Tanner Wale

1/24/2025

IT-549 Foundation in Info Assurance

The first scenario of Alic sending a password and Bob comparing it to a database of passwords is not secure due to the protocol of Password Authentication Protocol or (PAP). This is because PAP sends passwords in plain text making them insecure and easily readable according to (Password Authentication Protocol (PAP) Security Explained | Okta, 2025) even using Point to Point Protocol or (PPP) this would be considered “weak authentication scheme.” This weak authentication protocol has no encryption method for user’s username or password. This is because the username and password are in plain text making them easy for a bad actor to view in a PPP session (Password Authentication Protocol (PAP) Security Explained | Okta, 2025).

The alternative for better authentication would be to use Challenge-Handshake Authentication Protocol or (CHAP). CHAP is more secure than PAP because it sends a three-way handshake request, the first request being an ask request for user to enter in a random generated string of numbers (Password Authentication Protocol (PAP) Security Explained | Okta, 2025). The next request, step two is according to (Password Authentication Protocol (PAP) Security Explained | Okta, 2025) a one-way hash password that is known to the server and user (Password Authentication Protocol (PAP) Security Explained | Okta, 2025). Lastly, the server will verify the hash string by decrypting it and compare it to the challenge sent in step two. Once passed the PPP will be established (Password Authentication Protocol (PAP) Security Explained | Okta,

2025) for the user. CHAP does not send a password making it more secure than PAP authentication method that sends username and the password in plain text.

The second scenario of Alice sending a password, and Bob hashing it and comparing it against a database of hashed passwords is insecure depending on protocol being used. As hashing being used in this scenario adds a layer of security by not storing these passwords in plain text creating a more challenging way for actors to view, steal, or change the passwords (Kinde, 2025). This is because hashing makes the password according to (Kinde, 2025) impossible to read, be stolen, or be reversed. Also, in the scenario it is not clear if salting is being used. If salting is not being used this would make the hashed password less secure because the process of salting hashed passwords is to “add random data to the input of a hash function to guarantee a unique output” according to (Arias, 2021). According to (Arias, 2021) the reason salting is used for hashed passwords is to make it more secure against “hash table attacks, slowing down dictionary and brute force attacks offline”.

For hashing the passwords SHA-1 should not be used as according to (Dashlane, 2023) this produces a 160-bit value that is no longer the recommended length as this is prone to collision attacks. Instead, SHA-2 should be used as this has a SHA-256-bit length and a SHA-512-bit length and is more secure than SHA-1 according to (Dashlane, 2023). For adding the protocol so that we can apply salting to the hashed passwords we need to use bcrypt. As according to (Arias, 2021) “data at rest and in motion should have at the very least be encrypted with TLS with 128-bit AES encryption.”

The third scenario Alice computes a hash and uses it as a secret key in a challenge response protocol. Here we are using CHAP again for challenge response. There is no password sent out for this as you are using the hash value to authenticate. This means according to (Understanding the Risks Associated with NTLM Authentication, 2024) if an attacker gains access to the hash value it can use it to impersonate the user without ever needing to know the password. With using CHAP again, we are at risk of making it easier on the actor if we use weak hashing algorithms. There is also a chance that a replay attack happens if attacker can intercept a valid challenge response and can store it for later use (Understanding the Risks Associated with NTLM Authentication, 2024). When the attacker is ready, they can send this vali challenge to the user tricking them by thinking they are communicating with the authenticator (Lokajit Tikayatray, 2023). This is possible if timestamps or unique identifiers are not present to stop actors.

To help safeguard CHAP replay attacks using timestamps will help the server know if a response is current (Lokajit Tikayatray, 2023). To help against attacker getting access to hash value and using it to gain access to authenticate with the system salting should be used. By adding random string to the output of the hash value the attacker would have known the original hash value. For salting bcrypt should be used. If we are hashing using the best hashing glorithms to make it harder on attacker to gain access to authenticate would be best. We can use SHA-2 with 256-bit or 512-bit length encryption with storing and in motion procedures for the data with TLS 128-bit at the very least with bcrypt for data in rest and in motion (Arias, 2021).

Last scenario is Alice computes the hash of a password and sends it to Bob, who hashes it and compares it against a database of double-hashed passwords. According to (Ilott, 2023) double hashing passwords is considered unnecessary as this does “significantly increase its security against brute-force attacks.” This can also slow down performance of computing power as this method of double hashing adds longer times to verify passwords (Sindastra, 2020). Also, with using double hashing for passwords according to (Ilott, 2023) this can make security best practices seem unimportant to install, such as salting. Double hashing does not according to

(Ilott, 2023) solve for salting. As you would still need to salt these passwords to not be at risk of brute-force attacks, dictionary attacks, and hash table attacks (Tshedimoso Makhene, 2024).

To make this authentication better we would not double hash. Instead, we would use Argon2 for salting to add another layer of security and make it harder on attackers to decrypt (Gupta, 2024). Then, using TLS encryption with 256-bit AES encryption for data at rest and in motion. Next, we would use Argon2 to make brute-force attacks even harder to execute successfully (Sindastra, 2020). Finally, we would use SHA-256 for hashing algorithm and according to (Gupta, 2024) can be used within Argon2 to help make a secure key for AES encryption.

References

Password Authentication Protocol (PAP) Security Explained | Okta. (2025). Www.okta.com. https://www.okta.com/identity-101/pap-security/

Arias, D. (2021, February 25). *Adding Salt to Hashing: A Better Way to Store Passwords*. Auth0. https://auth0.com/blog/adding-salt-to-hashing-a-better-way-to-store-passwords/

Kinde. (2025). *Password Hashing and Why It's Important*. Kinde Guides. https://kinde.com/guides/authentication/passwords/password-hashing-salting/

Dashlane. (2023, July 18). What Is Password Hashing? Dashlane. https://www.dashlane.com/blog/what-is-password-hashing

Understanding the Risks Associated with NTLM Authentication. (2024). Controlgap.com. https://www.controlgap.com/blog/understanding-the-risks-associated-with-ntlm-authentication

Lokajit Tikayatray. (2023, August 22). A Comprehensive Guide to Replay Attacks. Lokajittikayatray. https://www.lokajittikayatray.com/post/replay-attack

Ilott, M. (2023, July 27). Password Hashing and Storage Basics. Medium. https://markilott.medium.com/password-storage-basics-2aa9e1586f98

Sindastra. (2020, October 29). Why you probably shouldn't double-hash passwords and some thoughts. Sindastra's Info Dump. https://www.sindastra.de/p/1497/why-you-probably-shouldnt-double-hash-passwords-and-some-thoughts

Tshedimoso Makhene. (2024, February 10). What is password salting? Paubox.com; Paubox. https://www.paubox.com/blog/what-is-password-salting

Gupta, D. (2024, July 25). Password Hashing Showdown: Argon2 vs bcrypt vs scrypt vs PBKDF2. Meet the Tech Entrepreneur, Cybersecurity Author, and Researcher; Meet the Tech Entrepreneur, Cybersecurity Author, and Researcher. https://guptadeepak.com/comparative-analysis-of-password-hashing-algorithms-argon2-bcrypt-scrypt-and-pbkdf2/