Cybersecurity

Implementation

3.2.3 Database Security

How is a database secured?

Overview
Given a scenario, the student will implement host or application security solutions.

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 3.2

- Given a scenario, implement host or application security solutions.
    - Database
        - Tokenization
        - Salting
        - Hashing

Database Security

Usage of Database

Data is all around us in today's society. Keeping track of it all is a monumental task. This job is made easier through the use of databases. A ***database*** is a structured, organized storage and retrieval system for data. A database can be very small or unbelievably enormous, from just a few dozen rows of information to millions or even billions of rows of information. Most information found on the web is housed in some form of a database. All social media activity is stored in a database by the company that operates the social network. Google's search queries, the products listed for sale on Amazon or eBay, all the streaming videos on Netflix, YouTube or any other streaming service, and even the local car dealership's website are all housed in a database of one form or another.

Tokens

(As a reminder) ***Tokenization*** is the process of turning meaningful data into a random string of characters called a ***token*** that has no useful value if breached. Tokens are used to reference the original data, but a proper token cannot be used to guess the original data. This is different from encryption because tokenization does not use a mathematical process to transform the information into the token. Tokenization uses a database, referred to as a token vault, which stores the relationship between the original data and the token. Unfortunately, this is difficult to scale securely and maintain proper performance as the database increases in size.

My Hashbrowns are too Salty

A ***hash*** in the world of cryptography is a one-way encryption that converts any form of data into a unique string of text. Since hashing is a one-way

Teacher Notes:

algorithm, it cannot be reversed; therefore, original passwords cannot be recovered. Any piece of data, regardless of size or type, can be hashed. Traditional hashing produces data of the same length as the original, unhashed data. There are many uses of hashing in cryptography, including message integrity checks, digital signatures, authentication, information security applications, and cryptocurrencies.

In an attempt to increase the strength of a hash, the cybersecurity industry now employs a technique called ***salting***. Salting is the adding of random data to a password before hashing it and then storing the "salt value" with the hash. Notice the difference between:

- Password: CYBER.ORG**e7y65i84jf83idj**

  SHA1 Hash: E3737A1FC56F2F4388B9926875BD33CBD5058E49
- Password: CYBER.ORG**j4l3ld092kldl377**

  SHA1 Hash: A730E92EAA92E25BE5EEECDB5C22CBFDF4AD9708

The bold characters after CYBER.ORG are two different salts.

We use hashing to index and retrieve items in a database. Typically, it is faster to find the item using the shorter hashed key than to find it using the original value (assuming the original value is not "small").