Mylo Counts

Dr. Wang

CIS437: Secure Coding in Cloud Environments

May 2, 2024

CIS437 Final Project

**Intro**

I built a user handling system across a distributed environment with security considerations. This includes login, registration, as well as admin functions such as password changing, user promotion, and user demotion. This was accomplished using two virtual machines running Ubuntu Server: an Apache2 web server to handle client interaction, and a MySQL database server to handle the user registration & alteration.

**Design**

My stack includes a Windows-based Chrome browsing client, an Apache2 web server, and a MySQL database server. The web server & MySQL server are ran on separate instances of Ubuntu Server with SSH built-in. I reused user handling code I wrote for my CIS215 final which allowed me to focus on security implementations.

ChatGPT was used to get me on the right track in terms of designing the web & database VM servers. Apache2 & MySQL were chosen as I have familiarity in these tools.

In terms of security, OpenSSL was used to secure the client’s connection to the web server. Then, password data was encrypted using SHA224 encryption. These two processes allow for end-to-end encryption of the sensitive data. Additionally, ufw firewall built into Ubuntu was used on both servers to ensure only critical ports were open (SSH, HTTP, HTTPS, and MySQL).

Validation of data in the registration & admin command pages is done using PHP on the server-side.

SSH was used for secure shell connections,

**Implementation**

Ubuntu Server was installed on separate VMs. The web server utilizes Apache2, PHP & PHP’s mysqli to allow web clients to access the database. The database server utilizes a MySQL table based on username, password(encrypted in SHA224), and whether the user is admin based on a binary value. mysql-secure-installation was used to remove default users, remove the test table, ensure password strength, and denying remote root access.

ChatGPT & StackOverflow were instrumental in debugging & finding solutions for server errors. VSCode was used to build the scripts, and WinSCP used SSH(which was included in the VMs during Ubuntu Server installation) to facilitate file transfer & modification.

I implemented the PHP functions and a basic login page first to establish an end-to-end connection between my client, the web server, and the database server. Additionally, I also created & loaded data into the userauth.users table.

Getting end-to-end connection was the most difficult task, as many errors arose. The Apache2 error log file found in /var/log/apache2/error.log was used to debug & locate the necessary configuration files to solve the issue. To solve, PHP-MySQL need to be both installed and configured(**/etc/php/8.3/apache2/php.ini)** on the web server to properly reference the mysql connection function in PHP. Then, MySQL also needed configuration on the database server to accept connections from the web server using the following MySQL commands:

CREATE USER ‘loginHandler’@’192.168.182.135’ IDENTIFIED BY 'S3cur3Pa22';

GRANT ALL PRIVILEGES ON userauth.\* TO ‘loginHandler’@’192.168.182.135’;

FLUSH PRIVILEGES;

After finishing end-to-end functionality, I imported the user registration & admin command php files to make sure the users table was being read & altered appropriately.

A diagram of a web page

Description automatically generated

Next, I improved the security of the web server by installing OpenSSL & creating a self-signed certificate to allow for HTTPS connectivity to the web server. This was the second biggest hurdle, as certain Apache2 mods needed to be enabled for HTTPS connectivity. After enabling the SSL module using the following command:

sudo a2enmod ssl

the apache2 configuration file(apache2.conf) was then edited to define the server name & file paths to the server & SSL keys.

After completing the end-to-end connection and the secure connection between client and server, I then ensured a secure data transfer between the server and database by encrypting password data in SHA224 using a password key. This was done by lengthening the users.password column to varchar(56) to allow for the added character length, and then the population.sql & user\_auth\_fns\_userauth.php files were edited to all use SHA224.

In the end, the application fulfilled all expectations with one minor exception: the Apache2 service has trouble starting up after booting the web server, and must be stopped & started again using the systemctl command. This may be due to a misconfigured configuration file associated with SSL.

**Security Analysis**

mysql-secure-installation:

* Protects the database server by removing the test user
* Removes the test database
* Disables remote login of root user
* Allows for password security settings(although I did not apply this in my configuration)

SHA224 password encryption:

* Protects the transfer of sensitive password data between the web server & the database.

OpenSSL:

* Protects the transfer of information between the client and the web server using a self-signed certificate. This type of certificate allows for a degree of safety on the local network, but can result in insecure data when applied online. However, without a proper validator which costs money, this is still a good option.

**Conclusion**

I was successful in building my application which allows for users to register, log in, and for admins to change their password, promote users, and demote users. After designing my application, I built it in small parts to guarantee a working end-to-end connection while debugging utilizing ChatGPT. After completing the distributed environment, I implemented safety features such as utilizing mysql-secure-installation to protect my database server, implementing HTTPS using a self-certified OpenSSL key, SHA224 encryption for database password data, and an Ubuntu ufw firewall to ensure protect the network. These safety features allowed for theoretical end-to-end encryption of the sensitive data, with HTTPS encrypting the client-to-web connection and SHA224 encrypting passwords in the web-to-database connection.