to block. We can make it a bit stronger by setting:

```
<Directory />
        Options None
        Order allow,deny
        Deny from all
        Satisfy all
        AllowOverride None
</Directory>
```

We will see what these statements mean shortly.

- Scroll down to the directory policy entry for your 'document root' directory. This is a fairly permissive policy, so we will lock it down a bit.

The `Options` line controls some of the behavior of this directory. The `Indexes` option controls how the server responds if no page is specified in a requested URL. Apache will return the `index.html` default page if it exists. If `index.html` is not found and the `Indexes` option is set, Apache will still return data. Give it a try:

- Go to the document root directory on VM2Server and rename the `index.html` file to ``<yourname>.html''.
- On VM1Client, try to access the webserver again.

Question 13: What did the web server return to the browser this time?

This could disclose information about the internal structure of the site and should normally be suppressed.

- Delete the word "`Indexes`" from the `Options` line.
- Save your changes and restart Apache.
- Then reload the VM1Client browser again.

Question 14: What did the server return with the `Indexes` option deleted?

- Ensure you are still be able to see the homepage at: `http://10.172.xxx.bbb/<yourname>.html`
- Leave `AllowOverride None` as it is. We will examine this statement in a moment.
- The next setting is `Order`. This specifies the order in which "allow" and "deny" statements are applied for access control and the default behavior.

  – `Order allow,deny` applies all "allow" statements first, then any "deny" statement, with a default "access denied" if no allow match applies. Use this if you want to allow access from a few specific sites or users and deny all others. Leave the setting this way.

  – `Order deny,allow` is the opposite, with a default ALLOW if a matching "deny" is not found. Use this where most everyone is allowed, such as a public web page, but specific sites or users will be denied explicitly.
- `Allow` and `Deny` statements control access by single IP address, network (in CIDR notation), or a domain (this is where those DNS reverse lookup zones come into play).
- Change the `Allow from all` statement to `Allow from 127.0.0.1`
- Save this change and restart Apache.
- On your client machine, attempt to access `http://10.172.xxx.bbb/<yourname>.html` again (or just try to refresh the page).

Question 15: What response do you get from the web server now?

- Change the allow statement to read:
  ```
  Allow from 10.172.xxx.0/24
  Allow from 127.0.0.1
  ```
- Save this change, restart Apache, and reload the site on VM1Client. You should get the pages again.

4.2.7 Setting Per-Page Access Policy

Sometimes it makes sense to apply global access policies to all of the content on a webserver, but individual content creators want to create more specific policies for some files. In a web content folder the `.htaccess` file is used to delegate authority to web site developers, allowing them to control aspects of site access without needing the ability to edit `httpd.conf` or restart the web server. (Remember "least-privilege"?) The `AllowOverride` option lets us control if, and how, an `.htaccess` file in a content folder is allowed to override the policy set in the master server config files. If you will not be delegating authority, or are not sure, set `AllowOverride None`, to prevent a stray or malicious `.htaccess` file from opening a hole in your site.

For this task, you'll be setting policies for one of the sample websites included with the VM2Server webserver.

- In the VM1Client browser, load the website:
  `http://10.172.xxx.bbb/webtracking/CameraStore`

- Look at the page to familiarize yourself with what it looks like.
- On VM2Server, after the end `</Directory>` of the "document root" directory policy, add a new directory policy:

```
<Directory /var/www/webtracking/CameraStore/images>
        Order allow,deny
        AllowOverride Limit
</Directory>
```

- Save your changes and restart Apache, then reload the site in the VM1Client browser. You should see that the image files are no longer rendered.

Question 16: Provide a screen shot.

Since we added an `AllowOverride Limit` statement, we can add an `.htaccess` file in the images directory to override the policy and allow access by IP address, network or domain.

- Go to the "images" directory listed above and create a file called `.htaccess`
- Add this text to the file:
  `Allow from 10.172.xxx.0/24`
  `Allow from 127.0.0.1`
- Save the file.
- Reload the site in the VM1Client browser. You should now see the images rendered. (Note that you do not need to restart Apache for this change to take effect; the `.htaccess` file is read when the `http GET` is processed.)

4.2.8 Password Authentication

Finally, we will look at password-protecting the site. Note that the standard Apache install supports only Basic and Digest authentication. Apache modules supporting stronger authentication methods are available. Internet Information Server also supports stronger authentication, especially within a Windows AD domain.

The Apache password files consist of `username:password_hash` pairs. It is possible but HIGHLY DISCOURAGED to use the `/etc/shadow` or `master.passwd` file for your web site authentication. We can create a new web site password file with the `htpasswd` command.

- Create a new password file and a user account for Scooby with:
  `sudo htpasswd -cs /etc/apache2/.htpasswd-users scooby`

  (Note that the -c option creates a new file, overwriting any existing file, so only use -c the first time.)
- Set Scooby's password to: `snacks`

Question 17: What does the -s option for htpasswd mean?

- Now add an account for Shaggy:
  `sudo htpasswd -s /etc/apache2/.htpasswd-users shaggy`
- Set Shaggy's password to: `zoinks`
- Now add password authentication to the web site policy. Edit the main `000-default` config file. In the directory policy entry for your "document root" directory, add these statements:

```
<Directory /var/www/>
    AuthType Basic
    AuthName "Mystery Machine"
    AuthUserFile "/etc/apache2/.htpasswd-users"
    Require valid-user
    Order allow, deny
    Allow from all
</Directory>
```

- Save the config file and restart Apache.
- On VM1Client, start Wireshark and capture traffic on the main interface.
- Reload `http://10.172.xxx.bbb/<yourname>.html` in the VM1Client browser. You should receive a dialog box asking for a username and password for "Mystery Machine". Enter one of the accounts you created.
- In your packet capture, find the HTTP GET request packet after the AUTH request packet. Look at the Authorization: Basic field under the HTTP protocol, you should see that the user name AND password were easily sniffed by WireShark. This is one reason why using your system's shadow file for basic web site authentication is discouraged. NOTE: Don't worry later in this course we will create certificates that will enable us to use SSL i.e. HTTPS.

Question 18: Provide a screen shot.

4.2.9 Web Server Logging

Apache maintains an access log, an error log, and a referrer log. These are useful for troubleshooting and monitoring web site activity.

- Look at the most recent entries in the main log file:
  `tail /var/log/apache2/access.log`

Question 19: What browser and operating system versions did Apache record for your connection from VM1Client?

An additional step to secure these services would be to `chroot` them. That process is beyond the scope of this lab, but is an excellent addition for security of a production system (*e.g.*, CDX)

Question 20: What is `chroot` ? Describe one way `chroot`-ing Apache and/or MySQL would improve security.

There is a great deal more to security for production web sites. This should provide an introduction to some of the issues and give you a start in learning more about this topic.

5 Submission requirements

5.1 Partner Submission

Provide one written lab report, answering each question properly labelled with the number and original question, per partner team. Be sure to include the time spent on the lab and document any external resources used. Again good documentation:

1. clearly enumerates tasks with a description of you did and evidence.
2. shows the progress you were able to achieve.
3. explains your troubleshooting attempts.
4. accurately describes an issue and the potential solution (if really good, I will give near full credit).

5.2 Individual Submission

Each member needs to submit a detailed lab reflection. This includes

- approximately one half page that talks about the various security issues and princples from Chapter 2.
- any challenging points or thoughts on what you found interesting during the lab
- time spent you personally spent and how much effort you put forth
- time your partner spent, and how much effort they put forth
- be sure document any external resources used.