Lesson Plan: Password Cracking

Workshop Outcomes

- Appreciate the security considerations of password storage
- Identify the characteristics of a strong password
- Describe different techniques (brute force; dictionary and mask attacks) which can be used to crack passwords
- Apply password cracking techniques to solve simple challenges
- See that writing code is a superpower to overcome weak passwords.

Instructor tip: Try to be conversational. Encourage interaction and ask questions of your audience. These could be yes/no questions, or prompts for individual response.

Key messages

1. Passwords aren't often stored as plaintext. Rather, we calculate a *cryptographic hash* for a password and store/compare that instead. (The hash function is a key concept in Computer Science).
2. Lots of people choose weak passwords, and there are several techniques to exploit this and recover the password.
3. Computer technology can make our data more secure, but it also gives us the tools to attack that same security. Password cracking is a superpower you can attain by writing code to do the hard work for you.

Materials

- Participants have their own computer with "Password Cracking Workshop" Jupyter Notebook.
- Projector screen, ideally with speakers because the Myspace theme music is definitely a mood
- Google doc with short URL, or some other way to enable quick sharing of text among participants

Script

Introduction (1-2 mins)

Take a moment to introduce yourself as today's presenter. Use this as an opportunity to build rapport and find similarities between yourself and the audience to foster a connection.

- Who are you and where are you from?
- Tell us about your own interest in computer science – what sparked it? What continues to amaze you? What do you want to do with your qualifications?
- Cute or fun facts: interests, pets, etc

Icebreaker: group discussion on breakfast foods (3 mins)

Note to instructor: the key point in this section is that "hash" means "chopping and mixing". Introduce it here in a playful manner, but we will rely on the imagery later.

I love breakfast foods. Pancakes with maple syrup are pretty cool. Scrambled eggs are nice and hearty. Vegemite is patriotic.
Q: What did you eat for breakfast?
Q: What do we think about hash browns, and does anybody know what the word "hash" means?


Fun fact: "hash" means "chopping and mixing". It's derived from 'hatchet' in English, or 'hacher' in French. So hash browns are potatoes that have been *chopped and mixed.* A trendy cafe might serve you something like a corned beef hash, which is to say, corned beef and assorted vegetables that have all been *chopped and mixed*.

Hash browns have nothing to do with the # symbol (when referred to as 'hash' or 'hashtag'). They are different, unrelated meanings of the same word.

Note to instructor: participants may mention *hashish*, a traditional herbal preparation obtained through a process of *chopping and mixing*. *Hashish* is an Arabic-origin word with coincidentally similar meaning to *hash*.

Q: Are scrambled eggs a kind of hash? (I would argue no, because they aren't chopped, only mixed).

Overview and Value Promise (3 min)

Q: Who uses passwords to keep an account or device secure – we all do, right? Today I'm going to show you two things:

1. How passwords work, and

2. How you can easily beat them – especially if they're weak!

Security researchers (and hackers!) have learned that people make really bad and easily guessable passwords, and then reuse them across multiple accounts. Password technology isn't *necessarily* insecure, but as is often the case in security, humans often make poor choices that we can exploit.

We'll be doing a little bit of programming in Python, but don't worry if you haven't done any before. Most of the code is already written in the Jupyter notebook you have, and you'll be making small modifications. All you have to do is execute it, which you can do by clicking in the cell with the code and pressing Shift+Enter.

Activity: have a go at running the print statement at the start of your notebook.

Case Study Intro (10 mins)

We're going to start with the story of a real-world password security incident.

Myspace was an early social network site popular in the mid-2000s. It still exists today, but in a completely different form from its beginnings. The Classic Myspace aesthetic makes people in their 20s and 30s feel nostalgic for a simpler, more innocent time when crafting bomb CSS for their profile was their only care in the world.

Instructor to show sample Windows 93 Myspace profiles. Discourage self-directed browsing until after the session as some profiles may contain indecent content. Try to get emotional buy-in from participants from showing off awesome profiles, before we explain that the site was shut down prematurely due to poor password security. Feel free to find other cool profiles to show off.

One amateur Classic Myspace clone was hosted in conjunction with the Windows 93 browser-based gag operating system. It was home to profiles such as The Muffin Man (who bakes a mean breakfast spread), Vladimir Putin, F A X M A C H I N E, pizza girl, Lala the cat, Doge, and Tom, who was everybody's friend. Janken Popp *[theme music NSFW]* was the site's creator and operator.

In June 2021, Jankenpopp abruptly shut the site down and replaced it with the following message (the site has since been restored read-only, effectively freezing it in time):

> Hello everyone,
>
> It's with great regret that I announce the end of myspace93. No, the site has not been hacked and accounts are not compromised, but there are some old passwords floating around, and rather than keep the drama going I prefer to stop the service now.
>
> I have to explain what happened.
>
> Some time ago some users close to me (who had access to my server and who I thought were friends) had fun stealing source code and passwords from me and

> shared it on discord. Among these people was a malicious person, determined to destroy my site and its community.
>
> Being the big brother of 85k people is very demanding and tiring. It sometimes prevents you from doing constructive things in real life. I had planned to stop the site at 100,000 users for this reason, but these recent events have accelerated my decision. I would have preferred to stop the site on a happier note, unfortunately it will not be the case, too bad.
>
> Nevertheless, as announced a long time ago, I will give you the source code of the site (optimized and secured) in a few weeks, so that you can continue the adventure without me if you wish by running your own myspace clones.
>
> Thanks to all of you for these few years spent together <3 At a particularly difficult time in my life, it's been really wonderful for me to meet you all. You have been an amazing and incredibly creative community. I wish you all success in your future life endeavors. Just believe in yourself and nothing will stop you! I love you all very much.
>
> Thanks for the add ;D
> - Tom

Many users were gutted to lose access to this fleeting community. We don't know the details of the security incident, and some reports cast doubt on the truthfulness of Tom's story. However, we can speculate about how it might've gone wrong as we learn more about password security.

How passwords are stored: not in plaintext! (10 mins)

Q: Where do you think passwords are stored?
Whenever you set a password, it has to be stored somewhere. This could be on your device's disk, or it could be in a database run by your online service like Myspace, Gmail or Snapchat. But storing passwords exactly as you would type them (ie, in plaintext) is insecure, because anyone with physical access to the device can just open the disk and view your password.

By some accounts, Windows 93 Myspace passwords were stored in plaintext, and Jankenpopp just didn't care to fix the situation. At any rate, someone managed to get the password file (perhaps through a security hole in the Myspace website, or via shared access to the server as hinted at in Tom's message). While Myspace is low-value content, people who use the same email+password login combination on other web services were put at risk by the Myspace breach: those who acquired the leaked passwords could just try their luck with them on popular services like Gmail, which typically store far more sensitive personal information.

So storing plaintext passwords is a very very bad idea. Instead, we can use a *cryptographic hash function* to transform the password text. **Q: What do you think a hash function might do?** [A: it takes a string of text, chops and mixes it, and returns a unique, random-looking string.] There are many hash functions, and they take a lot of consideration to design well. A common one is Message Digest 5, or MD5. (The hash function is a key concept in computer science that is used in many other ways outside of security and cryptography).

Activity: think of some passwords and calculate their MD5 hash in the "Cryptographic hash functions" section of your notebook. Make sure to run line [1] ("from hashlib import md5").

Note: We've used the MD5 hash function in this workshop because it produces shorter hashes that are easier for humans to read. It's cryptographically broken (ie, there are known attacks against it) and not recommended for general use: instead, pick another hash function!

Good hash functions have the following properties:

1. They're deterministic: will always give you the same hash for the same input.
2. They're one-way: the hashes they produce should be extremely difficult to "unscramble" and recover the original text.
3. They minimise collisions: it is extremely difficult to find any two inputs that produce the same hash.
4. Similar input string (eg. changing just one letter) will produce wildly different hashes. This makes it extremely difficult to guess how "close" any two original strings might be by comparing their hashes.

Now we have a solution: With a good hash function, we can store the hashed password to disk. Whenever the user enters a password, we hash that too. If the two hashes match, the password is correct… but it's exceedingly hard to guess what password will produce the correct hash. **Q: How might this password scheme be less secure if the hash function used doesn't satisfy some of the above properties?** [A: If not deterministic, then you won't match correct passwords; if the hash values are easy to unscramble, the password can be recovered; if there are lots of hash collisions then more incorrect passwords will still produce the correct hash]

We're done!? No, we're not done. Even though the hash is irreversible, that doesn't mean it can't be attacked. Often, an adversary can see the hash, and can use a range of techniques to try to work out the password that created it. In the case of Windows 93 Myspace, this could have been an untrustworthy friend who had access to the Myspace server (and could therefore read the password file, even if it was hashed). We'll now look at some of the methods of recovering passwords from hashes.

Brute Force (10 mins)

Brute force methods take the least amount of skill and the most amount of time: try every possible solution until you find one that works. Sometimes the amount of time is acceptable, and if not, it provides a base case from which to improve. In this case, we'd cycle through all possible passwords and calculate the hash for each one until we get a match.

If we know that the password is a four-digit number – such as a combination bike lock, or an ATM pin – the password will be a number between 0000 and 9999: 10,000 possible combinations. A dedicated thief trying one combination each second could steal your bike within a few hours, but a computer could do it much quicker.

Fortunately, for our passwords we can use many more characters and make it much longer. Between letters of both cases, numbers, and keyboard special characters ($%&*;>?), there are 94 options for each character in a password.

Activity: With 94 options per character and varying password length (try it with 6 characters, then with more), calculate how many passwords there could be. Then, calculate how long it would take to try them all.

Debrief Q: How long would it take to try all the combinations for your more secure passwords?

Dictionary Attack (10 mins)

This type of attack exploits the fact that we know humans tend to be creatures of habit, and they like to use passwords that are easy to remember, for example, English words or their favourite person's name.

Wordlists are available all over the internet, although master hashcrackers build their own and guard them jealously. We're using one that contains 1 million of the top passwords used in English speaking countries. Here's a look at the first 100 of those passwords.

Activity: “Dictionary Attack” section of notebook
Q: How long does it take to hash one million known passwords?
Q: Are some of the passwords you use in the “million passwords” list?

Mask Attack (TIME OPTIONAL; 10 mins)

A mask attack is a step up from a brute force attack. The "mask" allows you to set conditions on the characters you'll test in each of the password character positions. This is useful, because we know about humans and how they choose passwords, and we can use this information to drastically reduce the number of passwords we need to try.

A very common pattern that people use to create their password is *NameYYYY* where YYYY represents their birth year. We can use a mask to tell the computer to check for upper case letters in the first position, then three lower case letters, then four numbers.

Activity: Read the “Mask Attack” section in your notebook and try to break the provided MD5 hashes using a mask attack.

Challenge (10 mins)

Activity: Use the techniques we’ve just learned to crack the passwords in the “Challenge” section of notebook

Extended Challenge (TIME OPTIONAL; 15 mins)

1. Think of a few passwords of varying degrees of security, and find their hashes.
2. Share your name and hashes on the Google Doc:

Your name	Password hash	Plaintext

3. Try to crack other people’s hashes, and paste the plaintext password next to it once you recover it.

Conclusion

We’ve learned how easy it can be to crack passwords. Real-world systems and attacks on them are a little more sophisticated than what we’ve covered, but the principles we’ve covered are still at the core of it.

Q: Based on what we’ve seen about guessing passwords, what makes for a strong password? What other security measures can we use?

- Think of a passphrase that is made up of at least four words, including at least 13 characters, for example 'horsecupstarshoe'. Make it meaningful to you so it is easy to remember.
- Install a password manager on your computer, smartphone or tablet. It will generate and remember secure passwords for you and some password managers will sync across your devices.
- Use two-factor authentication.
- Don't use the same password for multiple services or websites (Windows 93 Myspace serves as a cautionary tale)
- Use password tiers- unique and complex passwords for high risk accounts such as online shopping or social media, and less complex ones for low risk ones such as newsletters.
- Never give your password to anyone, ever!