

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE

100%


Hashing

TOTAL POINTS 4

1. How is hashing different from encryption?

1 / 1 point

- ☐ It's less secure.
- ☒ Hashing operations are one-directional.
- ☐ Hashing is meant for large amounts of data, while encryption is meant for small amounts of data.
- ☐ It's faster.


 **Correct**

Great job! Hash functions, by definition, are one-way, meaning that it's not possible to take a hash and recover the input that generated the hash. Encryption, on the other hand, is two-directional, since data can be both encrypted and decrypted.

2. What's a hash collision?

1 / 1 point

- ☐ When a hash digest is reversed to recover the original
- ☐ When two different hashing algorithms produce the same hash
- ☐ When two identical files generate different hash digests
- ☒ When two different files generate the same hash digest


 **Correct**

Awesome work! If two different files result in the same hash, this is referred to as a hash collision. Hash collisions aren't awesome, as this would allow an attacker to create a fake file that would pass hash verification.

3. How is a Message Integrity Check (MIC) different from a Message Authentication Code (MAC)?

1 / 1 point

- ☒ A MIC only hashes the message, while a MAC incorporates a secret key.
- ☐ They're the same thing.
- ☐ A MAC requires a password, while a MIC does not.
- ☐ A MIC is more reliable than a MAC.


 **Correct**

That's exactly right! A MIC can be thought of as just a checksum or hash digest of a message, while a MAC uses a shared secret to generate the checksum. This also makes it authenticated, since the other party must also have the same shared secret, preventing a third party from forging the checksum data.

4. How can you defend against brute-force password attacks? Check all that apply.


1 / 1 point

- ☒ Run passwords through the hashing function multiple times.

 **Correct**

Correct! A brute-force password attack involves guessing the password. So, having complex and long passwords will make this task much harder and will require more time and resources for the attacker to succeed. Incorporating salts into password hashes will protect against rainbow table attacks, and running passwords through the hashing algorithm lots of times also raises the bar for an attacker, requiring more resources for each password guess.


- ☒ Enforce the use of strong passwords.

 **Correct**

Correct! A brute-force password attack involves guessing the password. So, having complex and long passwords will make this task much harder and will require more time and resources for the attacker to succeed. Incorporating salts into password hashes will protect against rainbow table attacks, and running passwords through the hashing algorithm lots of times also raises the bar for an attacker, requiring more resources for each password guess.

- ☐ Store passwords in a rainbow table.

- ☒ Incorporate salts into password hashing.

 **Correct**

Correct! A brute-force password attack involves guessing the password. So, having complex and long passwords will make this task much harder and will require more time and resources for the attacker to succeed. Incorporating salts into password hashes will protect against rainbow table attacks, and running passwords through the hashing algorithm lots of times also raises the bar for an attacker, requiring more resources for each password guess.