CS50's Introduction to Cybersecurity

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan) (https://www.threads.net/@davidjmalan)
(https://twitter.com/davidjmalan)

Lecture 1

Securing Data

- This is CS50's Introduction to Cybersecurity.
- Last week, recall we focused on accounts.

Passwords

- We focused on our responsibility to keep our data secure.
- However, a third party is always involved in the storing of our data.
- You can imagine how a system may store usernames and passwords within a text file.
- You can also imagine how an adversary may get access to such a text file.
- Could we minimize the risk of storing passwords in plain text?

Hashing

- *Hashing* is a method by which we convert some plain text and output it as a hashed value that is less readable.
- Therefore, a *hash function* creates a hash value. A password is provided to a hash function and then is outputted as a hashed value.
- Without access to the precise hash function, an adversary cannot output the correct password.
- Generally, we want the hash function to output something very cryptic and lacking a pattern. Accordingly, adversaries cannot guess what the algorithm is doing.
- With the username and hash values stored in the server, an adversary cannot easily access the accounts on that server.
- When a user now inputs their password to log in, the password is passed to the hash algorithm again and compares the hash value created with the hash value stored.
- Hence, we have increased the cost, time, and resources required for an adversary to access protected data.
- Still, a *dictionary attack* could input one value after another from a dictionary into a hash function as a way by which to break it.
- Further, a *brute-force attack* could attempt to sequentially feed one character after another to attempt to break the password.
- Hypothetically, *rainbow tables* are another threat, whereby the adversary has a table of all the potential hashed values in a hash table. This, however, would take terabytes, if not petabytes, of storage capacity to accomplish.

- Finally, a problem arises when users utilize the same password and the hashed value of these passwords is exactly the same. How could we solve this problem?

Salting

- *Salting* is a process by which an added value is "sprinkled" into the hash function, such that a hash value changes.
- The utilization of a salt value nearly guarantees that the hash values provided by users, even those that have the same passwords, receive a different hashed password.
- Therefore, again, the cost for adversaries to crack these passwords is quite costly.
- NIST recommends that memorized secrets be both hashed and salted.

One-Way Hash Functions

- *One-way hash functions* are written in code and take in a string of arbitrary length and output a hash of a fixed length.
- Utilizing such a function, the holder of the hash value and hash function will never know the original password.
- Indeed, in some systems utilizing a one-way hash function, certain passwords may map to the same hash value.

Codes

- Cryptography is the study of transmitting secure data from one party to another.
- One way we can secure data is through codes.
- Codes convert the words we want to say into a less understandable string of words.
- *Encoding* involves taking plaintext and converting them into codetext.
- *Decoding* is the opposite, converting codetext into plaintext.

Ciphers

- Ciphering involves taking plaintext and *enciphering* them into ciphertext.
- This process of ciphering is called *encryption*. The process of deciphering them is called *decryption*.

Keys

- Keys are really big strings. These keys are used in encryption and decryption.
- *Secret-key cryptography* involves the passing of a key and plaintext into an encryption algorithm, where ciphertext is outputted.
- In this scenario, both the sender and receiver have a shared secret with one another in that they both have access to the encryption and decryption algorithm.

Cryptanalysis

- *Cryptanalysis* is the field of study and practice where individuals study how to encrypt and decrypt data.
- By evidence of your being part of this course, you, too, may be interested in cryptanalysis.

Public-Key Cryptography

- You can imagine a scenario where the sender and receiver of secure data may have never personally met. How can one establish a shared secret between two such parties?
- *Public-key encryption* or *asymmetric-key encryption* solves this problem.
- First, the sender uses a public key and plaintext and feeds these into an algorithm. This results in ciphertext.
- Second, the receiver uses their secret key, feeding in both this secret key and ciphertext into the algorithm. This results in deciphered text.
- *RSA* is a standard of encryption that describes this process.

Key Exchange

- An alternative algorithm is called *Diffie-Hellman*, the goal of which is key exchange.
- An agreed upon value `g` and a prime value `p` are used.
- Party A and Party B have a shared secret value called `s`.
- Both Party A and Party B have their own private keys.

Digital Signatures

- Using the building blocks of public keys and private keys, you can use these to sign documents.
- One can *sign* a document through a two-step process.
- First, a message, the content of a document, is passed to a hash function, resulting in a hash value.
- Second, a private key and a hash are passed to a digital signature algorithm, which results in a digital signature.
- The recipient is able to *verify* your digital signature by passing the message, the content of the document, to the hash function and receiving a hash. Then, the recipient passes the public key and the signature provided to the decryption algorithm, resulting in a hash value that should match the hash value previously calculated.

Passkeys

- *Passkeys* or *WebAuthn* are a more and more widely available technology.
- Soon, usernames and passwords will become less frequent.
- Passkeys will be device-dependent. For example, when visiting a website on your phone that prompts you to create an account, your phone will generate a public key and a private key.
- Then, you will send your public key to the website.
- From that point forward, to log into the website using that device, or a service that synchronises your passkeys across devices, you will pass a private key paired with a challenge value. An algorithm will produce a signature.

Encryption in Transit

- *Encryption in transit* relates to securing data as it moves back and forth through data networks.
- Imagine a scenario where two parties want to communicate with one another.
- We want to prevent a third party from intercepting data in between.
- Third-party services–like email providers–that function as intermediaries may indeed be reading your emails or viewing your messages.
- *End-to-end encryption* is a way by which users can guarantee that no third party in between can read the data.

Deletion

- Let's now consider a fairly mundane scenario, like deleting a file.
- Once files are deleted on a computer, a fingerprint of those deleted files may still be on your computer.
- Operating systems often delete files by simply *forgetting* where they exist. Hence, the computer may overwrite previous files with new files.
- However, there is no guarantee that the free space on your hard drive is entirely wiped off the fingerprints of old files.
- *Secure deletion* is a process by which all the remnants of deleted files are changed to zeroes, ones, or a random sequence of zeros and ones.

Full-Disk Encryption

- *Full-disk encryption* or *encryption at rest* entirely encrypts the content of your hard drive.
- If your device is stolen or you sell your device, no one will have access to your encrypted data.
- However, a downside is that if you lose your password or your face changes enough, you will not have access to your data.
- Another downside is that hackers may use this same type of technology through *ransomware* to encrypt your hard drive and hold it hostage.

Quantum Computing

- *Quantum computing* is an emerging computer technology that may be able to provide exponential computing power to adversaries.
- This technology may be used by adversaries to cut down on the time required to guess passwords and break encryption.
- Hopefully, we will have access to such computing power before bad actors do.

Summing Up

In this lesson, you learned about securing data. You learned…

- How websites and services store passwords;
- How text values can be hashed to ensure secrecy;
- About the roles of salting, one-way hash functions, keys, encryption, and decryption in securely storing data;
- About public and private keys;

- How technologies leverage public and private keys to keep data secure;
- How to secure your own hardware;
- Emerging benefits and threats posed by quantum computing.

See you next time!