COMP7904 Information security: attack and defense

Lecturer: SM Yiu

Teaching Assistants: Ken {Ma, Wong}

Course web page

http://moodle.hku.hk

COMP7904 [Section 1A, 2022]

+

27- 30 “lecture + lab” hours (3 hours per week)
- face to face (video recording as a backup)?

2-3 laboratory exercises

One test (may skip)

Consultation:
by appointment

Emails

Discussion forum

This course contains a bit of hands-on exercises, don’t wait until the last minute to ask questions.

Coursework: 40%; Final Exam: 60%

What this course is about?

Objective: Teach students how to conduct ethical hacking so as to better protect a computer system.

Ethical hackers: hack a system as a malicious hacker, but with the permission of an authorized person to find out weaknesses of the system and improve it.

Topics: Password cracking, physical security, network hacking, operation system hacking, application hacking, and maybe some R&D problems related to hacking and defense.

Emphasis: Balance between theory and practice (i.e., quite a bit of hands-on exercises)

Remarks:

- It is NOT an easy course. Do NOT enroll the course because you are attracted by the fantasy word "hacking".

- We expect students to have knowledge in
  * Network
  * Operating systems
  * Programming
  * Basics of security & cryptography

- And students need to read extra material not covered by the lectures and spend (a lot of) time doing the hands-on exercises.

Outcome-based learning (OBL)

OBL – a process to help improving teaching/learning with the focus on students (how much you have learned)

* Define a set of outcomes
* Evaluate if students can achieve the outcomes via assignments, tests, and questionnaires

How you help?

- provide comments on the expected outcomes
- feedback on whether we can help you to achieve outcomes

Goal: to provide a better course for you.

Course outcomes: [What we expect you to achieve]

On successful completion of the course, students should be able to:

CO1: master the key techniques and theories behind various hacking activities and provide solutions on how to protect a computer system against these attacks.

CO2: analyze and propose similar attack and defence methodologies.

CO3: Able to acquire and self-learn the latest hacking and defence technologies and try to develop new ideas.

Disclaimer

- Any actions and/or activities related to the material contained in this course is solely **your responsibility**.
- The misuse of the information in this course can result in **criminal charges** brought against the persons in question.
- **The instructor and the TA(s)** will not hold responsible in the event any criminal charges be brought against any individuals misusing the information in this course to break the law. [Always make sure that what you do is legal, consult us if you are in doubt before you do it!]

Today, since this is the first lecture,
let us start with an easy topic as a warm-up:
password cracking

Password cracking (easy or difficult?)

Password: A string of characters used to (i) authenticate a user by comparing the provided password to a value that has previously been stored and is associated with a specific user ID.

Username	Password
smyiu	24680
twchim	happy
kkma	system
.....	.....

Q: Can we store passwords in plain text in server?

Cryptographic hash function

Of course not! We should store the "hash values" of the passwords instead.

Have you learned cryptographic hash?

Roughly speaking, a hash function takes an arbitrary-length message m and returns a fixed-length hash value (message digest) h(m).

$$h: \{0, 1\}^* \rightarrow \{0, 1\}^r$$

A fixed length (=r) bit string.

* Means an arbitrarily long bit string (e.g. a document)

e.g. For MD5, r = 128; for SHA-1, r = 160.

Note:
h is known to everyone and easy to compute so that everyone can check the hash value;

Q: Will two documents have the same digest? Yes!

Some requirements of (cryptographic) hash function

1. One-way
2. Weak Collision Resistance
3. Collision Resistance (Strong Collision Resistance)

Reminder: h is known to everyone.

[P1] Given y where y = h(x), to find x.
Req. 1: It is difficult (computationally infeasible) to solve P1.

Req. 1: If you know the message digest of an message x, it is hard to compute the message x.

If a function satisfies this requirement, we call it an "one-way" function.

Some requirements of hash function

1. One-way
2. Weak Collision Resistance
3. Collision Resistance (Strong Collision Resistance)

[P2] Given x, find x' such that $x \neq x'$ but $h(x) = h(x')$.
Req. 2: It is difficult (computationally infeasible) to solve P2.
[x, x' is called a collision if $h(x) = h(x')$!]

Req. 2: You know the message x and of course, you know its hash value (message digest), it is difficult to find another message x' so that they have the same hash value. That is, it is hard to replace the message x by some other message x' so that the receiver does not notice it.

If a function satisfies this requirement, we say that it is "weak collision resistant".

Some requirements of hash function

1. One-way
2. Weak Collision Resistance
3. Collision Resistance (Strong Collision Resistance)

[P3] Find x and x′ such that $x \neq x'$ but $h(x) = h(x')$.

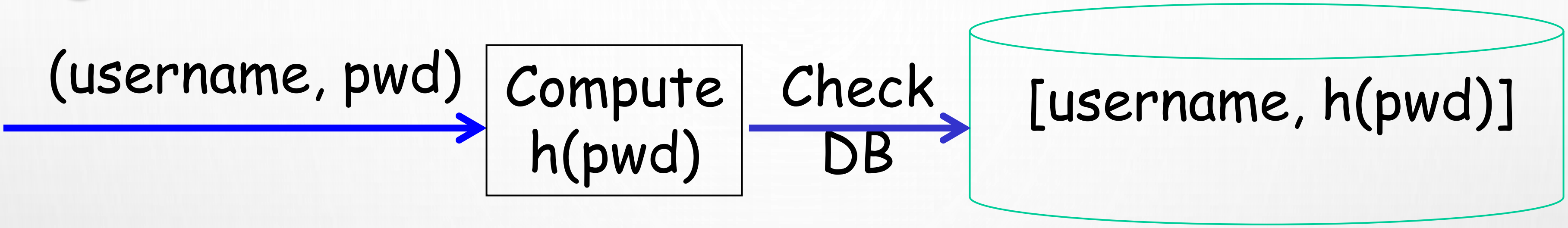
Req. 3: It is difficult (computationally infeasible) to solve P3.

Note: Such x, x′ pairs must exist since # of possible documents >> # of possible hashed values.

Req. 3 implies that it is not easy to find two different messages x and x′ such that they have the same message digest although we know that these (x, x′) pairs must exist!

If a function satisfies this requirement, we say that it is "strong collision resistant".

No plaintext of passwords will be stored in the server.



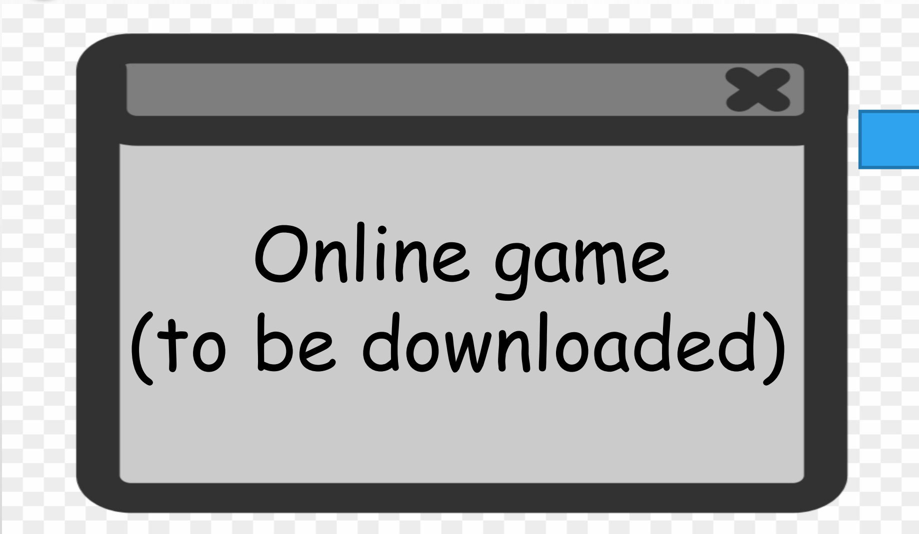
Q: Which requirements of hash functions are required in this application (R1, R2, R3)?

R1 ✓

R2 ✗

R3 ✗

Another application: To make sure that the file(s) has (have) NOT be tampered.



Downloaded version

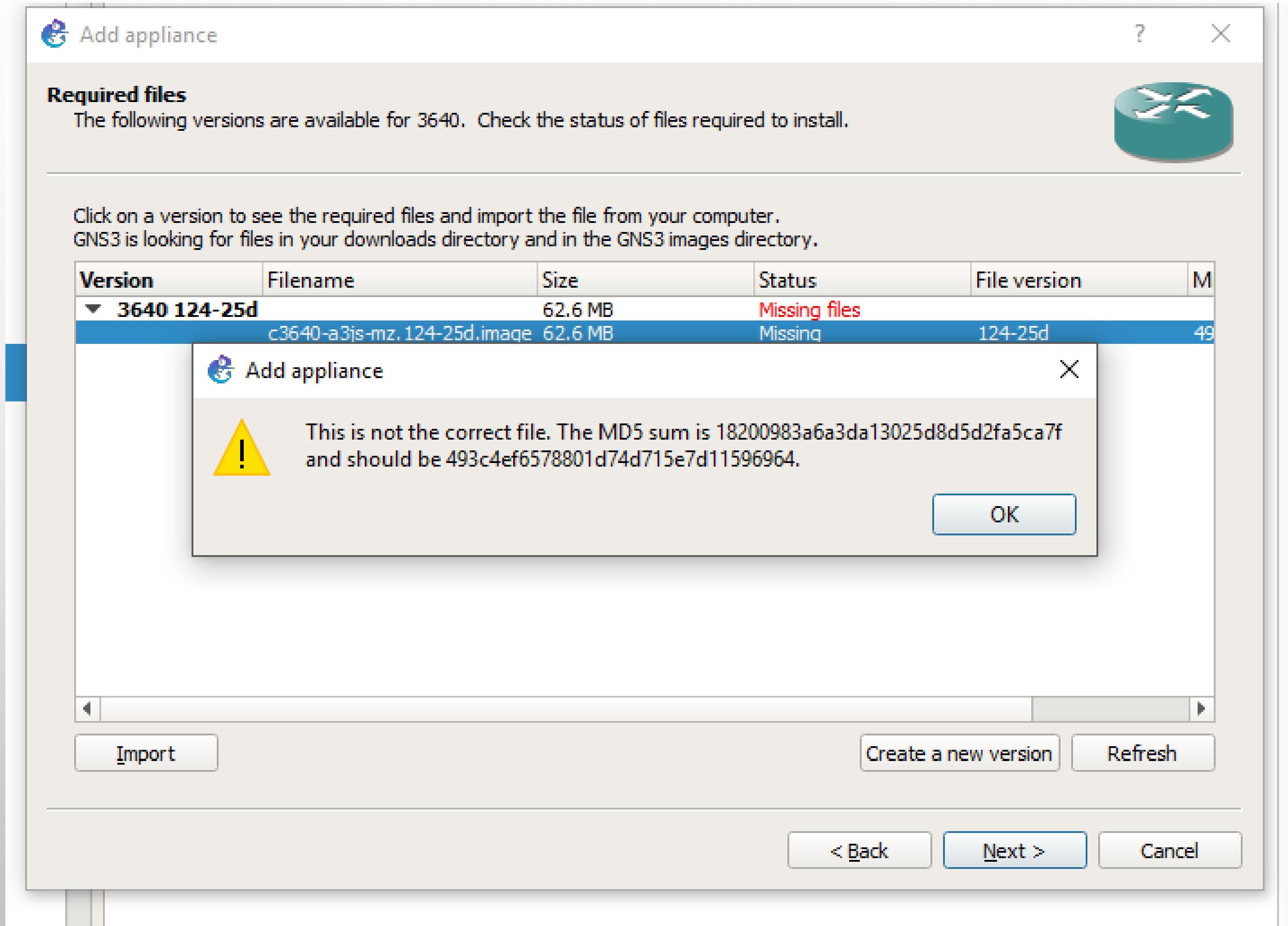
Calculated hash value

Matched?

Hash value posted somewhere

Q: Which requirements of hash functions are required in this application (R1, R2, R3)?

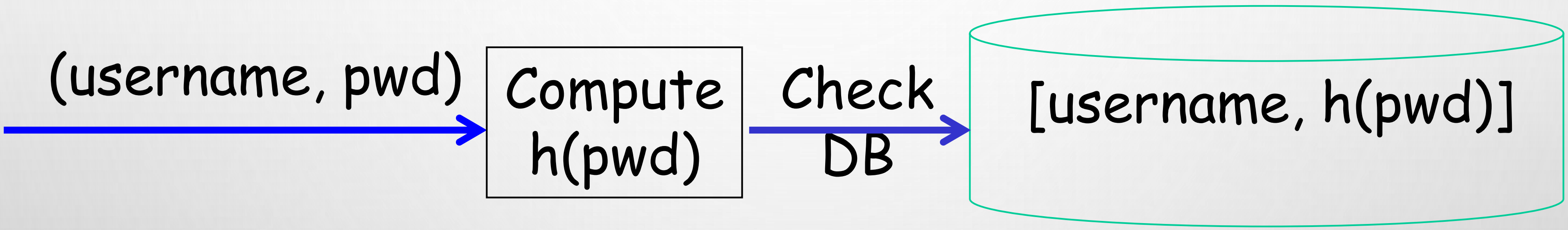
Q: Can you think of more applications that require R1, R2 and/or R3?



Add appliance
Required files
The following versions are available for 3640. Check the status of files required to install.
Click on a version to see the required files and import the file from your computer.
GNS3 is looking for files in your downloads directory and in the GNS3 images directory.
Version
Filename
Size
Status
File version
M
3640 124-25d
62.6 MB
Missing files
c3640-a3js-mz.124-25d.image
62.6 MB
Missing
124-25d
49
Add appliance
This is not the correct file. The MD5 sum is 18200983a6a3da13025d8d5d2fa5ca7f and should be 493c4ef6578801d74d715e7d11596964.
OK
Import
Create a new version
Refresh
< Back
Next >
Cancel

Password cracking (easy or difficult?)

Password: A string of characters used to (i) <u>authenticate</u> a user by comparing the provided password to a value that has previously been stored and is associated with a specific user ID.



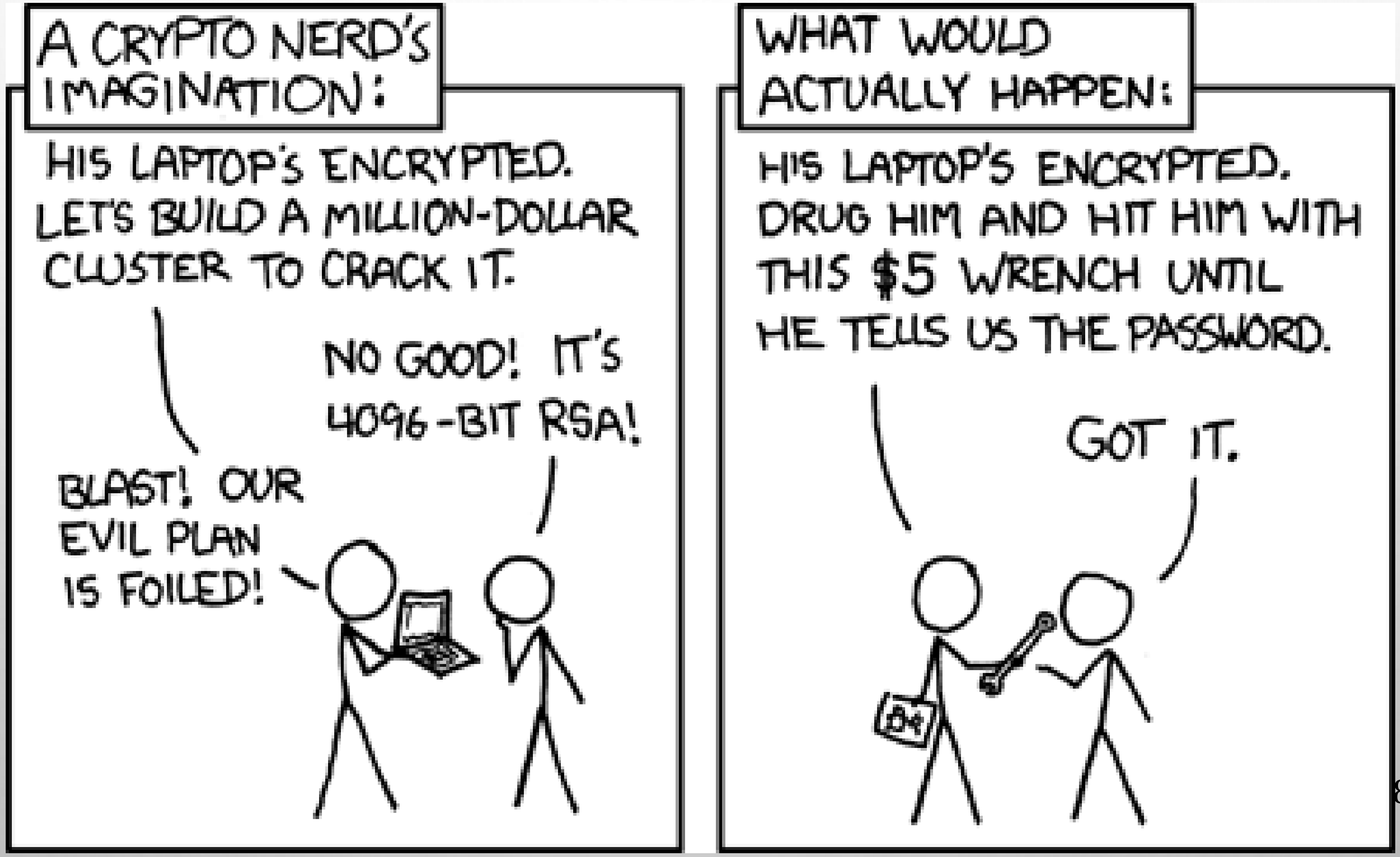
(ii) Or to protect a file from access by unauthorized users e.g. Microsoft office files; pdf files

Q: Do you know how to create a password protected pdf file or a Microsoft office file?

Q: Do you know which encryptions are used in these tools? Are they secure?

How to acquire passwords

(a) A very simple approach: try to ask for it



(b) Another simple approach: try to look for it

<u>Common places</u> to look for (examples):

- Monitor (front, sides, top…)
- Under the keyboard
- In drawers (e.g. underside of it, under pen cases…)
- Inside mobile phone (e.g. In memo)
- Text files in hard disk (e.g. password.txt)

(c) Another simple approach: guess it

<u>Examples:</u>

- Birthday of user, spouse, kids….
- Telephone numbers
- Names of spouse, boy/girl friends, kids, pets, …
- Commonly used passwords

A paper (2016) on top-ten least-secure passwords

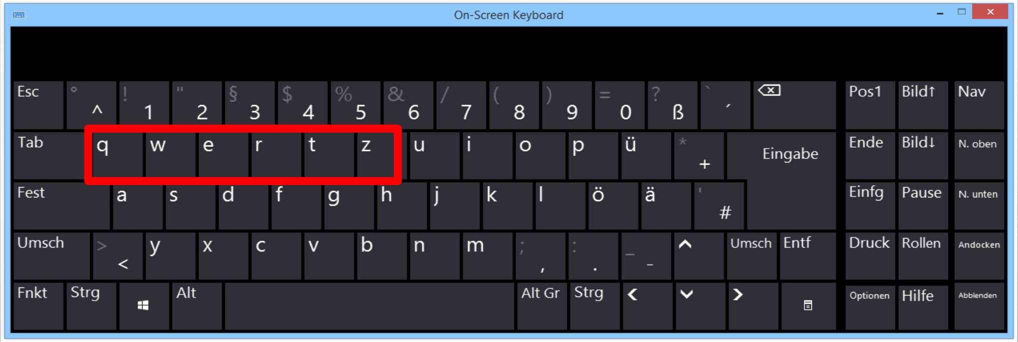
- Lancaster U, Fujian Normal U, Peking U
- Based on a leaked Yahoo database of personal info.

123456
password
welcome
ninja
abc123
123456789
12345678
sunshine
princess
qwerty

Other results:

❖ Able to guess passwords for more than 73% user accounts

❖ 1/3 of these passwords can be guessed right within 100 guesses.



Link: http://www.securitymagazine.com/articles/87581-the-10-easiest-passwords-to-hack

SplashData revealed the list for 2017 by analyzing more than five million user records leaked in 2017.

2016

123456
password
welcome
ninja
abc123
123456789
12345678
sunshine
princess
qwerty

2017

123456
password
12345678
qwerty
12345
123456789
letmein
1234567
football
iloveyou
………
starwar
qazwsx

How about 2019?

123456
123456789
qwerty
password
1234567
12345678
12345
iloveyou
111111
123123

[Also, by SplashData]

How about 2020?

123456
123456789
picture1
password
12345678
111111
123123
12345
1234567890
senha

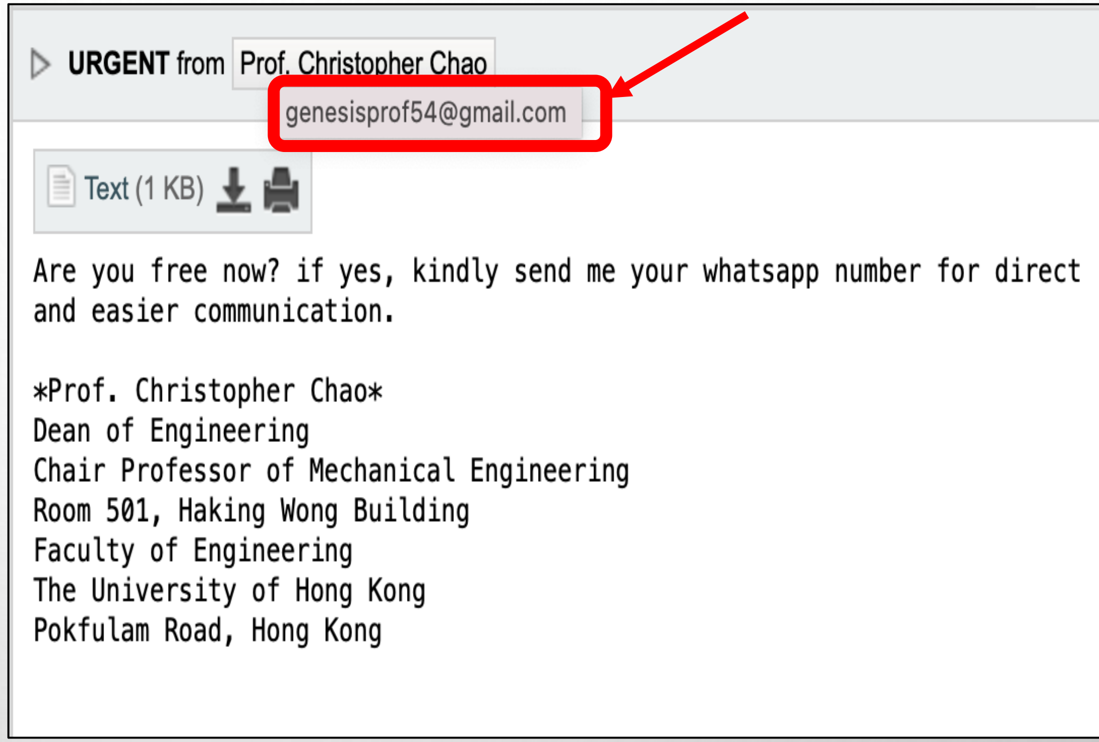
[By NordPass]

(d) Another approach:

trick the user to give you the password or sensitive information

Not a correct email

Phishing attack



URGENT from Prof. Christopher Chao

genesisprof54@gmail.com

Text (1 KB)

Are you free now? if yes, kindly send me your whatsapp number for direct and easier communication.

Prof. Christopher Chao
Dean of Engineering
Chair Professor of Mechanical Engineering
Room 501, Haking Wong Building
Faculty of Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong

Better attack may use a near-identical email address:

abc@companyname.com

abc@company-name.com

Social engineering attacks

Social engineering is the term used for a broad range of malicious activities accomplished through human interactions. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information.

Real case:

Lithuanian Man Pleads Guilty To Wire Fraud For Theft Of Over $100 Million In Fraudulent Business Email Compromise Scheme

Geoffrey S. Berman, the United States Attorney for the Southern District of New York, announced that EVALDAS RIMASAUSKAS, a Lithuanian citizen, pled guilty today to wire fraud arising out of his orchestration of a fraudulent business email compromise scheme that induced two U.S.-based Internet companies (the "Victim Companies") to wire a total of over $100 million to bank accounts he controlled. RIMASAUSKAS entered his guilty plea today in Manhattan federal court before U.S. District Judge George B. Daniels.

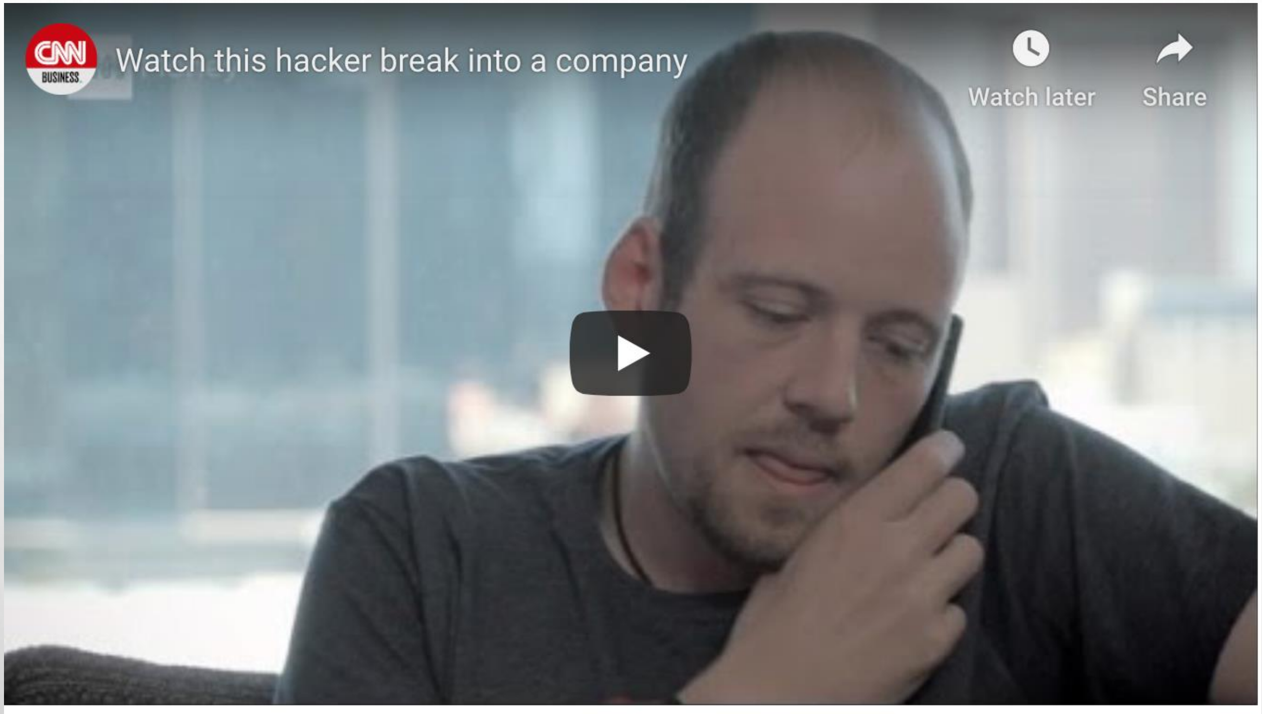
1. Rimasauskas and his team set up a fake company with bank accounts.
2. They pretended to be a computer manufacturer worked with Google and Facebook.
3. Sent invoices to specific Google and Facebook employees, directing them to deposit money into their controlled accounts.

Between 2013-2015, cheated over $100M

Phishing – one type of social engineering attack
Q: can you dig out the other types?

Can occur by "phone", "SMS", "Email", "Face-Face" and a combination

Look at a demonstration how easy a social engineer can hack people:



https://youtu.be/PWVN3Rq4gzw

Q: E.g. Phishing vs spear phishing: what are the differences?

Q: How to prevent it? [Also, tools available :-P?]

(e) Another approach:
try to crack the password

Cracking a password involves trying every possible password combination, or every combination in a defined subset, until you find the right one.

There are four basic approaches, or "attack types," that password-crack utilities employ.

- Brute force attacks
- Dictionary attacks
- Hybrid attacks
- Rainbow table attacks

Brute-force attack

Guess the password by systematically trying every possible combination of characters until the correct password is discovered.

E.g. If a password must be 10 characters, with each character being 0 to 9, how many passwords an attacker need to try in the worst case to crack the password.

Q: How about on average?

More exercises:

Length	domain	# of pwds
1	Lower letters	??
	Lower/upper letters	??
	Lower/upper letters, digits	??
4	Lower letters	??
	Lower/upper letters	??
	Lower/upper letters, digits	??

# of passwords depend on

(i) Length of the password (longer, the better)

(ii) Domain of the characters (larger, the better)

Brute-force attack usually fails if the total number of passwords is large enough.

(i) Assuming a desktop computer could try one million passwords per second.

	lower case	lower/upper	lower/upper/digits	lower/upper/digits/symbols
1	26 microseconds	52 microseconds	62 microseconds	95 microseconds
2	676 microseconds	2.704 milliseconds	3.844 milliseconds	9.025 milliseconds
4	≈.5 seconds	≈7 seconds	≈14 seconds	≈81 seconds
8	≈2.42 days	≈1.7 years	≈6.9 years	≈210 years
16	≈1.38 billion years	≈91 trillion years	≈1.5 quadrillion years	≈1.4 quintillion years

(ii) ElcomSoft Co. claims that they can try 2.8 billion passwords per second. [Use high end GPU graphics card.]
Check: https://blog.elcomsoft.com/2020/12/breaking-passwords-with-nvidia-rtx-3080-and-3090/ for latest information.

	lower case	lower/upper	lower/upper/digits	lower/upper/digits/symbols
1	9 nanoseconds	19 nanoseconds	22 nanoseconds	34 nanoseconds
2	241 nanoseconds	966 nanoseconds	1.373 microseconds	3.223 microseconds
4	≈163 microseconds	≈2.61 milliseconds	≈5.28 milliseconds	≈29.1 milliseconds
8	≈74.6 seconds	≈5.307 hours	≈21.6 hours	≈27.4 days
16	≈.5 million years	≈32 billion years	≈.5 trillion years	≈.5 quadrillion years

Dictionary attacks

An attack that tries different passwords defined in a list, or a database, of password candidates.

- Basis: passwords are very rarely random. People tend to follow very similar conventions when selecting a password.
- Creating a dictionary that contains commonly occurring passwords would help an attacker guess correct passwords more efficiently than brute-forcing.

Hybrid attacks

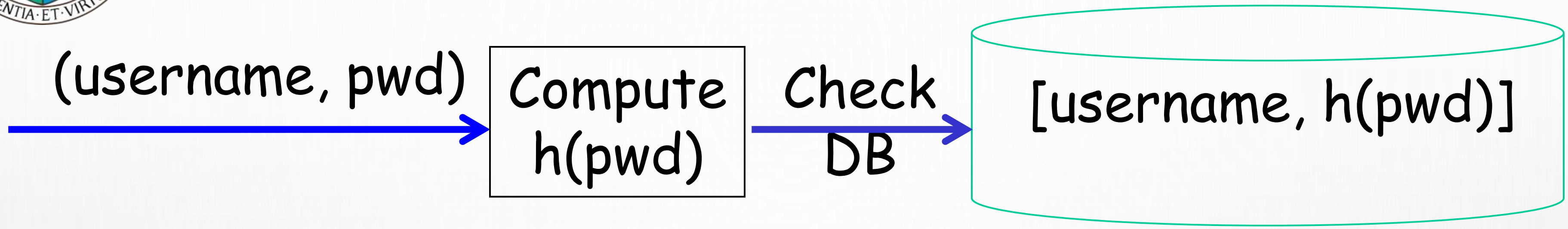
A modification of the dictionary attack that tries different permutations of each dictionary entry. In a hybrid attack, the utility starts with a dictionary entry and tries various alternative combinations.

For example, if the dictionary entry were "lord," the hybrid attack utility would look for these possible alternatives: "Lord", "l0rd", "1ord", "10rd" and so on.

Go back and try ElcomSoft:

Trial versions can be obtained here:
https://www.elcomsoft.com/download.html

Recall:



In theory, if the attacker have enough time and space, we can do the following:

Pre- compute all possible h(pwd) values for all possible pwds:

pwd1	h(pwd1)
pwd2	h(pwd2)
pwd3	h(pwd3)
…….	

Length	domain	# of pwds
1	Lower letters	26
	Lower/upper letters	52
	Lower/upper letters, digits	62
4	Lower letters	26^4
	Lower/upper letters	52^4
	Lower/upper letters, digits	62^4

$14{,}776{,}336 \approx 14\text{M}$

Q: If we use 128 bit hash value, how much space we require for the attack?

Each entry is [pwd, h(pwd)]:
Total size: 14M x [4+ 128/8] bytes ≈ 280M

Q: How about if the length of password is at least 8 characters, each character can be any printed symbols (95 symbols)?

\# of possible pwds $\geq 95^8$ $\approx 7 \times 10^{15} = 7{,}000$T

(i) I.e., we need to compute 7×10^{15} hashes. If 1 sec, we can compute 1 Million (10^6) hashes,

We need 7×10^9 seconds $\approx$ 200+ years

(ii) How about space (128 bits hashes)?

7,000T x [8 + 128/8] bytes = 168,000T

Q: why the server can store the table: [username, h(pwd)]?

Total # of people in the world $\approx 7.5 \times 10^9$ only

Actual table size = (# of users) x 24 bytes only.

Rainbow table

- A rainbow table is a lookup table offering a time-memory tradeoff used in recovering the password from its hash value.
  - **Given:** H: a password hash function; P: a finite set of passwords.
  - **Goal:** Pre-compute a data structure that given any output h of the hash function, can either locate an element p in P such that H(p) = h, or cannot determine p.

p, if can be found, is the password!

Critical ideas

- **Hash chains** are used to decrease the space requirement. The idea is to define a **reduction function** R that maps hash values back into values in P. For example,

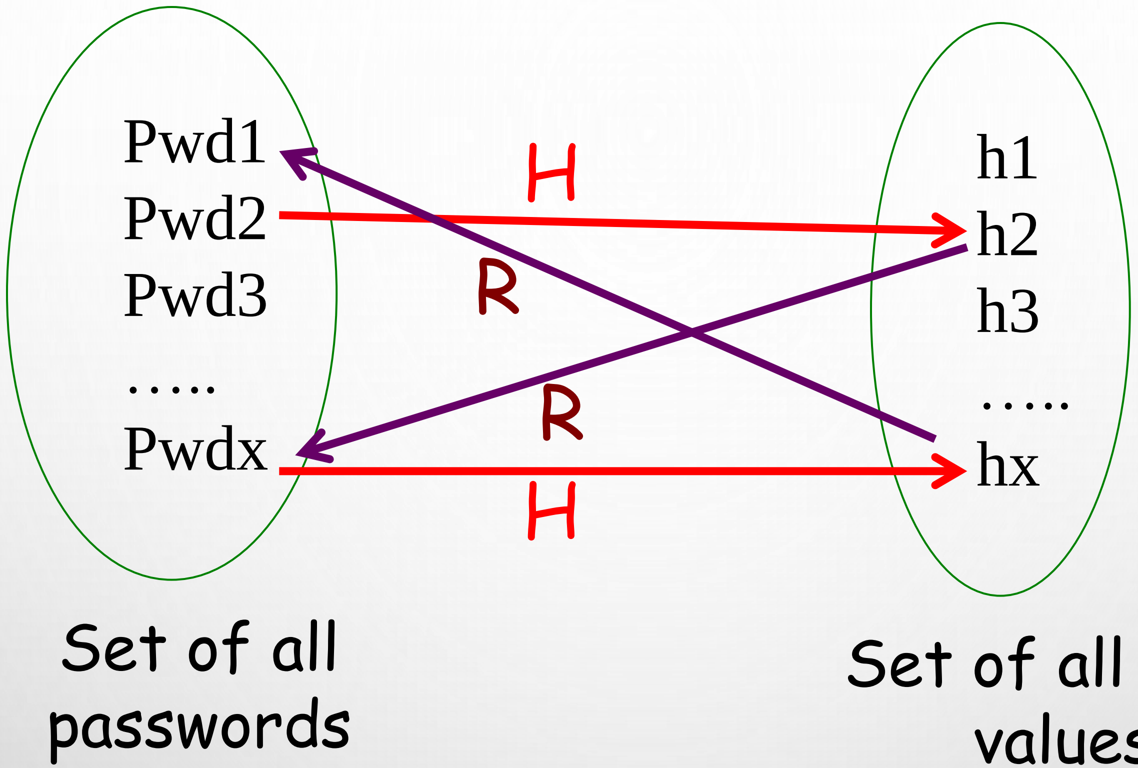
$$\text{aaaaaa} \xrightarrow[H]{} \text{281DAF40} \xrightarrow[R]{} \text{sgfnyd} \xrightarrow[H]{} \text{920ECF10} \xrightarrow[R]{} \text{kiebgt}$$

(i) R is NOT an inverse function of H

i.e., R(H(p)) may not = p

(ii) R can be quite arbitrary
(see the example in the next slide)

A conceptual diagram: What pre-computation does?



Take an arbitrary password p_i, compute $h_i=H(p_i)$, then compute $p_{i+1} = R(h_i)$, obtain $h_{i+1} = H(p_{i+1})$, …..

Hash chain: p_i -> h_i -> p_{i+1} -> h_{i+1} -> …. -> p_k -> h_k

1. Compute as many chains as you can within reasonable amount of time.

Hash chain C1: p_i -> h_i -> p_{i+1} -> h_{i+1} -> .... -> p_k -> h_k
Hash chain C2: p'_i -> h'_i -> p'_{i+1} -> h'_{i+1} -> .... -> p'_k -> h'_k

....

Hash chain Cn: .....

2. Q: What could be the length of the chain?

H() and R() can be computed fast
=> length of the chain = tens, hundreds, thousands?

Q: what is the effect of this length? [Hint: pay attention to how to work with a rainbow table for password cracking]

A simple example for reduction function R():

Take MD5 as an example, hash value has 128 bits.
Assume our password is of 8 characters.

$H(p_i) = h_i$ (128 bits)

Define $R(h_i)$ as follows:
Divide h_i into 16 8 bits patterns.
$R(h_i) = x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8$
where
x_1 = character of the ASCII (1^{st} 8 bits of h_i)
x_2 = character of the ASCII (3^{rd} 8 bits of h_i)
…
x_8 = character of the ASCII (15^{th} 8 bits of h_i)

R() is not an inverse of H(), but easy to compute!

Two problems left:

(i) How to store a chain?

(ii) How to use the chain to crack passwords?

(i) Only store the starting password and the last hash of each chain

Hash chain C1: p_i -> h_i -> p_{i+1} -> h_{i+1} -> .... -> p_k -> h_k
Hash chain C2: p'_i -> h'_i -> p'_{i+1} -> h'_{i+1} -> .... -> p'_k -> h'_k
....
Hash chain Cn: .....

Rainbow table: (p_i, h_k); (p'_i, h'_k), ...

Q: how much saving in storage we can get?

Depends on the length of a chain.
Let L be the length of a chain.
Saving:

(L/2) x [64+128] bits => 64+128 bits

If L = thousands, we save thousand times of memory

Note: we can store passwords as both end points too

Also, no need to compute for all possible passwords as we will NOT have enough time to do it, just compute as many as you can.

How to crack passwords with these chains?

Given a hash value x, we want to find the corresponding password

- (a) compute a chain starting with x by applying R, then H, then R, and so on.
- (b) If at any point we observe a value matching one of the endpoints in the table, we get the corresponding starting point and use it to recreate the chain.
- (c) If this chain contains the value x, the immediately preceding value in the chain is the password p that we seek; otherwise, ignore the match and continue to extend the chain of x looking for another match. [Q: why sometimes the chain we chase does not contain x?]

Sample chains in the table

$$\mathbf{aaaaaa} \xrightarrow[H]{} 281DAF40 \xrightarrow[R]{} sgfnyd \xrightarrow[H]{} 920ECF10 \xrightarrow[R]{} \mathbf{kiebgt}$$

$$\vdots$$

$$\cdots\cdots\cdots \quad FB107E70 \xrightarrow[R]{} bvtdll \xrightarrow[H]{} 0EE80890 \xrightarrow[R]{} \mathbf{kiebgt}$$

Note: Multiple hashes can map to the same password since R is not 1-1, so we may not get to the password using the chain whose end point matches our value.

E.g., if we are given the hash 920ECF10, we would compute its chain by first applying R

$$920ECF10 \xrightarrow[R]{} \mathbf{kiebgt}$$

- Since "kiebgt" is one of the endpoints in our table, we then take the corresponding starting password "aaaaaa" and follow its chain until 920ECF10 is reached

$$\mathbf{aaaaaa} \xrightarrow[H]{} 281DAF40 \xrightarrow[R]{} sgfnyd \xrightarrow[H]{} 920ECF10 \xrightarrow[R]{} \mathbf{kiebgt}$$

- Thus, the password is "sgfnyd".

Flaws of simple chains

Simple hash chains have several flaws. Most seriously, if at any point, two chains collide (produce the same value), they will merge and is not useful. Since the entire chains are not stored, not possible to detect this scenario.

$$Hous \xrightarrow[H]{} 012BXFD \xrightarrow[R]{} aaaa \xrightarrow[H]{} 2823FAB \xrightarrow[R]{} sgfn \xrightarrow[H]{} \cdots$$

$$bbbb \xrightarrow[H]{} 345A0BC \xrightarrow[R]{} xiao \xrightarrow[H]{} 2823FAB \xrightarrow[R]{} sgfn \xrightarrow[H]{} \cdots$$

Q: How to resolve it?

Solutions

Hint: When the values collide in two chains, the values may not be at the same position, e.g. fouth value in Chain C1 is the same as the second value of the Chain C2

How about using more than one reduction functions, say R1, R2, R3?

$$C1: p1 \xrightarrow{R1} h1 \longrightarrow p2 \xrightarrow{R2} h2 \longrightarrow p3 \xrightarrow{R3} h3 \; \ldots\ldots$$

$$C2: p1' \xrightarrow[R1]{} h2 \longrightarrow p3' \xrightarrow[R2]{} h3' \longrightarrow \; \ldots\ldots$$

$p3' \neq p3 \quad h3' \neq h3$

Even collision occurs, higher chance the rest of the chain may not be the same

Remarks on rainbow tables

Generating a rainbow table requires a significant amount of time.

- Advantages of rainbow tables
  - They can be used repeatedly for attacks on other passwords;
  - Rainbow tables are much faster than dictionary attacks;
  - The amount of time needed on the attacking machine is greatly reduced.

Q: Do you know what the length of the chain affects?

Both the storage and speed for tracing the chains

RAINBOW TABLE ATTACK

- Here's an illustration comparing brute-force attacks and rainbow table attacks:

Password Characteristics	Example	Maximum time to break using brute force	Maximum time to break using rainbow tables
8-digit password of all letters	abcdefgh	1.6 days	28 minutes
9-digit password of letters and numbers (mixed case)	AbC4E8Gh	378 years	28 minutes
10-digit password of letters and numbers (mixed case)	Ab4C7EfGh2	23,481 years	28 minutes
14-digit password of letters, numbers, and symbols	1A2*3&def456G$	6.09e + 12 years	28 minutes

Q: Will rainbow table always be able to identify the corresponding password?

No, usually not enough time to compute all possible passwords, i.e., there exist passwords not appearing in any of the stored chain.

Q: Elcomsoft claims that they have a way to catch these missing passwords and store them in something called "Thunder Tables*"

(i) How to identify these missing passwords?
(ii) How to store them in a compact way?

*https://blog.elcomsoft.com/2009/05/thunder-tables/

Exercise:

(i) comp79

(ii) cyb410

Fig 1(a): Rainbow Table

Start of Chain	End of Chain
kate	lugirl
lein	gboy
alice	attic

Fig 1(b): hash function

Password	Hash
kate	gfi479
hoco	comp79
snofi	dte345
lein	eo4cg1
imini	d89qwf
mexia	st456h
alice	cyb410
xuey	i63gt7
notes	2sjlp
succes	st456h

Fig 1(c): R1 reduce function

Hash	Password
gfi479	hoco
eo4cg1	imini
cyb410	xuey

Fig 1(d): R2 reduce function

Hash	Password
comp79	snofi
d89qwf	mexia
i63gt7	notes
cyb410	succes

Fig 1(e): R3 reduce function

Hash	Password
dte345	lugirl
st456h	gboy
2sjlp	attic
comp79	great
cyb410	heiut

Countermeasure against rainbow table attack: Avoid pre-computation

A simple defense is:
- When hashing the password, we include a random sequence of bits as input along with the user-created password.

- These random bits are known as a **salt**
  - Make brute force, dictionary, and rainbow table attacks much more difficult

PASSWORD SALTS

A table showing the MD5 hash values for unsalted and salted passwords

	Password	Hashed Value
No Salt	this1sAg00dPASSword!!	a5a5baa0c16166260e9ef8a48dbde112
Salted	6789o3uigtbgeat7this1sAg00dPASSword!!	53cffc58904a10b9dcc40345433862dc
Salted	v8734ihv6!nre432tthis1sAg00dPASSword!!	28b8f782262a890b4d730f8001d23bd5
No Salt	love	b5c0b187fe309af0f4d35982fd961d7e
Salted	12bg55tygsdf4gvi9yrdslove	65c96e15930d34dd9a9ce916b81fb044
Salted	879rughq2ebt5dfxcasedlove	a35436c0e0f2821db2703c1983a641ab

Salting makes rainbow table attack much more difficult

(username, pwd) → (i) append salt to pwd
(ii) Compute h() value

Username	Salt	h(pwd,salt)
smyiu	45akb56..io	346901
zoe	879328....ry	56445
......		

Q: Why rainbow table attack is more difficult?

** UNAUTHORIZED PASSWORD CRACKING IS ILLEGAL **

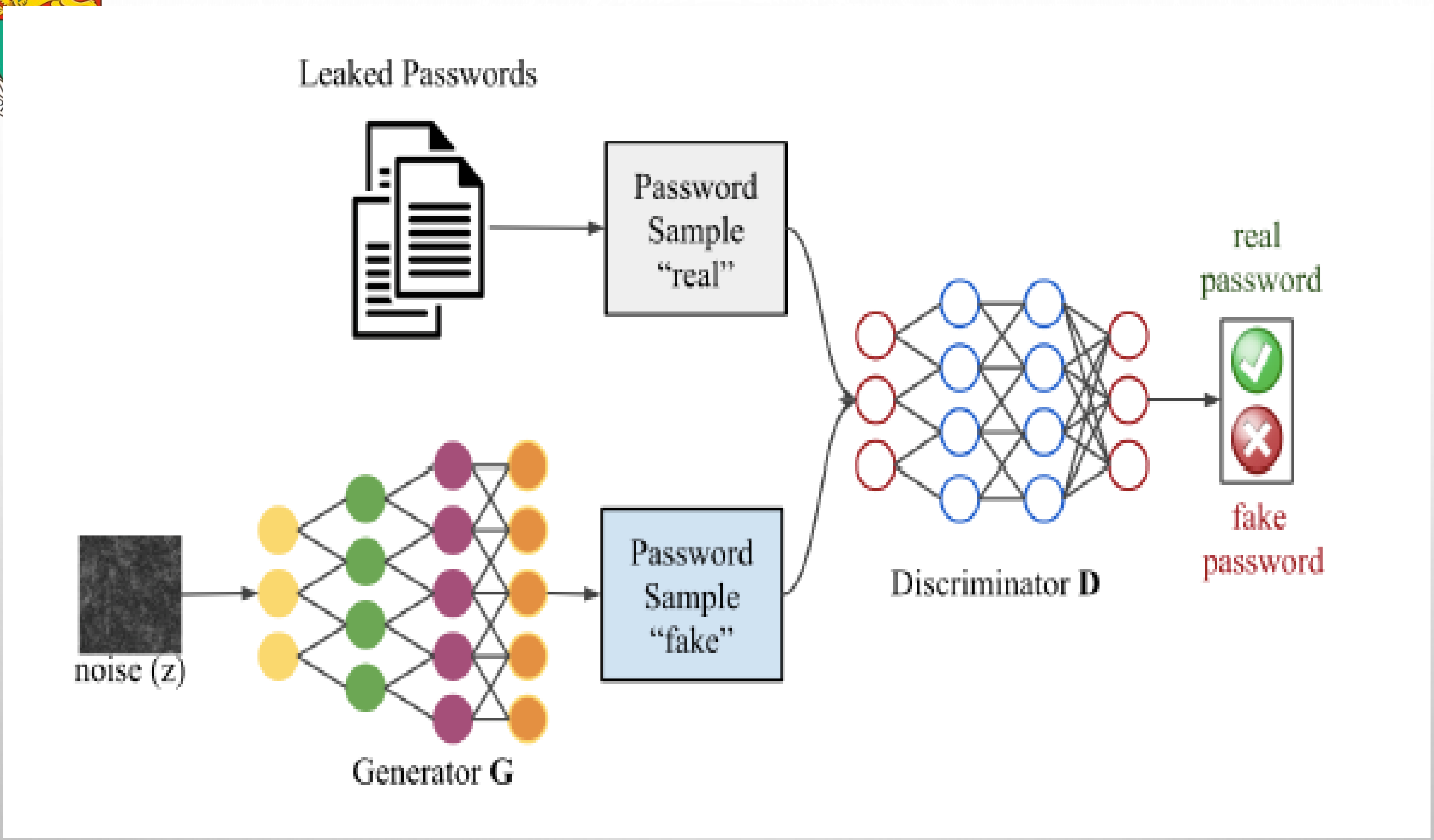
- NEVER ATTEMPT TO CRACK PASSWORDS UNLESS YOU HAVE SPECIFIC, WRITTEN AUTHORITY TO DO SO.

- The main reason to crack a password is to obtain evidence that is protected by that password. You can obtain permission to crack a password from the computer owner or a court. In cases where the computer owner is unwilling to provide the permission to crack a password, a court order is needed.

State-of-the-art

Idea: Recent tools try to model how human comes up with passwords
e.g. rule-based attacks: generating password guesses by transforming dictionary words.

John the Ripper (JTR)
HashCat



PassGAN (deep learning approach)

Highlights of PassGAN

- Use generative adversarial networks (GANs)
- A GAN has two neural networks:
  - (i) a generative deep neural network G
  - (ii) a discriminative deep neural network D
- Goal of G: produce "fake" samples that will be accepted by D.
- Goal of D: learn how to distinguish fake samples produced by G from real ones.
- Target: G can produce samples cannot be distinguished by D

Usage: Use G to produce as many possible passwords as possible for checking!

Note:
Although password cracking seems to be an old problem, a lot of work is still going on.

E.g.
"Key stretching technique"
- Strengthen a "weak" key, submitted by a user, to make it more difficult for brute-force attack.

PBKDF2
- Part of RSA Lab's public-key crypto standards (PKCS)
- Idea: derive a key based on a password by iterating a process x times (e.g. x = 1,000)

Technically, PBKDF2 works as follows:

DK = PBKDF2(PRF, password, salt, n, len)

PRF: pseudorandom function (e.g. keyed HMAC)

User-defined password

# of times the process will be repeated

Length of the derived key (DK)

$DK = T_1 + T_2 + \ldots + T_{len/hlen}$,
where hlen: length of output of PRF
and
T_i is F(password, salt, n, i)

F will run PRF n times by XOR n results of PRF:
t_1 = PRF(password, salt + i), t_2 = PRF(password, t_1), t_3 = PRF(password, t_2), ….
F(password, salt, n, i) = t_1 XOR t_2 XOR t_3..

What is the implication?

More difficult for attackers to launch the attack as they need to go through the same process to get the final hashed value.

But at the same time, user authentication process will also take longer.

Q: any better balance of it?

Project ideas:

- (last year) PassGAN + wireless router pwd setting
- Survey study + evaluation
- AI + password cracking
- Solving the key stretching time consuming issue?

Take-home exercises (no need to hand in)

Wireless network password cracking:

Q1: Which of the followings are more secure?
WEP, WPA, WPA2, WPA3

Q2: Do you know what protocols (e.g. encryption algorithms) they use?

E.g. WEP uses RC4

Q3: Which one has been cracked?

Others:

Q4: What are the differences between online and offline password attacks?

Q5: What is two factor authentication? Why it is more secure?

Q6: There are many password cracking tools. Try some of them (quite a number of them are free)

Next Thursday: we are going to have a laboratory session, so prepare your portable for the lab session, we will install Kali Linux.
Remark: OSCP exam (www.offensive-security.com)

REFERENCES

1. SAM MARTIN AND MARK TOKUTOMI, "PASSWORD CRACKING"
2. HTTP://EN.WIKIPEDIA.ORG/WIKI/RAINBOW_TABLES
3. HTTP://KESTAS.KULIUKAS.COM/RAINBOWTABLES/
4. MICHAEL G. SOLOMON, K RUDOLPH, ED TITTEL, NEIL BROOM AND DIANE BARRETT, COMPUTER FORENSICS JUMPSTART, SYBEX, 2ND EDITION, 2011.
5. HASHCAT. (2017) HTTP://HASHCAT.NET
6. J. THE RIPPER (2017) HTTP://OPENWALL.INFO/WIFI/JOHN/MARKOV
7. HITAJ ET AL., PASSGAN: A DEEP LEARNING APPROACH FOR PASSWORD GUESSING, HTTPS://ARXIV.ORG/PDF/1709.00440