ILLINOIS TECH

Application Software Design

ECE 528

Homework 4

Abhilash Kashyap Balasubramanyam

abalasubramanyam@hawk.iit.edu

A20566944

Spring 2025

Illinois Institute of Technology

Date: April 29, 2025

Table of Contents

Questions and Solutions

1 Question 1

Q: Why should we use TLS 1.1 or above but avoid SSL 1.0, 2.0, 3.0, and TLS 1.0? *(5 points)*

Solution

The reason for one to use TLS (Transport Layer Security) 1.1 or above over the SSL (Secure Socket Layer) 1.0, 2.0, 3.0 and TLS 1.0 is that the newer version is known to have fixed the general security vulnerabilities that were prevalent in the older versions. In generality, the purpose of these layers is to provide security over communications and computer networks. In particularity, TLS is the successor to SSL along the gradual advancement of technology.

The versions of SSL 1.0, 2.0, 3.0 are very obsolete to today's standards in terms of paddling oracle attacks, POODLE attack et al. which allows decryption by the attackers thereby manipulating encrypted data which are critical vulnerabilities. Nevertheless, TLS 1.0, although offering an improvement is still susceptible to attacks owing to its vulnerabilities to the BEAST attack where the initialization vectors are generated in CBC mode for block ciphers.

On the other hand, TLS 1.1 brought in a wave of many advancements with its introduction like the explicit initialization vectors for CBC mode to defend against BEAST attacks and also disabling the support for insecure SSL versions. Likewise, with each iterative version of TLS (1.2, 1.3), modern encryption algorithms were added and supported to cater to the discovery of additional vulnerabilities in earlier versions and newer attacks.

2 Question 2

Q: Illinois Tech Wi-Fi system utilizes WPA2-Enterprise and 802.1x for authentication (by utilizing your Illinois Tech email address and password). Describe these authentication methods. *(5 points)*

Solution

WPA2-Enterprise is an advanced Wi-Fi security protocol that is designed for organizations with the purpose of strengthened individual security by requiring each user to authenticate individually with unique credentials (IIT portal login). The WPA2-Enterprise, unlike the WPA2-Personal doesn't use shared passwords but rather a RADIUS server to verify every user's credentials.

802.1x is a protocol for network authentication that acts as a framework that controls the access to the network ensuring authenticity of the user's devices. This happens when each time a user tries to connect, the access point (authenticator) communicates with the user's device (supplicant) using 802.1x protocol and forwards the credentials to the RADIUS server to verification.

The RADIUS server would then authenticate the user's credentials and digital certificate, the access point enables the network port, ensuring the user to join the IIT Wi-Fi network. IIT verification also provides a 2-step verification through OKTA verification via multiple ways (Security question, PIN generator, Notification prompt etc.) which further heightens the security of this process. The combination of both WPA2-Enterprise and 802.1x protocol enables robust security by ensuring only authorized users to be able to connect and encrypting all data transfers using AES and also enabling a centralized management hub of all user access.

3 Question 3

Q: In this course, in order to access the Endeavour git repository, you need to execute setup_ece448528_endeavour script before git. Explain this script, and describe how this script enables you to git without password authentication every time. *(5 points)*

Solution

The setup_ece448528_endeavour script enables us to setup and prepare the environment necessary for the project in terms of the dependencies like JaCoCo, Gradle, libraries (available and locked) etc. It also enables towards the establishment of the handshake and connect to the Endeavour server with the client system so that the client system does not necessarily have to enter all the credentials every time there is a need to connect to Endeavour.

This connection by the script also facilitates the generation of an RSA key exchange between the client and the server (public and private keys) to ensure a secure channel and an authenticated user connection to the server since the user can provide authentication to the server's public key with its generated private key, thereby securing the server from attacks and securing the data transfer channel between the user and Endeavour.

Thus, the run of this script enables the student (client) and Endeavour (server) connection to be secure and seem-less on both ends. The authentication is automatically handled automatically and securely each time, and the client can straight up start using necessary commands like git pull, git add ., git commit, git push et al. without the need for authentication every single time.

4 Question 4

Q: Git is a useful tool to use version control. Currently, you are pushing your code to a 'master' branch. Suppose the course asks you to submit each project code to a specific branch (e.g., Project 1 to project_1 branch, Project 2 to project_2 branch), explain how you would git push your latest code to the Endeavour git repository. *(5 points)*

Solution

Git branches is an option that enables developers to maintain codes of a single project with multiple modules through multiple streams (branches) owing to complexities of the project that the codes from different branches shouldn't interact with each other during development and/or that there are multiple teams/individuals contributing to the project code development and that they need to be merged after a certain stage in development and suchlike reasons.

Currently for the course IoT main project, a single master branch is being used and in a case scenario where multiple branches were to be used for each project (project 1, Project 2, etc.) under the main project, then the first task would be to verify that the branch that the user is working in is established and is correct. This can be done by using the commands '`git branch <branch_name>`' where '`<branch_name>`' is replaced with the name given to that branch like Project_1, Project_2 etc. To switch or check the branch of the current workspace, the commands '`git switch <branch_name>`' or '`git checkout <branch_name>`' is used.

Thereafter, committing to that branch would be the same as the how it is done currently. Like using commands '`git add .`', '`git commit -am "<Commit_Message>"`'. However, while pushing the code, the git command can be something like '`git push -u <branch_name>`' where '`-u`' is a flag that sets up upstream tracking reference thereby linking the local branch with the remote branch – this enables the ease of use of these commands where thereafter commands like '`git pull`' and '`git push`' can be used directly without specifying the branch name.

If a branch doesn't exist in the remote repository, then this push command will create that branch, and the commits would be uploaded there. To verify that a branch is healthy with latest commits, the git web interface (GitHub) can be used, or it can be checked with the command '`git branch -r`'. Thus, pushing each individual project code to their respective branch and then at the end for the final submission, a git merge should enable all the different codes for different projects should work harmoniously with one another.

5 Question 5

Q: Since Git is a version control system, suppose you would like to roll back to your previous commit and discard the latest commit. Explain the procedure of how to achieve this with an example. *(5 points)*

Solution

In the likeliness of entering a rabbit-hole in coding where it so generally happens that the user working might be trying to implement a feature or fix a faulty feature and in that process, he/she would have worked in depths to get that working, only to find out that their previous/other features have stopped working in this process. Thus, for the user to revert his/her changes and undo the latest commit and return the project to the previous commit's state, they can choose to perform the following.

Git's commit history is a list of instances snapshots of the codebase and the `HEAD` points to the latest commit which is the last one in the branch. To now revert back to the previous commit, first the user would need to use the command '`git log`' which provides all the commits made to the server along with the timestamp, commit message and then the user can rollback to previous commits using the command '`git reset`' with appropriate flags.

The main ways to commit with these flags are '`--soft`' where the changes made are staged; '`--mixed`' ensures that the changes made by the user are kept in the working directory but are un-staged; '`--hard`' completely discards any new changes made. For instance, if the user doesn't want any of the new changes and wants to revert back to the previous commit, then they would first note down the ID of the latest commit (or the commit that they want to revert to) from the log of the git and then use the command '`git reset --hard <commit_ID>`'. This reverts by disregarding all the changes made and move the codebase back to the previous commit point.

Lastly, to undo the changes and effects of the last commit without rewriting history, the command '`git revert HEAD`' is used which creates a commit that undoes the changes made from the latest commit.