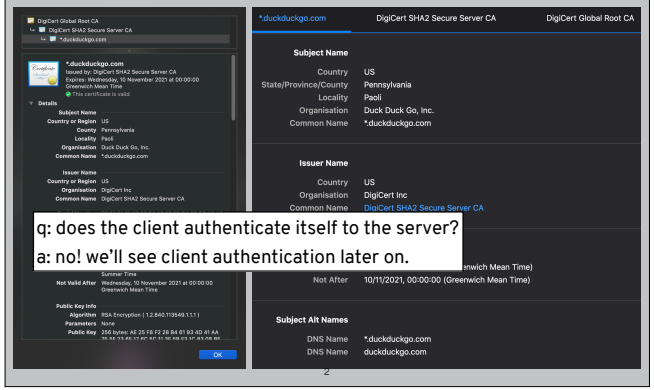
SECURITY (COMP0141): AUTHENTICATION

UCL

DIGITAL CERTIFICATES



q: does the client authenticate itself to the server?

a: no! we'll see client authentication later on.

Remember that question we asked back in Week 4: a certificate lets a server authenticate itself to the client, but how about the other way around?

AUTHENTICATION

Authentication is:

What you know

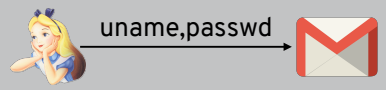
What you have

What you are

3

Authentication is split into three categories

PASSWORDS

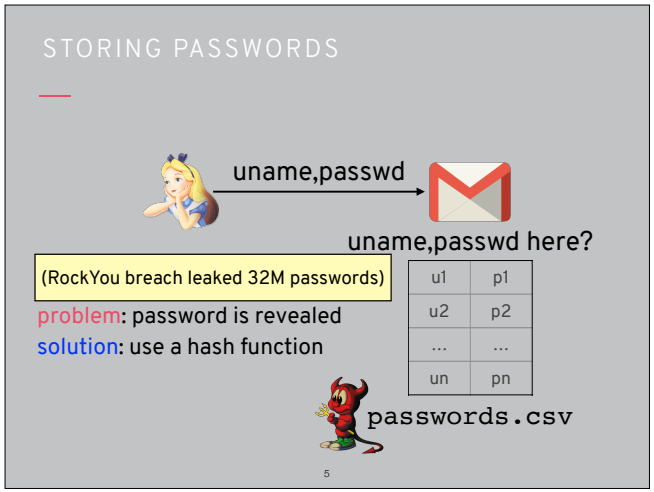


simple? how do you make them:

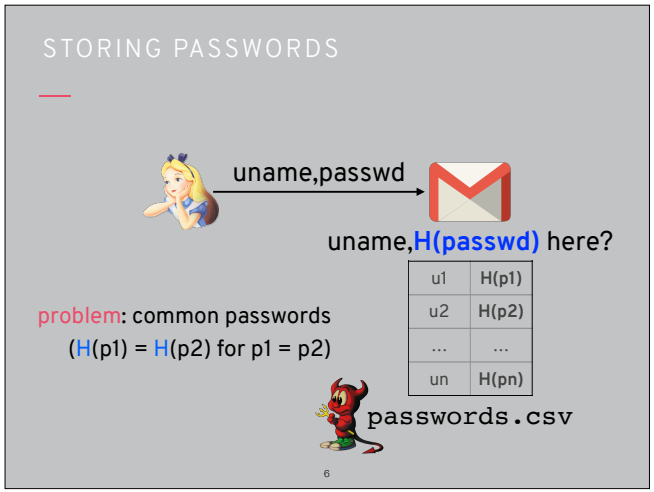
easy to memorise?	hard to guess?
easy to enter?	resettable?
hard to leak/steal?	retrievable?

4

The most obvious thing you know is passwords. They might seem simple and obvious because we're used to them by now, but actually they're not! We'll look first at the question of how to store passwords



How to store passwords? Consider a data breach scenario in which a password file stored on the server is taken by the attacker (so, our threat model assumes adversary has this ability). If passwords are stored in the clear then this is terrible since they learn everyone's passwords



Easiest solution is to store a hash of the password, then when user enters their password hash it and check if it matches. The problem though is hash functions are deterministic, so common passwords hash to the same value

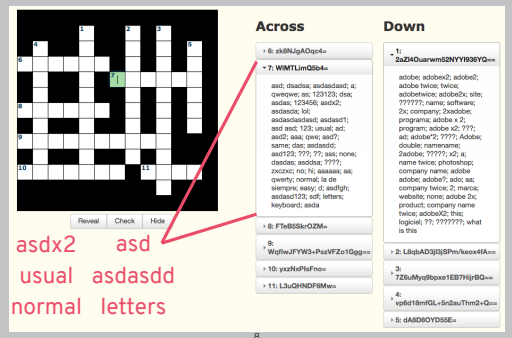
COMMON PASSWORDS

general purpose	service specific
password(1)	hotmail
123456	gmail
asdasd	dreamweaver
qwerty(123)	macromedia
<names>	linkedin

Most common password is ‘password’ and all these other ones

COMMON PASSWORDS

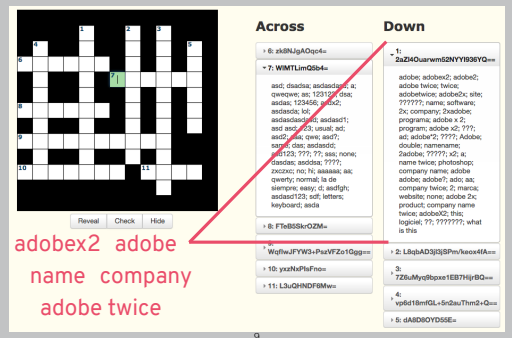
Adobe: 38 million password records stolen



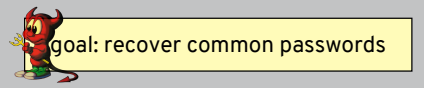
You can find this crossword at https://zed0.co.uk/crossword/

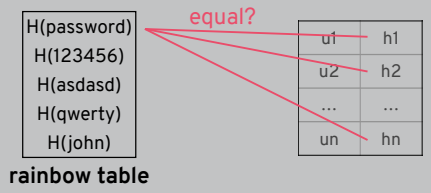
COMMON PASSWORDS

Adobe: 38 million password records stolen



PASSWORD CRACKING

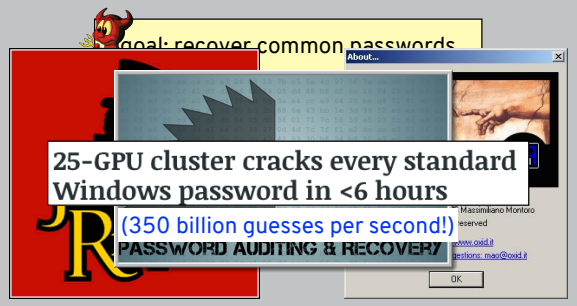




this is a **dictionary attack**

To get common passwords, attacker performs what's known as a dictionary attack: form rainbow table consisting of hashes of most common passwords, then go through hashes to see if they match (again, this exploits determinism of the hash function)

PASSWORD CRACKING

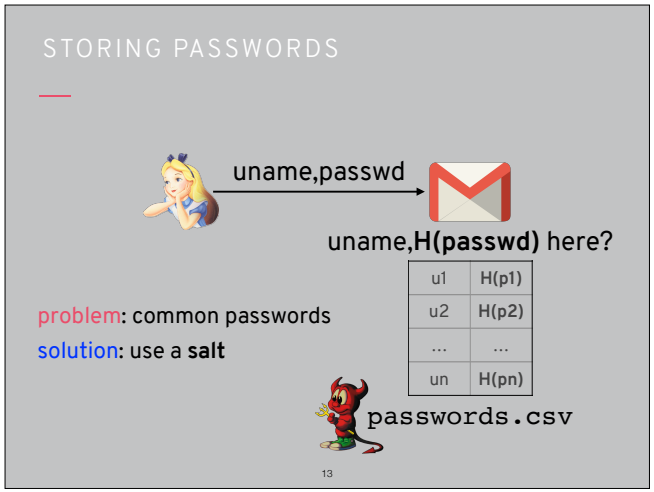


this is a **dictionary attack**

Do we think this is hard, even for non-technical attackers? Of course not! Lots of programs exist to do this for you, and they can do it very quickly

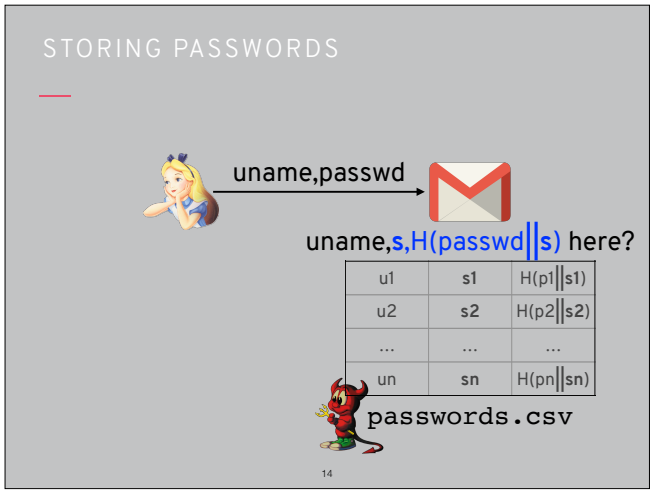
JOHN THE RIPPER DEMO

```
smeiklej@nocciola demos % john --wordlist=rockyou.txt --stdout | head
Using default input encoding: UTF-8
Press 'q' or Ctrl-C to abort, almost any other key for status
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
smeiklej@nocciola demos %
```



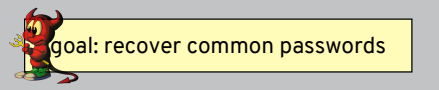
The solution to this problem is called salting, and this is currently the best practice

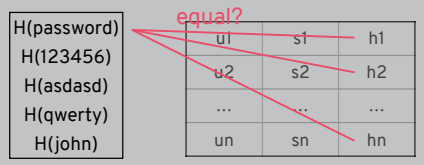
---



Add a little bit of randomness into the hash function, as we've seen before this results in a very different hash because of uniformity

PASSWORD CRACKING





dictionary attack won't work!

(H(password||s1) ≠ H(password||s2) if s1 ≠ s2)

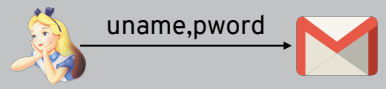
This means attacker can't use one single rainbow table to get everyone's passwords, since users will have different salts (and thus different hashes) even if they have the same password

JOHN THE RIPPER DEMO

```
smeiklej@nocciola demos % john --wordlist=rockyou.txt --stdout | head
Using default input encoding: UTF-8
Press 'q' or Ctrl-C to abort, almost any other key for status
123456
12345
123456789
password
iloveyou
princess
1234567
rockyou
12345678
abc123
smeiklej@nocciola demos %
```


It's still important to not use common passwords though

PASSWORDS



simple? how do you make them:

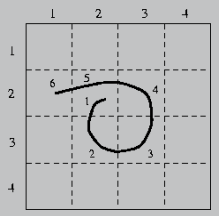
easy to memorise?	hard to guess?
easy to enter?	resettable?
hard to leak/steal?	retrievable?

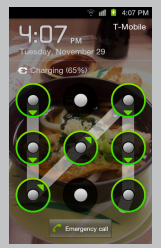
how to retrieve hashed and salted password?

See a tension here, because hashing and salting passwords makes them impossible to retrieve

---

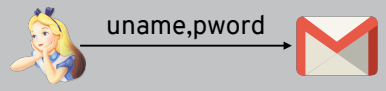
GRAPHICAL PASSWORDS





In terms of entering passwords, also see some graphical solutions

PASSWORDS



simple? how do you make them:

easy to memorise?	hard to guess?
easy to enter?	resettable?
hard to leak/steal?	retrievable?

19

Can also be hard to memorise passwords

SECURITY QUESTIONS

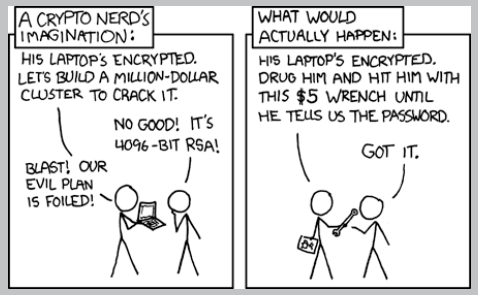
What is your pet's name?
In what year was your father born?
In what county where you born?
What is the color of your eyes?

What is the first name of the person you first kissed?
What is the last name of the teacher who gave you your first failing grade?
What is the name of the place your wedding reception was held?
In what city or town did you meet your spouse/partner?
What was the make and model of your first car?

20

Security questions don't have to be memorised, on the other hand they can be researched easily, so it's not just you who knows the answer. They can also embarrass or otherwise upset users

COERCION ("RUBBER-HOSE") ATTACK



21

Being easy to memorise can actually be a bad thing since it means you'll reveal your password if bribed / coerced

COPING STRATEGIES

try to enforce:	**users will:**
different passwords	use same password
long passwords	add '123' (or other padding)
random passwords	write passwords down
letters and numbers	add '1' (or '123' or other)
regular passwords changes	add, e.g., 'spring' or 'june'
more complicated stuff	use reset functionality

22

Usability / psychological acceptability is a big issue here, people have all sorts of tricks for dealing with complicated policies

This can sometimes lead to embarrassing situations. Do you see it?


MilFlip Logon Details
ATTENTION ALL
CRMS STUDENTS


MilFlip Logon Details

AUTHENTICATION AS A USER TASK

Properties of password-based authentication:

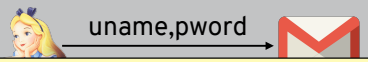
- Unaided recall (violation of "recognition rather than recall")
- Recall and entry have to be 100% correct
- No corrective feedback on failure

Coping strategies include:

- Re-using the same password (or small set of passwords)
- Writing passwords down
- Relying on password reset

A lot of these violate the principles of usable design that we looked at last week

PASSWORDS



security: want long and random passwords
usability: humans can't generate or remember these

easy to memorise?	hard to guess?
easy to enter?	resettable?
hard to leak/steal?	retrievable?

Passwords display classic tension between security and usability

AUTHENTICATION

Authentication is:

What you know

text passwords | graphical passwords | personal details

What you have

What you are