Cybersecurity

Architecture and Design

2.8.1 Hashing and Digital Signatures

What are common applications for hashing and what can be added to hashes to make them more secure?

Overview
The student will summarize the basics of cryptographic concepts.

Grade Level(s)
10, 11, 12

Cyber Connections

- Threats & Vulnerabilities
- Networks & Internet
- Hardware & Software

This content is based upon work supported by the US Department of Homeland Security's Cybersecurity & Infrastructure Security Agency under the Cybersecurity Education Training and Assistance Program (CETAP).

CYBER.ORG

Teacher Notes:

CompTIA SY0-601 Security+ Objectives

Objective 2.8

- Summarize the basics of cryptographic concepts.
    - Digital signatures
    - Key length
    - Key stretching
    - Salting
    - Hashing
    - Key exchange

Hashing and Digital Signatures

Cryptography

The term ***cryptography*** is used frequently in the world of cybersecurity. The two root words used to form cryptography come from the Greek words ***kryptós***, meaning "a secret" or "to hide", and ***graphein***, meaning "to study." Put together, cryptography is defined as the practice and study of writing or solving codes.

Analyzing Crypto

The art or study of solving/cracking encryptions (without being told the key) is called ***cryptanalysis*** (analysis from the Greek word ***analýein*** meaning "to loosen" or "to untie"). Since no cipher is 100% secure, there are constant "attacks" to determine any flaws/weaknesses in the cipher. Cryptography (and all forms of security) is an endless game of "cat and mouse." Every time a new security tool is invented, attackers begin searching for weaknesses in the tool. As soon as a weakness is found, a new tool replaces the old and the cycle repeats.

More Terms!

One of the most overwhelming parts of learning anything new is the massive amount of new vocabulary terms introduced in rapid succession. We have already defined the terms cryptography and cryptanalysis while using the term ***cipher***. A cipher or key is the algorithm used to ***encrypt*** (convert) and ***decrypt*** (revert) a message. Starting with the ***plaintext***, the message to be sent which has not been encrypted yet, the cipher encrypts to ***ciphertext***, the

Teacher Notes:

message that has been encrypted that typically looks like a jumbled mess, and then the cipher decrypts back to plaintext.

Do Not Lose Your Keys

Since the key deciphers your message, you will want to keep the key private. If the public knew the key, anyone would be able to decipher the ciphertext, thus losing secrecy. There are many different types of cryptographic keys, so knowing what yours looks like won't necessarily help you determine a key for a different system. Some keys are called "one-way keys" or one-way hashes (to be discussed later) because there is no way to reverse the encryption.

Sometimes the algorithm used in a cryptosystem is well known. Just because others know ***how*** a message is encoded does not mean they can easily decode the message. Without the key, decrypting messages is not an easy task, so you should still feel secure with your messages (so long as the algorithm has been found to be secure. Some algorithms are inherently insecure due to weaknesses discovered by cryptanalysts.).

Defense

To increase the security of a key, the following advice is highly recommended. Do not use short keys! The shorter the key, the less time it would take for a brute force (trial and error) attack to break the key. Modern keys are typically 128-bit or more. As you learn more about keys, you will notice that more secure keys use very large numbers or the product of two large numbers (making an even ***larger*** number!). You can also implement the idea of ***key stretching*** to convert a password to a longer and more random one.

Another way to increase the security of a key is to always use a trusted domain when exchanging keys. Transmitting vital information through a channel where attackers may be watching may compromise the encryption. Diffie-Hellman is a common key exchange method that can be used over an untrusted channel. When possible, key exchanges should also be encrypted and protected to further increase the privacy of the key. (Of course, how do you establish an encrypted channel to exchange keys without first exchanging keys to encrypt the channel? We will talk about that in the section on Public Key Infrastructure.)

Teacher Notes:

Hash

A ***hash*** in the world of cryptography is a one-way encryption that converts any form of data into a unique string of text known as a ***digest***. Since hashing is a one-way algorithm, it cannot be reversed; therefore, the original text (often passwords) cannot be recovered. Any piece of data, regardless of size or type, can be hashed. There are many uses of hashing in cryptography including message integrity checks, digital signatures, authentication, information security applications, and cryptocurrencies.

Examples of Secure Hash Algorithm (SHA)

The following is a comparison of the passwords CYBER.ORG and cYBER. ORG among different Secure Hashing Algorithms.

Hashed Text/Algorithm	Hash/Checksum
CYBER.ORG/MD5	F330A7EB502673C4B04ED73AAC76D5FE
cYBER.ORG/MD5	896384B05DFF46415253AEDE49F94CAE
CYBER.ORG/SHA256	F0BAAB664607878E36B30CBADD482F99C4E 9A8BFAD30E6D82EC9301C1224B1C9
cYBER.ORG/SHA256	95DEEEACED21E9452BF34DB5E98F7A7BD5D 66A95A09D4F78AD9938A99B74803A
CYBER.ORG/SHA512	94E0EB75FEFB8AFAC47244C0A574EDD80FD 007AE635947A42F8D2B13129DD28DCB1C5B3 F79F687F6175E5F3C41AB73160BE7B57246543 DBB24E895E3CE0F68BE
cYBER.ORG/SHA512	A6210AFA70F3BA32FD9224A270915AC8E815E AA45CB639B27DCF147C1A1DB3B1DE2243D62 A4CC4638E3485C730EF2516B42154647569798 387DDC8A373FFD292

Obviously, do not try to memorize these. The point is to recognize how just one change in password (CYBER.ORG versus cYBER.ORG) provides a completely different, unique hash, and that as the number referenced in the SHA name increased so did the number of characters in the resulting hash.

Teacher Notes:

Collisions

Hashing should never have two different inputs with the same hash. If such a thing occurs, it's referred to as a ***collision***. It's extremely rare for a collision to occur in a good hash algorithm; however, in March 2005, Xiaoyun Wang and Hongbo Yu of Shandong University in China demonstrated a collision in the MD5 proving that collisions can exist in widely used hashing algorithms. In 2005, it was determined that SHA-1 can have collisions. Since then, SHA-1 has been considered "broken" and has been replaced by SHA-2 and SHA-3 algorithms with longer digests and thus fewer collisions, such as SHA256 and SHA512.

Salting

In an attempt to increase the strength of a hashed password, the cybersecurity industry now employs a technique called ***salting***. Salting is the addition of text to a password before hashing it and then storing the "salt value" with the hash. So while the user provides their password, the system adds the salt and then computes the hash digest. This results in a more secure hash in the event the user's plaintext password ever gets compromised. This also protects against a dictionary attack where an attacker might attempt to guess a password based on a list of words. The user's plaintext password may be on the dictionary list, but it's highly unlikely the password and salt will appear on the list together.

Example

In this example, the user's password is "panda". This could be easily guessed by a dictionary attack, but not if it gets "salted". Notice how two different salts (in bold) produce very different SHA256 digests. It is unlikely an attacker would have the word "panda" and the salt in their password list:

- Password: panda**e7y65i84jf83idj**
- SHA256 Hash:

fbe6f54415a28b23a5792796f297718de9a56b64e60ac3ad919e6da8d43e6de7

- Password: panda**j4l3ld092kldl377**
- SHA224 Hash:

752389e2b80ca552699fa58cf380e9f2fc60f040da28864ac13fc68a1e137490

CYBER.ORG

Teacher Notes:

Application

As a reiteration, three of the most common applications for hashing are to store passwords, verify a downloadable file, and provide ***digital signatures***. When storing passwords, the actual password isn't stored only the hash is. Hashes are then compared (instead of passwords) for verification purposes, and the original plaintext password is irretrievable. It is also never stored on the server directly where an attacker may seek to steal it.

When verifying a downloadable file, websites provide the hashed value of the original file. As long as the hash on the website and the computed hash by the user is the same, the file is assured to be unaltered.

Finally, digital signatures provide proof that a file or message has not been altered. As with file hashing, it checks to see that no changes have been made in transit.