COMP8760 Week 16 Class Worksheet

Budi Arief (b.arief@kent.ac.uk) – Based on materials by Shujun Li and Pierre Mondon
Class Supervisors: Imad Mahaini (M.I.Mahaini@kent.ac.uk) and Joshua Sylvester (jrs71@kent.ac.uk)

Thursday 21 November 2024

In this week's class, you will work on some password cracking exercises to gain a better understanding of *password security and usability taught in Week 16's lectures* (i.e. Lectures 15 and 16 – please revisit the Panopto recording of those lectures if necessary).

The first four exercises are easier and mandatory, and others are more advanced and optional. The former (Exercises 1-4) should be included in your CW2 assignment. *You can try the latter exercises (Exercises 5-7) in your own time as a challenge, but these will not be assessed.*

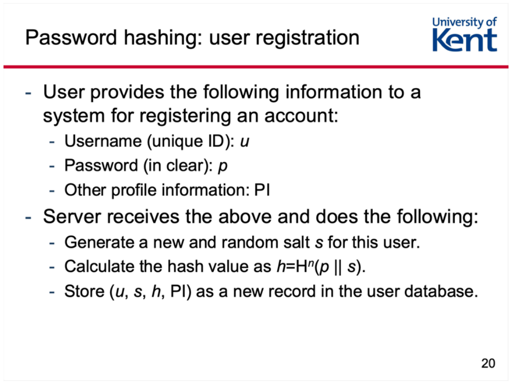
Different from the classes in the cryptography part of this module, for the class exercises from this week onwards, **you can use any third-party libraries and tools**, as long as you will still write some source code yourself to get the exercises done – and you give credits appropriately to any third-party resources you use. You are encouraged to use Google and other search engines to get help and look for hints. Work independently first before asking for help.

For Exercise 1, please show relevant information and the source code to a class supervisor to get your work checked. For Exercises 2-4 (as well as the optional Exercise 5), please show or send the recovered password with the source code to a class supervisor to get your work checked.

Note: This class is the first class that you will need to include in your CW2 practical report (i.e. the log book). *Please keep any evidence of the work you achieved (e.g. as screenshots or answers to specific exercises), and include them in the practical report that you will submit as part of CW2.*

Exercise 1: Password hashing and salting

Implement a function for the user registration process shown on Lecture 15 Week 16a Slide 20, and another one for the login process shown on Lecture 15 Week 16a Slide 21.

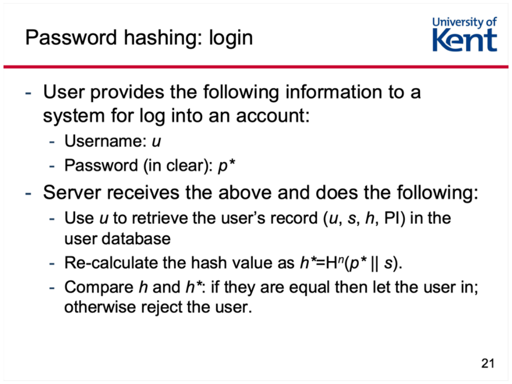


Password hashing: user registration

University of Kent

- User provides the following information to a system for registering an account:
  - Username (unique ID): *u*
  - Password (in clear): *p*
  - Other profile information: PI
- Server receives the above and does the following:
  - Generate a new and random salt *s* for this user.
  - Calculate the hash value as $h=H^n(p \| s)$.
  - Store (*u*, *s*, *h*, PI) as a new record in the user database.

20



Password hashing: login

University of Kent

- User provides the following information to a system for log into an account:
  - Username: *u*
  - Password (in clear): *p**
- Server receives the above and does the following:
  - Use *u* to retrieve the user's record (*u*, *s*, *h*, PI) in the user database
  - Re-calculate the hash value as $h^*=H^n(p^* \| s)$.
  - Compare *h* and *h**: if they are equal then let the user in; otherwise reject the user.

21

Write a program invoking the above two functions to show how you can use a password to register a username and then use the same password to log in successfully. For the login process, also use a wrong password to show how the system rejects a user who does not know the real password.

Exercise 2: General dictionary attacks

You found the following hash and salt values in a leaked password database on an underground forum:

```
3ddcd95d2bff8e97d3ad817f718ae207b98c7f2c84c5519f89cd15d7f8ee1c3b
```

This hash value is not accompanied by a salt, so either the original website did not use a salt at all, or the hacker removed the salt, rehashed the password using just one-round of a selected hash function, and then posted the database online (the latter is a common practice among hackers for releasing leaked password databases). There is no information about how many rounds of a hash function is applied, and you can try different values starting from 1. The hash function used is unknown, so you have to make a good guess (starting from the popular SHA-family).

As an experienced hacker, you know of a well-known leaked password database called phpbb [1] (https://raw.githubusercontent.com/danielmiessler/SecLists/master/Passwords/Leaked-Databases/phpbb.txt). Write a program to test if phpbb can be used as a simple dictionary to reveal the password. This is effectively about using passwords leaked from one website to crack passwords from another one. Why can you do this?

Exercise 3: Personalised dictionary attack

You found a hash value and the salt corresponding to the password of a user in another leaked database of an American website.

Hash value:	**`3281e6de7fa3c6fd6d6c8098347aeb06bd35b0f74b96f173c7b2d28135e14d45`**
Salt:	**`5UA@/Mw^%He]SBaU`**

Together with the hash value and the salt, you also saw a lot of personal information associated with this user in the database (which must be from a website for family members):

- Username: laplusbelle
- Name: Marie
- Surname: Curie
- Name of pet: Woof
- Birthday: 2 January, 1980
- Name of employer: UKC (University of Kent in Canterbury)
- Name of mother: Jean Neoskour
- Name of father: Jvaist Fairecourir
- First name of husband: Eltrofor
- Birthday of husband: 29 December, 1981

Because many users create passwords based on personal information (remember the YouTube video from the "Jimmy Kimmel Live!" show?), you wonder if this user did the same. Write a program to create a personalised dictionary and to try a personal dictionary attack to reveal the password.

Hint: Think about different ways to format dates, and how to handle white spaces.

Additional challenge: If the hash value is a different one as follows, can you still find it? This one requires taking care of case toggling.

`fc2298f491eac4cff95e7568806e088a901c904cda7dd3221f551e5b89b3c3aa`

Exercise 4: Breaking a graphical password (an Android unlock pattern)

NB: This exercise sets a realistic scenario where you will play the role of an attacker, in order to let you understand how an attack can happen. Do NOT try this in real world as it is illegal to access another person's phone without his or her explicit consent!

Your colleague Tom has a rooted mobile phone. One day, he switched his phone on by entering his unlock pattern, but got a phone call and went out of office to receive it. He somehow forgot to lock his phone. You

[1] This is actually the name of a web forum system and a website, from which the leaked password database was obtained originally.

sat next to Tom and quickly grabbed his phone and recorded the following hash value of his unlock pattern from the file "/data/system/gesture.key" on his phone's file system:

```
91077079768edba10ac0c93b7108bc639d778d67
```

You now want to recover the unlock pattern so that in future you can secretly check Tom's phone without him being aware of this (so you cannot just delete the file "/data/system/gesture.key" to erase the unlock pattern). Write a program to reveal the pattern.

Assume the 3x3 grid is labelled using 9 letters as follows:

```
a b c
d e f
g h i
```

Each unlock pattern is represented as a textual string and then hashed using one-round of the SHA-1 hash function. Each point (letter) can only be used once, so the maximum length of an unlock pattern is 9. Note that Android system actually represents the unlock pattern as a binary byte array and uses 0-8 to represent the 9 points, not a textual string like the above. We make the above changes to make this exercise different from some existing solutions what you may find on the Internet.

Exercise 5 (Optional): Personalised dictionary attack based on OSINT (open-source intelligence)

You found yet another hash value of a password alongside a (not so random!) salt in yet another leaked password database:

Hash value:	`3daf4e689df0c54d899ca2c0abb052c3af0fdadcd088b871ad51e207b8277d2b`
Salt:	`THISISASALT`

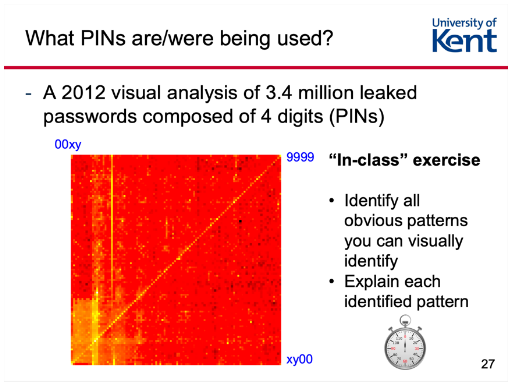
The leaked database also shows the name of the person the password belongs to: Athila Kita. Discover the password behind this hash value!

Hints:

1. Google "OSINT" and think about what it is about.
2. This password includes some special character(s)!
3. Ask the class supervisors for help if you are really stuck, but very keen to have a try.

Exercise 6 (Optional): Password visualisation

Pick up one or more leaked password databases containing a large number of 4-digit PINs. Try to reproduce the heat map shown on Lecture 15 Week 16a Slide 27. Can you observe any differences? Can you explain them?



What PINs are/were being used?

University of Kent

- A 2012 visual analysis of 3.4 million leaked passwords composed of 4 digits (PINs)

00xy
9999
xy00

"In-class" exercise

- Identify all obvious patterns you can visually identify
- Explain each identified pattern

27

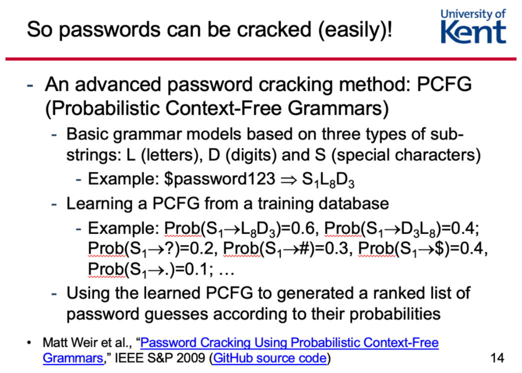
Extend the above to 4-digit segments of all passwords in the leaked password database. Redraw the heat map.

Can you find a way to visualise other types of passwords or segments of passwords?

Reference:
Rafael Veras, Julie Thorpe and Christopher Collins, "Visualizing semantics in passwords: the role of dates," VizSec 2012, https://doi.org/10.1145/2379690.2379702 or http://vialab.science.uoit.ca/pwdates/2012-vizsec-visualizing-semantics-in-passwords.pdf.

Exercise 7 (Optional): Training-based password cracking

Choose a leaked password database. Split it into two halves. Use one half to train a password cracker, using a probability model (which can be as simple as a word list of frequencies or as complicated as PCFG shown on Lecture 16 Week 16b Slide 14). Use the trained password cracker to guess passwords in the other half.



So passwords can be cracked (easily)!

- An advanced password cracking method: PCFG (Probabilistic Context-Free Grammars)
  - Basic grammar models based on three types of sub-strings: L (letters), D (digits) and S (special characters)
    - Example: \$password123 ⇒ $S_1L_8D_3$
  - Learning a PCFG from a training database
    - Example: Prob($S_1 \rightarrow L_8D_3$)=0.6, Prob($S_1 \rightarrow D_3L_8$)=0.4; Prob($S_1 \rightarrow$?)=0.2, Prob($S_1 \rightarrow$#)=0.3, Prob($S_1 \rightarrow$\$)=0.4, Prob($S_1 \rightarrow$.)=0.1; …
  - Using the learned PCFG to generated a ranked list of password guesses according to their probabilities

• Matt Weir et al., "Password Cracking Using Probabilistic Context-Free Grammars," IEEE S&P 2009 (GitHub source code)

14

Repeat the above for on two different leaked password databases (one as the training set and the other as the target database). Compare the results. What can you observe? Can you explain the observations?