- **Confidentiality**: Find two specific settings that aim to improve the confidentiality of the server and it's content - either at rest or in transit.
- One: Install a Valid Trusted Certificate
    - In a few sentences for each, describe the settings and what they do.
        - Installing a Valid certificate ensures that the site is secured and the user is communicating with the site. Without a valid certificate, it is very possible for a malicious site to attempt to steal the users data. Having a certificate that is signed with a trusted authority's signature is an easy way to tell if a site is secure.

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - As we did in the lab recently, you first must have two boxes, a CA and the webserver. Then you would create a certificate using openssl, and have the CA sign it and send it back to the webserver. Then restart the webpage services and the site should now be using HTTPS://
- Two: Disable WebDAV Modules
    - In a few sentences for each, describe the settings and what they do.
        - The WebDAV is the "Web-based Distributed Authoring and Versioning". This is used as an extension to HTTP that allows clients to edit and delete files and other data on the web server. Disabling this would help reduce the amount of vulnerabilities that the server has. It stops vulnerabilities that are exposed to the network and stops unexpected access to the webserver files.

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - Open the httpd.conf file within the webserver and add in the DAV modules. The context in the file:

```
##LoadModule dav_module modules/mod_dav.so
##LoadModule dav_fs_module modules/mod_dav_fs.so
```

- **Integrity**: Find two specific settings that protect the server and its content from data manipulation or destruction.
- One: Public/Private Keys
    - In a few sentences for each, describe the settings and what they do.
        - Public and Private keys are one of the most important ways to protect data that should only be accessed by a select group of users. The public key is *publicly* available and is not as important to keep hidden away. The private key is just that, private. The private key is used for user authentication and makes sure that only the proper users have access. Limiting this access is one way to ensure integrity of the webserver as only authorized users can make changes.

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - When creating a private key, make sure that it is hidden in a place that is not easily accessible. For example, keep it in /home/david/keys/private… rather than just throwing the private key file into the /etc directory.
- Two: Restrict the users permissions
    - In a few sentences for each, describe the settings and what they do.
        - On all files within Linux, there are read, write, and execute permissions that allow the users to interact with the files in specific ways. Read access allows the user to see, but not edit the file, write allows you to edit it but not execute it, and execute allows the file to be run. Giving proper users the proper permissions allows the administrators to make sure that not just anyone can go in and change a bunch of files.

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - This is relatively simple to do as the administrator just has to remove group access from all the files with the apache prefix. This is done by using chmod and if more specific permissions need to be made, this can be done within the boxes as an elevated user.

```
# chmod -R g-w $APACHE_PREFIX
```

- **Availability**: Find two specific settings that protect the web server from service disruption or degradation.
- One: Denial of Service Mitigations (Changing the TimeOut to a lower Value)
    - In a few sentences for each, describe the settings and what they do.
        - A DOS is a very common cyberattack that is meant to limit access or slow down a server. By flooding the server with many connections and requests to the server at the same time, the server is often slowed down or

completely taken down. In order to stop all of these connections, we can lower the timeout value within the configs in order to get rid of old connections in a more efficient way.

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - In the apache config file, find the *Timeout* Value (that is defaulted to 60) and change it to 10 or less so that new connections (if inactive) will time out sooner and cause less strain on the server.
- Two:Limiting the request line
    - In a few sentences for each, describe the settings and what they do.
        - The request line limit limits the size of the request line. I know that sounds obvious, but it stops unreasonably sized request from coming into the server. Making this limit however is also a delicate line to walk as you dont want to make the limit too small as to cause issues during common use, but also dont want the limit too big as that could lead to malicious attacks such as Buffer Overflow attacks (providing more data than the application buffer can handle)

    - Explain (not cut/paste) how to implement the settings on an Apache Server
        - In the Apache config file, change the *LimitRequestLine* Value to 512 or shorter. It is defaulted at 8190 and that is much too big. This can be edited to fit the users needs on a server-by-server basis.