CS356 - Scribe - September 12th

- Assignment 2 is up!!
  - OS Hardening
  - Hardening = strengthening
  - Can be hardware, network
  - Overview of assignment:
    - log in and go to node to root.
    - You will be admin and you can remove unneeded access for users.
    - shutting down services and then rebooting

- Weak versus strong collision resistance:
  - Alice = Good
  - Bob = Good
  - Charlie = Bad

- Weak: Alice, Bob and the Bank are involved in a communication line where Alice wants to send a message to the bank.
  - Alice sends a message “x” through the communication line using a hash function H, signed with her private key, PR.
  - Bob intercepts the message and decrypts it using the PU.
  - Bob notices that he can find a special message that when hashed will return the same value as hash function H.
  - Bob can now alter the message to suit the hash and forward the message to the Bank.
  - In this case, the message “x” was feasibly obtained by finding a collision in the hash map. The fact that the message is signed does not matter anymore.
  - Pigeon hole problem:
    - At least one hole must have 2 pigeons
  - In mathematics, the pigeonhole principle states that if n items are put into m containers, with n > m, then at least one container must contain more than one item. This theorem is exemplified in real life by truisms like "in any group of three gloves there must be at least two left gloves or two right gloves". ~google.com~

- Strong: Alice and the bank are involved in a communication line where Alice wants to send a message to the bank. Bob is not involved in this communication.
- Bob exhaustively searches for a pair of messages that when hashed return the same value.
- Bob notices this weakness and reports it to the bank.

User Authentication
- On the internet we need to make sure that the other person is who they say they are.

- Password authentication
  - Password Vulnerabilities
    - User error
    - Popular passwords
    - Workstation hijacking
    - Multiple password use
  - Countermeasures
    - Don’t allow access to /etc/passwd
    - Allow more bits of entropy (longer passwords, more symbols, etc)
    - Account lockout (annoying)
    - Automatic workstation logout
  - Implementations
    - Unix: uses modified DES, 56bit keys and a 12bit salt taken from system clock
    - Use blowfish
- Password Cracking
  - Users often use the same password for multiple interfaces
  - Use rainbow tables or dictionary attacks
- Password Choices
  - Users pick “easy” passwords:
    - Consist of <8 characters
    - Guessable
- Password File Access Control
  - Often stored in /etc/passwd (encrypted)
○ Make sure not to store plaintext passwords

● Making better passwords
  ○ Eliminate guessable passwords
  ○ Longer and easy to remember

● Proactive password checking
  ○ 8+ characters with more symbols

● Token Authentication
  ○ Keycard a user holds as secondary authentication
  ○ Eg Memory card, embossed card, smartcard

● Memory Card
  ○ Stores data but does not process data
Questions:

- When using a public wifi, is there any way to safely connect to sensitive material like banking?
  - A VPN would be a good start, but securing an insecure channel is a very hard problem in the first place.

- Why or why not is a Registration Authority such as Google safe to use?
  - The problem with centralizing authentication is that it creates a single point of failure. If your credentials at the registration authority are compromised, an attacker has access to everything registered with that authority.

- Why use /etc/passwd instead of /etc/shadow?
  - Older systems use /etc/passwd, but the two are largely interchangeable in terms of discussing weaknesses of *nix auth. On newer systems, /etc/shadow has less information- only the information required to verify a password, whereas /etc/passwd has more general user information like group memberships, home directory path, login shell, et cetera.

- What are security countermeasures that we see in movies, but are really not possible or the reality?
  - Those full-palm scanners are there all the time in the movies, but in reality they're less secure than fingerprints because the features they extract and compare are basic and general, like relative finger lengths or palm sizes. To more directly answer your question about things that seem like a good idea but aren't feasible, a number of movies will talk about absolutely unbeatable cryptosystems or about firewalls as though they're perfect and impenetrable. In truth, any networked system is likely to be vulnerable to some attack. Those attacks could be theoretically possible but practically impossible, based on currently available computing power, or they could simply be zero-day vulnerabilities.

- How safe are password management products such as Keepass?
  - A lot of this comes down to personal taste. I don’t like that password managers create a single point of failure, but avoiding their use encourages password reuse or simply using bad passwords. There are differences between one password manager and another, and those differences can make or break the safety of their use. For example, I wouldn’t trust a password manager that doesn’t encrypt the vault locally (on MY machine). I don’t want my passwords backed up in the
cloud, because I have no control over what’s done with them there. Does the manager encrypt them before backing them up? I hope so, but some don't. Investigate features before deciding on a product. Personally, I’d look for an open-source tool that creates a vault file encrypted with a master password that I can then back up at my own discretion.

- Is hashing the password the reason some places have a max password length?
  - No. For example, a hash can be 64 bits. The password can be 64, 124, or 256 bits, or whatever number of bits, and after it is hashed is will only be 64 bits.

- Why do majority of websites force users to restrict the length of passwords to 8 or 9 characters? why not more?
  - If a website has a small maximum password length, it’s because someone decided “8 or 9 is enough” without considering the security implications. It’s because of some other engineering limitation, like “the field in the database only allows 8 or 9 characters,” or “the web server allocates a buffer of that size and will drop any extra characters on the floor to avoid overflow.”

- I’ve heard biometrics being broken easily, such as a picture of someone’s eye easily fooling retinal scanners. Are some versions more secure or should biometrics be avoided for now?
  - As far as I know, Iris scanners are pretty good (but they’re relatively new, so they might be vulnerable to unknown attacks). Some fingerprint scanners are good, but a lot of them are cheap garbage you can fool with a piece of tape. In general, biometrics shouldn’t be used as the sole means of authentication. Biometrics (particularly static biometrics, like a fingerprint or iris or retina) make an excellent second factor, but probably shouldn’t be trusted to replace a password or hardware token.

- Is a microchip a feasible way of user authentication because it is harder to lose or misplace?
  - I assume you mean an implanted chip, like an RFID tag or something like how pets are “microchipped” for identification when animal control catches escaped dogs. To my knowledge, those microchips are highly effective at identifying animals and there’s no reason to assume they’d be any worse at identifying humans. There are privacy and consent concerns raised by the idea of implanting a microchip into someone, though, and it’s important to pay attention to them.

- Is there an easy counter measure to your /etc/passwd example?
- Access Control.

- Why not allow emojis as valid password characters?
  - There is a paper/study this year about this very concept.

- Does the keyring feature on various web browsers introduce any unique security threats?
  - Nothing unique, no, but using a browser keyring is no more secure than using a password manager- if an attacker can compromise your password store, then they've compromised every associated account.

- Do we know the details, from a systems security standpoint, of the Equifax data breach? What is the safest way to store personal passwords?
  - The safest way is to memorize them, but it can be a challenge to memorize different passwords to every service you use. The next best alternative is probably a password manager which encrypts them locally.

- What kind of encryption is used for passwords stored in web browsers?
  - Passwords to websites are stored, encrypted, by Firefox and Chrome. Where they store the master key depends on whether or not you’ve set a master password, or in Chrome’s case I think it’s just backed up to your Google account.

- What are the dangers with using facebook login for other services for an average users (grandma, etc)?
  - The average user won’t go out of their way to enable two-factor authentication at facebook. As a consequence, most facebook accounts aren’t very strongly protected, so an attacker can get access to grandma’s facebook (or even just capture the authorization cookie) and then log in and impersonate grandma at any service she’s authorized.

- How does a salt value differ from a hash value in terms of function?
  - A salt value (which should be a random number) is combined with a password before being put into the hash, in order to prevent /etc/shadow (for example) from containing the same hash every time two users happen to have the same password. A hash value permits verification of a password without storing the password itself.

- Am I understanding this right: the strong collision happens when Bob already has access to Alice's private key? Or, is it more like Bob can predict collisions of the hash function with an arbitrary key?
  - Strong collision occurs when Bob has two “messages” that hash to the same x value. Then he will send one to Alice to sign. Then he will send it to the bank
● How safe are password managers such as 1Password or Dashlane?
  ○ Without knowing the details of particular password managers (I don’t personally use one), if they store your credentials in a file on your computer such that the file is encrypted with a strong cipher and you can backup the file yourself, they’re reasonably safe. I don’t like password managers that force you to use “cloud storage,” because they aren’t transparent enough about what they’re doing with your most sensitive data.

● Is two factor (Two step) really that much more secure than simple passwords? should biometrics be a second factor in consumer electronics (password and thumbprint/face)?
  ○ A second factor increases an attacker’s search space substantially. An account secured with a two-factor auth system is much more secure than a simple password. Biometrics are good second factors, but they aren’t good replacements for passwords altogether.

● Do programs like LastPass (using randomized passwords stored in a database) improve security by allowing increased complexity of a single password? People often use passwords across sites anyway, under this assumption wouldn’t password vaults be a better alternative?
  ○ Yes, password managers which generate large random passwords use stronger passwords than humans can typically remember. They also generally use different passwords for every login. Their weakness is that they present a single point of failure: without a strong master password, an attacker can access all of your credentials easily. Managers that only store passwords in the cloud, or store passwords unencrypted, or both, are untrustworthy to the point of (in my opinion) near-uselessness. If you decide to use a password manager, use a good one.

● How did you get rid of your root account? Rename or delete or what?
  ○ There are a number of ways to disable a root account. One of them is to manually edit your /etc/shadow file, just deleting some of the characters in the hash of the password. Then, because the hash function always generates a bytestring of the same length, it will never match the stored hash.

CLASS QUESTIONS:
1) How does bob decrypt message from alice hash value i.e [h(x)]?
   Answer

2) Why don’t we encrypt hash any passwords on client side?
The verifiers on the host end have parts of the security they want to keep hidden. This can include, for example, the salting of the passwords. This allows for more security.