**Implementation of application security with authentication and network security by firewall configurations.**

Rimma Shilkina   
 Computer Science Department of College of Engineering  
 University of North Texas Denton, TX, USA

rimmashilkina@my.unt.edu

07.15.2021

**Introduction**

In the modern world people and organizations do not exist isolated, there are a lot of communication networks between them. To protect these networks and to ensure main principles of cybersecurity – confidentiality, integrity, and availability – they should be secured.

According to the SANS Institute, network security is the process of taking preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction, or improper disclosure. Implementing these measures allows computers, users, and programs to perform their permitted critical functions within a secure environment. In other words, unsecured networks are vulnerable to malicious attacks. In most cases, data, communicating through networks, is what interesting for hackers; but hardware damages and systems’ manipulations could be the goal of networks security attacks also. A couple cases could show importance of network security. In 2013 a retail company Target was attacked, attackers used Internet-enabled air-conditioning systems attached to the Target’s networks to get access to the network. From there they were able to compromise payment data for up to 70 million customers. Another example of the network security attack is the Capital One breach in 2019. Misconfiguration of a firewall allowed attackers to collect data of about 100 million customers.

There are several tools to ensure network security: firewalls, Intrusion Detection systems/ Intrusion Prevention Systems, honeypots, etc.

A firewall is a device that filters all traffic between a protected or “inside” network and a less trustworthy or “outside” network. Firewall is the fundamental network security tool. There are several types of firewalls: packet filtering gateways or screening routers, stateful inspection firewalls, application-level gateways (proxies), circuit-level gateways, guards, and personal firewalls. These types differentiate based on what kind of attack then can detect and the way of detection.

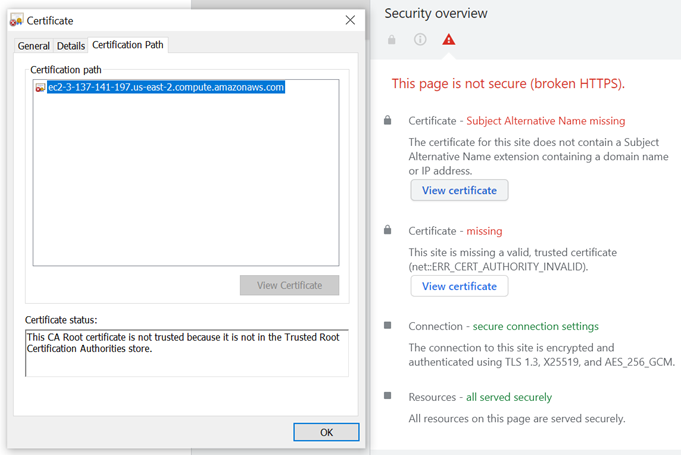
The goal of the project is to implement network security concept by creating a web application and configuring a firewall. For the first part of the project a simple website was created. It uses authentication as application security tool to allow users see pages with restricted access. Usernames and passwords are stored in a database. Passwords in the database protected with a hash function. For the second part of the project a built-in firewall was configured to allow specific communications, i.e., it will filtrate inbound traffic based on specified rules. This “filtration” will solve problem of protecting network from undesirable (malicious, in the worst case) communication. Because a webserver is hosted at AWS, a security group was configured to allow same network traffic. This is additional protection layer on top of the firewall on the server itself.

The project was designed to show importance of proper configuration for network security. It also provides basic ruleset for securing a webserver. The project can be used as basis for other projects that involve creating and securing a website.

**Methodology**

The project consists of three technical parts: creating a database, creating a website, and configuring a firewall.

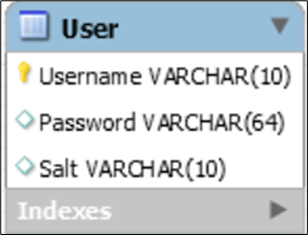
The website is hosted at AWS’s EC2 instance (Ubuntu server 20.04), it can be accessed via http://ec2-3-137-141-197.us-east-2.compute.amazonaws.com or https://ec2-3-137-141-197.us-east-2.compute.amazonaws.com. Apache server is configured to automatically redirect HTTP protocol into HTTPS protocol. The project uses self-signed TLS certificate.



Picture 1: A screenshot of TLS certificate.

*Database*.

MySQL version 8.0.25 database server was used to create a database “mydb” with a table “User” with three attributes: Username, Password, and Salt. Datatype for all of the attributes is varchar. “Username” stores unique identifiers of users. “Password” keeps hashed passwords with added salt. “Salt” is a number randomly generated in time of user creation. MySQL server is configured to allow only connections from the localhost (local server).



Picture 2: A table “User” created in MySQL.

*Website.*

For this project, a simple website was created. It uses Apache 2.4.41 web server, PHP 7.4, and MySQL to provide necessary functions. PHP was chosen because it provides user interactions with a database; PHP is a server-side language, it dynamically generates html-code without disclosing page’s logic. All components are hosted on the single Ubuntu 20.04 server.

Four PHP pages were created for the website. One is to provide user authentication (user\_login.php). Another is a page that presents to a user after successful authentication (user\_page.php). Also, there is a page add\_user.php which is used for creating users. There is a simple index.html file which redirects user to a user\_login.php page.

User\_login.php sets a PHP session cookie to prevent unauthenticated users from accessing user\_page.php. PHP code at user\_login.php reads Username and Password through a form and sends it through POST method to the same user\_login.php. A PHP code verifies that POST message is received, then it reads parameters from the POST. After that, it reads hashed password and salt for the username from MySQL database. Using SHA256 function it hashes password (from the form) and salt (from the database) and creates a new hashed password. If this new hashed password is the same as in the database, then it is possible to successfully authenticate the user. If hashes do not match, the PHP code shows login form once again.

User\_page.php also checks if a cookie is set. If it is not set it redirects a user back to user\_login.php. If user successfully authenticated at user\_login.php, it shows a message “Successfully authenticated”.

Add\_user.php page is used to create users (to add them in the table “User”). PHP code in this file reads username and password and generates a random salt. It creates a hash (SHA256) from the supplied password and a salt and creates a record in the database with username, hashed password, and salt.

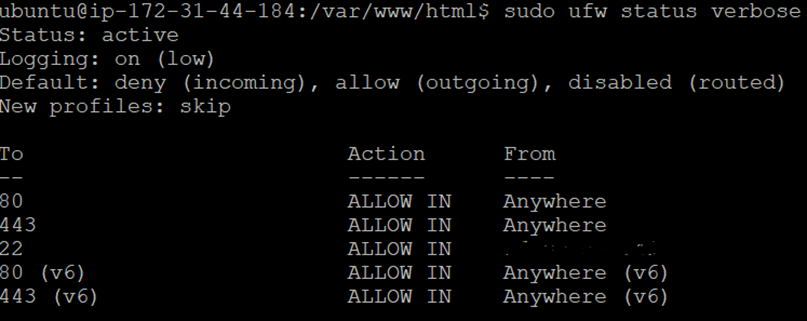
There is also a file secret.php in a subfolder no\_access. This file provides connection string for the database, and it contains sensitive data - username and password - for database connection. Permissions to this file set to 711, which provides full access to an owner and only execute permissions for everybody else. This prevents from reading content of the file.

*Firewall.*

Ubuntu server has default firewall (IPTables). It is installed in the system, but not enabled. Default policies for the firewall are: deny all incoming connections, allow all outgoing connections, and disabled for routed packets. There is a command-line utility to configure IPTables – UFW.

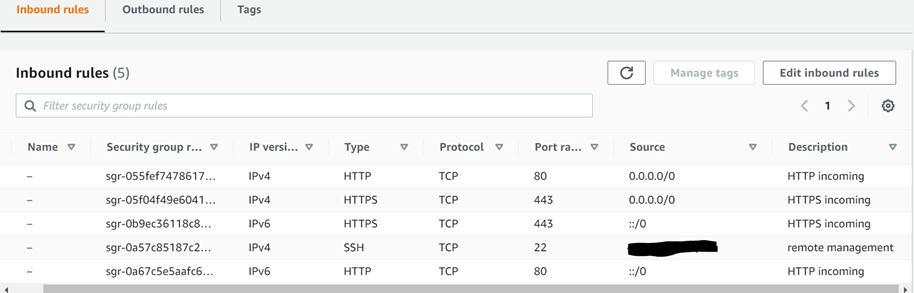
For this project HTTP traffic should be allowed to connect from outside to the website (incoming connections). Connections from any IP-address were allowed to port 80/tcp. For future use HTTPS (443/tcp) was allowed.

Also, for code deployment and remote management SSH protocol is required, but SSH access is limited to only management / deployment / code developing workstation. A rule for SSH was created to allow SSH traffic (22/tcp) from an IP-address.



Picture 3: Screenshot of UFW configuration. Management IP-Address is marked out.

One more tool to enhance security was used: security group enabled on AWS to restrict access to the server. This provides additional security for the server.



Picture 4: AWS security group.

**Results**

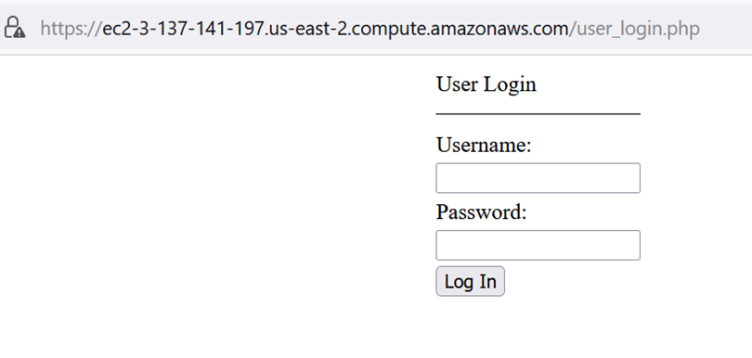
For the demonstration of the project, please, see the next YouTube video:

<https://youtu.be/kaah-AA14v0>

The goals of the project – to ensure network security and to use application security – were successfully reached.

Initial development of the website was done on a local virtual machine – Ubuntu Server 20.04. Then the website was moved to the Amazon Web Services platform, were another Ubuntu server 20.04 was created. The reason is that all settings and codes could be correctly transferred from one server to another without a need to change anything.

The application security was ensured by using authentication for a user to allow it to use the website. The website specifically created for the project is simple and is used for the purpose of demonstration how authentication works and how to ensure its security. There is a page with authentication form and a second page with a notification for the user that the user is successfully entered/opened website, i.e., authentication was successful. Authentication helps to prevent unauthorized users to access the website, and to keep the information on the website secret/restricted from reading.



Picture 5: User authentication page.

During the process of authentication, a PHP code extracts hashed password and a salt for a username from the database. The code calculates a new hash from the password supplied by user via web-page form and the salt from the database records. If new hash and stored hash match, then the user is correctly provided the password and can be authenticated.



Picture 6: User page in case of successful authentication.

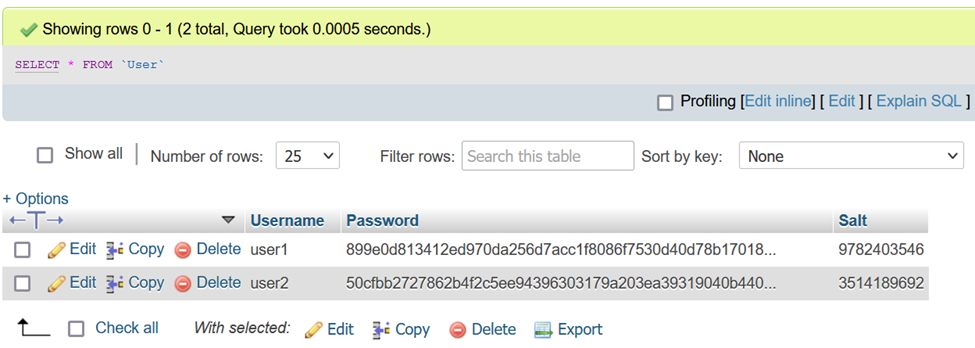
There is another PHP page that was used to create a database of authenticated users.



Picture: Helper page to add a user.

The database is simple: there is a table with three columns, each row in the database represents some user. The database saves user’s name, his password, and a random number, called salt, which is used for enhancing the password. This random salt is added to the password to make it longer, more complicated, i.e., more difficult to break via brute force attack. The password which is saved in the database is not in the form how it would be entered by the user during the authentication process. Each password is hashed by SHA256 algorithm. So, mathematical-logical formula for the password which is saved in the database is:

*Hash of (Password entered by the user + Salt) → Password saved in the database.*

**

Picture: A screenshot that shows content of “User” table.

Network security is ensured by configuring settings in the default firewall of the server. Firewall is a basic and effective tool. For simplicity, firewall works as a protective perimeter of the system with a gateway which allows traffic based on some rules. Most of websites use HTTP protocol with default port 80, and if they use HTTPS, the port would be 443 in most cases. The server’s firewall was configured to pass traffic to these two ports from any IP on the Internet. Also, a rule was added to restrict remote management access to the server. This rule allows connections from a specific IP-address to SSH port (which in most cases is port 22).

Amazon Web Services is concerned about security of their users as well. They provide help and advice on how to secure servers. When the first server in AWS is created, its wizard notifies that remote management port (SSH) is open to anyone on the Internet, and it helps to configure network security settings for the server or any number of servers in the same security group. For this project, the security group allows management (SSH) connection from a specified IP-address, HTTP and HTTPS traffic to come to the server.