Lecture 2: User Authentication Ep.1

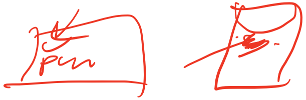
05506044 System Security

Dr. Rungrat Wiangsripanawan

First Objective

Understand and being able to explain

- Authentication ~ บุคคล
- Identification → ยื่น ยืนว่าเป็น คนคนนั้นจริง
- Verification → ยืนยัน ตัวบุคคล



- authentication factors
  - Know/posses/Are/Do

Explain = Know meaning and can give Examples

Second Objective

Understand and being able to explain

- Something you know
- Password based authentication
  - Client Side Threat/Control ความเสี่ยง
    - Password Guessing
    - Password Exposure (Shadow Surfing)
    - Malware
  - Server Side Threat/Control
    - Cryptographic Mechanism – Cryptographic Hashing/ hash( )
    - Salt
    - Access Control
  - Concept behind One Time Password (OTP)
    - Lamport OTP

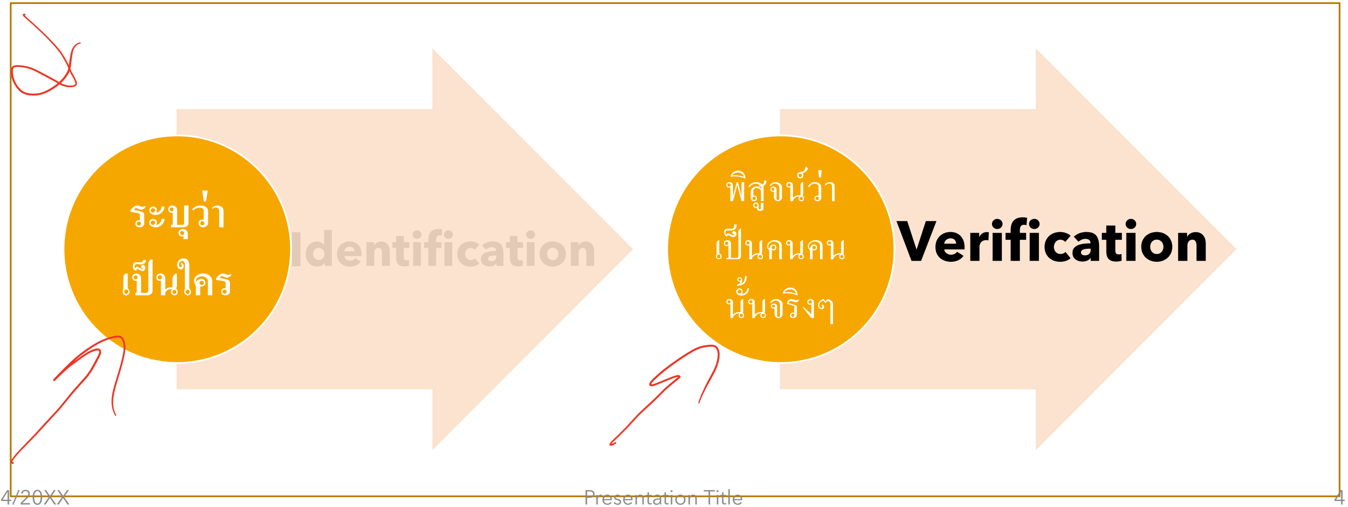
Salt

login

Explain = Know meaning and can give Examples

User Authentication = Identification + Verification

User Authentication



Yes/
No

<u>Why</u> is Authentication?

Fundamental building block and first line of defense in most computer security context.

<u>Two reasons</u> ในการ **authenticate user**

- **Access control** (การควบคุมการเข้าถึง) - ระบบส่วนใหญ่ใช้ user identity ในการตัดสินใจว่าจะให้เข้าถึงระบบหรือไม่
- **user accountability.** บันทึก user identity ในการทำ logging ของเหตุการณ์ทาง security เพื่อการตรวจสอบ (audit)

Accountability - The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity.

User Authentication: What is it? I

- process of verifying a user's identity [5].

กระบวนการในการตรวจสอบ user identity

authentication process มีสองขั้นตอน:

- Identification step ขั้นตอนการระบุตัวตน
  - specify an identifier to the security system
- Verification step ขั้นตอนการตรวจสอบว่าเป็นคนคนนั้นจริงๆ
  - Presenting or generating authentication information that corlaborates the binding between the entity(person) and the identifier.

ไม่เหมือน message authentication

Authentication Factors

ปัจจัย

Something you Know

- Password/Pin

Something you Posses

- Key card/

Something you are

- Fingerprint/Face/Retina => Static Biometric

Something you do (Some books)

- Your key stroke/Voice => Dynamic Biometric

key

วิธีการในการ Authentication [2]

A subject, (a user or an entity), must provide *information* to enable the computer system to confirm its identity.

This *information* could be one or a combination of the following four means which based on something the individual

KNOWS(รู้)	**POSSESES (ถือ)**	**IS (เป็น)**	**DOES (ทำ)**
e.g. password, PIN	**e.g. key, token, smartcard**	**(static biometrics) - e.g. fingerprint, retina**	(dynamic biometrics) e.g. voice pattern, handwriting characteristic and typing rhythm.

can use alone or combined

all can provide user authentication

all have issues

Something you know

Password

Passwords Are Like Underwear

Passwords are like underwear...
Change yours often.

Passwords are like underwear...
Don't share them
with friends.

Passwords are
like underwear...
The longer, the
better.

Passwords are like
underwear...
Be mysterious.

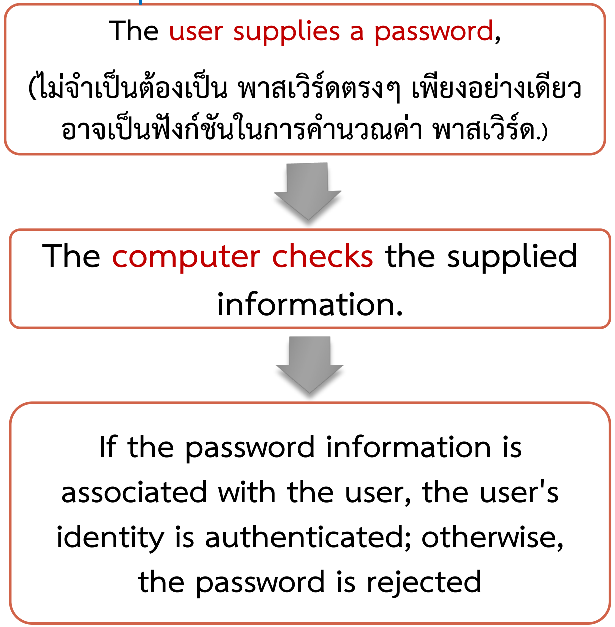
Passwords are like
underwear...
Don't leave yours
lying around.

Something you know: Password

- password-based method.
  - simplest method
  - possibly the worst but its use is widespread ☹
- Passwords involves authentication on the basis of what an entity ....

ให้ คิดช่องโหว่ของพาสเวิร์ดมีอะไรบ้างก่อนไปหน้าต่อไป

How password works:



The user supplies a password,

(ไม่จำเป็นต้องเป็น พาสเวิร์ดตรงๆ เพียงอย่างเดียว อาจเป็นฟังก์ชันในการคำนวณค่า พาสเวิร์ด.)

The computer checks the supplied information.

If the password information is associated with the user, the user's identity is authenticated; otherwise, the password is rejected

Top 25 hit passwords in 2018 and 2017 : Splash Data

Hack

- 2018

1. 123456
2. password
3. 123456789
4. 12345678
5. 12345
6. 111111
7. 1234567
8. sunshine
9. qwerty
10. iloveyou

- 11. princess
- 12. admin
- 13. welcome
- 14. 666666
- 15. abc123
- 16. football
- 17. 123123
- 18. monkey
- 19. 654321
- 20. !@#$%^&*
- 21. charlie
- 22. aa123456
- 23. donald
- 24. password1
- 25. qwerty123

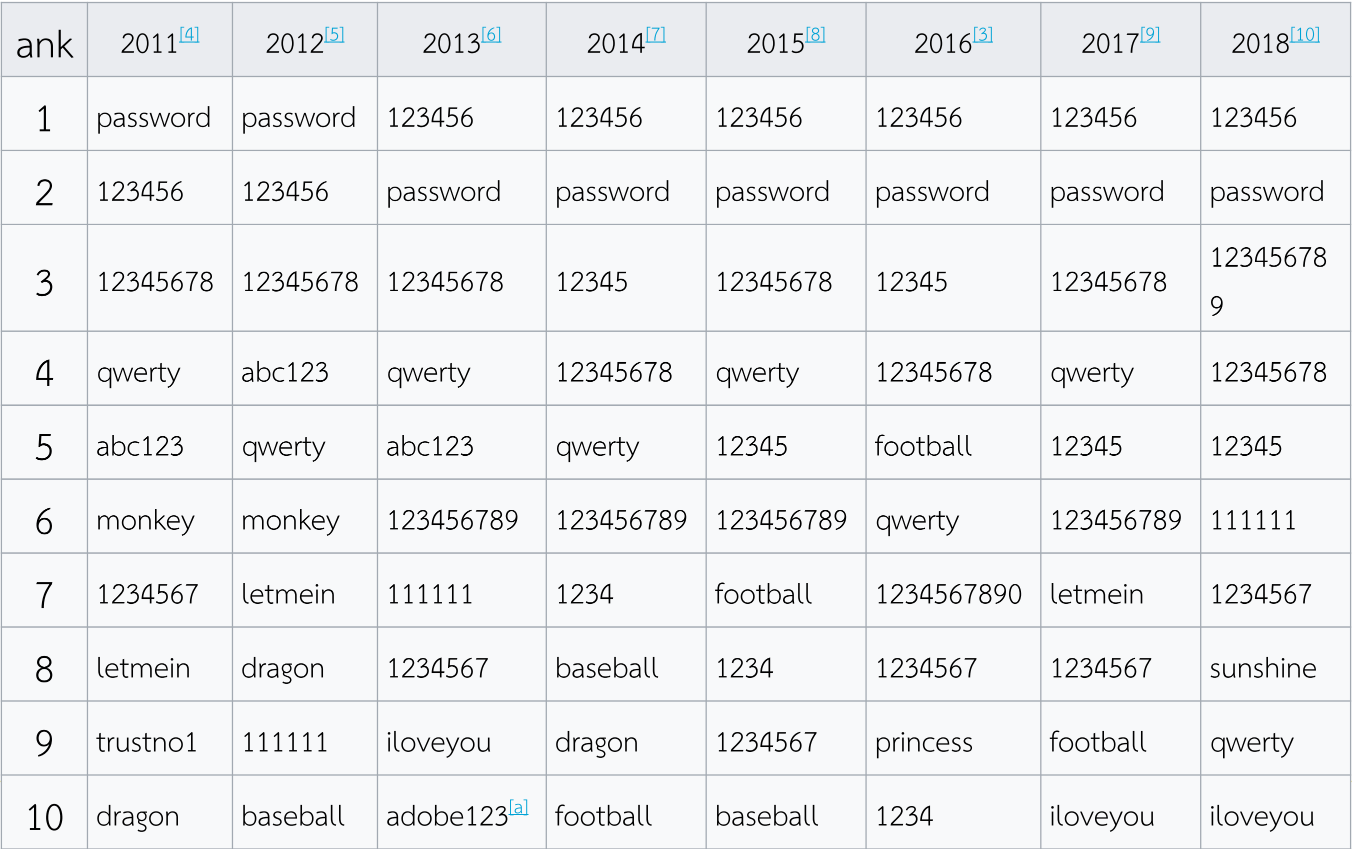
- http://techland.time.com/2012/10/25/these-are-the-25-worst-passwords-of-2012/

- 123456
- password
- 12345678
- qwerty
- 12345
- 123456789
- letmein
- 1234567
- football
- iloveyou

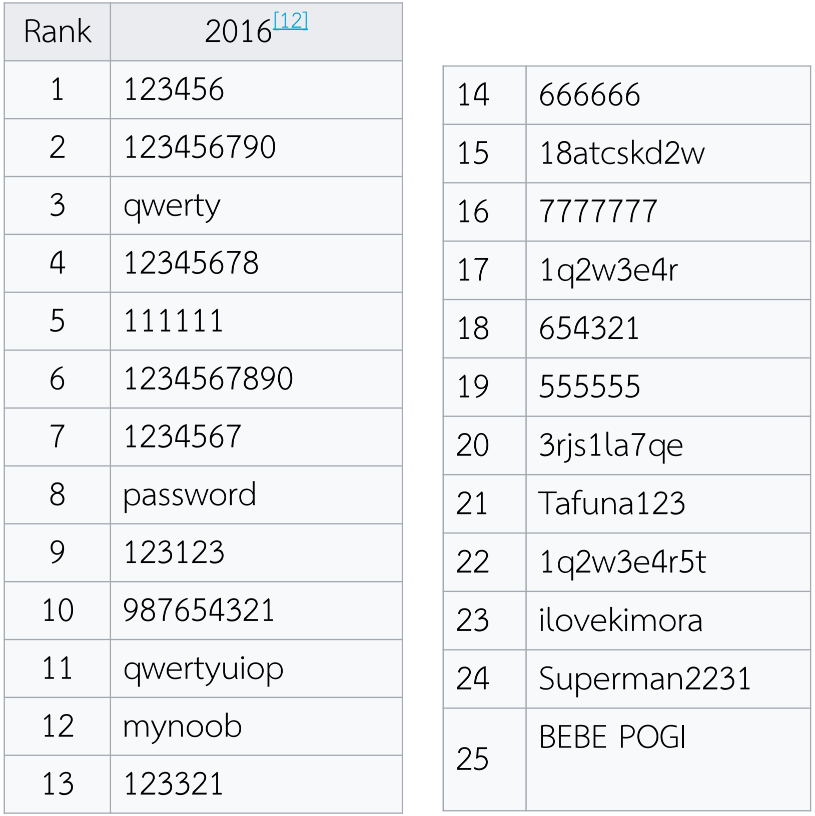
- admin
- welcome
- monkey
- login
- abc123
- starwars
- 123123
- dragon
- passw0rd
- master
- hello
- freedom
- Whatever
- qazwsx
- trustno1

http://newsfeed.time.com/2014/01/20/the-25-worst-passwords-of-2013/

https://en.wikipedia.org/wiki/List_of_the_most_common_passwords



ank	2011[4]	2012[5]	2013[6]	2014[7]	2015[8]	2016[3]	2017[9]	2018[10]
1	password	password	123456	123456	123456	123456	123456	123456
2	123456	123456	password	password	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345	12345678	123456789
4	qwerty	abc123	qwerty	12345678	qwerty	12345678	qwerty	12345678
5	abc123	qwerty	abc123	qwerty	12345	football	12345	12345
6	monkey	monkey	123456789	123456789	123456789	qwerty	123456789	111111
7	1234567	letmein	111111	1234	football	1234567890	letmein	1234567
8	letmein	dragon	1234567	baseball	1234	1234567	1234567	sunshine
9	trustno1	111111	iloveyou	dragon	1234567	princess	football	qwerty
10	dragon	baseball	adobe123[a]	football	baseball	1234	iloveyou	iloveyou



Rank	2016[12]
1	123456
2	123456790
3	qwerty
4	12345678
5	111111
6	1234567890
7	1234567
8	password
9	123123
10	987654321
11	qwertyuiop
12	mynoob
13	123321
14	666666
15	18atcskd2w
16	7777777
17	1q2w3e4r
18	654321
19	555555
20	3rjs1la7qe
21	Tafuna123
22	1q2w3e4r5t
23	ilovekimora
24	Superman2231
25	BEBE POGI

Password manager Keeper compiled its own list of the 25 most common passwords in 2016, from 25 million passwords leaked in data breaches that year

https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

Rank	2019[13]
1	123456
2	123456789
3	qwerty
4	password
5	1111111
6	12345678
7	abc123
8	1234567
9	password1
10	12345
11	1234567890
12	123123
13	000000
4	Iloveyou
15	1234
16	1q2w3e4r5t
17	Qwertyuiop
18	123
19	Monkey
20	Dragon

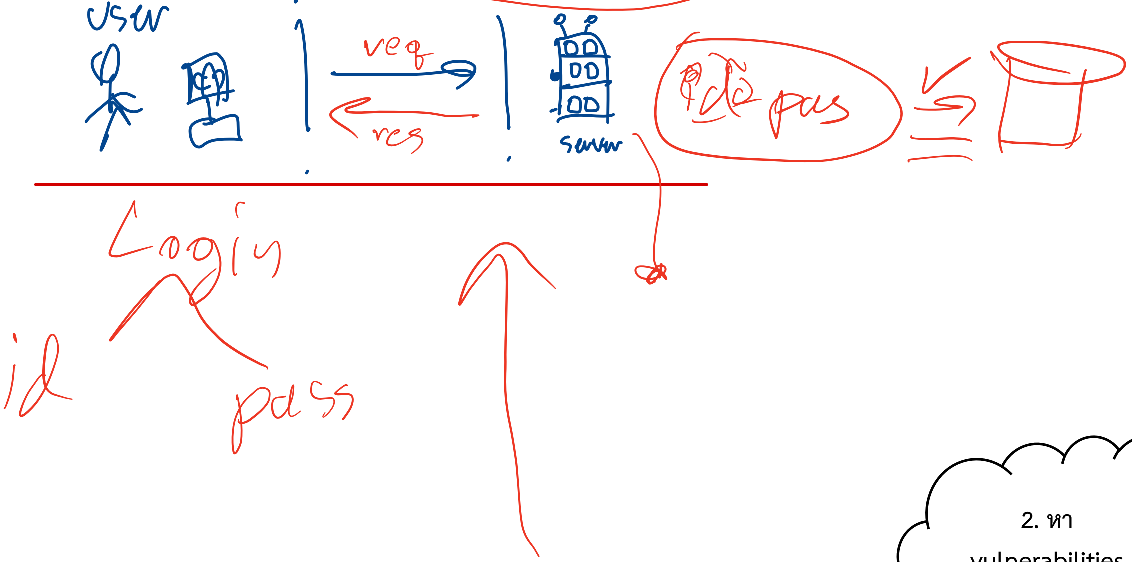
The National Cyber Security Centre (NCSC) compiled its own list of the 20 most common passwords in 2019, from **100 million passwords** leaked in data breaches this year

https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

8 passwo

Rank	2019[4]	2020[5]
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Threats against password systems : user side



2. หา vulnerabilities and threats

Threat : Password guessing

- It is pretty much always possible to attempt to guess a password on-line.
- Ways to do it..

Try all possible passwords (exhaustive or brute force attack)

Try frequently used passwords

Try passwords likely for the user

Threat: Password exposure (Vul)

An "eavesdropper" may see the password when it is typed.	• Typing very slowly isn't a good move. • ตัวอย่าง Threat เช่น Shoulder Surfing
Some user write their passwords down, even next to their computer.	• This is not a good idea
Some users pass *their* password to others. same	• Even if you appropriately protect your password, the other person may not.
Shadow Surfing	• ชะโงกดู ;)

id pass

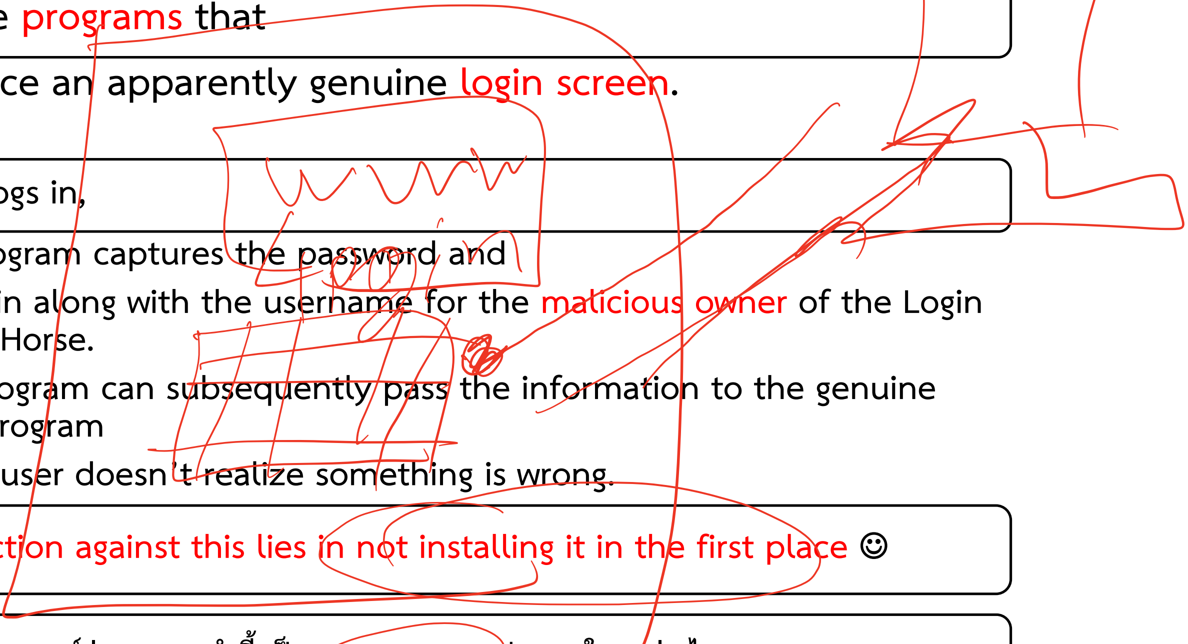
Threat: Login Trojan Horses Ex. Key logger

These are programs that

- produce an apparently genuine login screen.

The user logs in,

- the program captures the password and
- stores in along with the username for the malicious owner of the Login Trojan Horse.
- The program can subsequently pass the information to the genuine login program
- so the user doesn't realize something is wrong.



The protection against this lies in not installing it in the first place ☺

ให้ นักศึกษา วิเคราะห์ว่า การกระทำนี้ เป็น attack/threat ประเภทใด อย่างไร

T1: Password Guessing: Brute force

- All password systems are vulnerable to somebody guessing the correct password.
- A brute force attack involves trying every possible password.
- Brute force always works.
  - Eventually
  - The important factor is that this guessing is unlikely within the lifetime of the password.
- With a brute force attack, you start with the letter a, then try aa, ab, ac, and so on until zz; then you try aaa, aab, aac, and so on
- If passwords are words consists of A-Z and can be any length of 1-8 characters
  - $26^1 + 26^2 + 26^3 + 26^4 + 26^5 + 26^6 + 26^7 + 26^8$

    $= 26^9 - 1$ approx $5*10^{12}$

8

มาจากไหน ????

A-Z Time

From Kaufman, Perlman, and Speciner Network Security—Private Communication in a Public World book

Humans are *incapable* of securely storing high-quality cryptographic keys, and they have *unacceptable speed and accuracy* when performing cryptographic operations.

(They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)

T1: Poor passwords → Dictionary attacks

- แม้ suitable length แต่อาจเป็นคำจาก dictionary.
- A dictionary attack exploits this.
- Dictionaries of common words
  - => sets of passwords to try.
  - => steps through the words in a dictionary and tries them as passwords.
- This dictionary attack may not succeed but is quite fast.

Tailored dictionary attacks

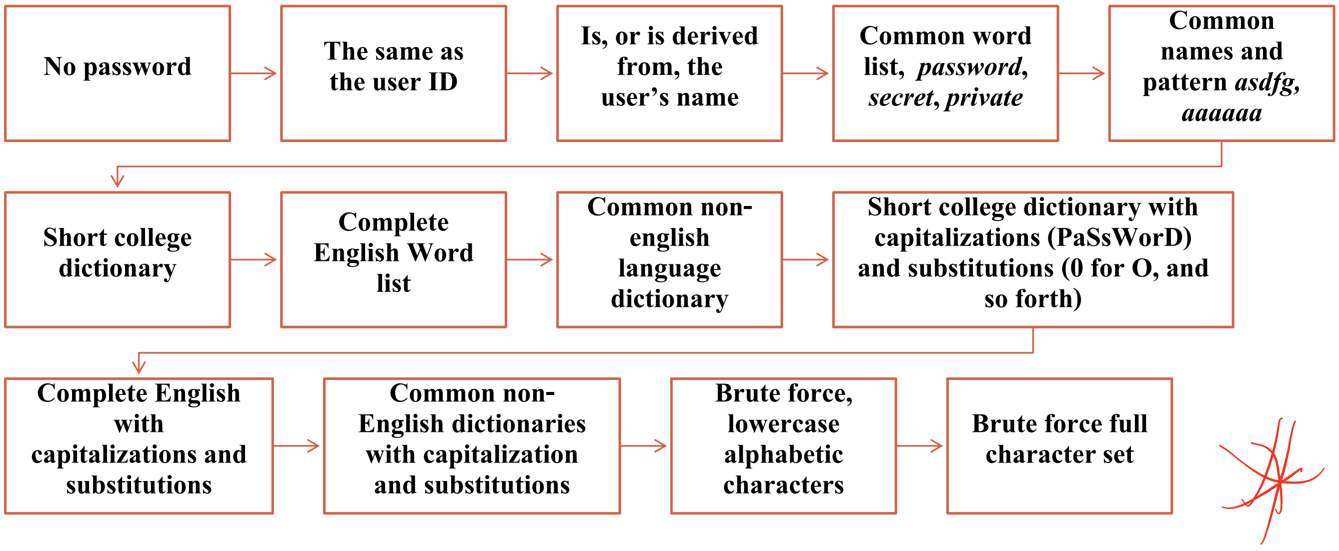
- It is possible to be more specific than a complete English dictionary, or targeted from another source.
- For example, users may like cars or motorbikes, and a suitable dictionary could be a list of car or motorbike brands.
  - Or sports teams or players names.
- Users may use even more personal information for passwords:
  - Birthdates, family names, pet names.

mike19

mike 089

Example of Guessing Step

mike cat



No password → The same as the user ID → Is, or is derived from, the user's name → Common word list, *password*, *secret*, *private* → Common names and pattern *asdfg*, *aaaaaa* → Short college dictionary → Complete English Word list → Common non-english language dictionary → Short college dictionary with capitalizations (PaSsWorD) and substitutions (0 for O, and so forth) → Complete English with capitalizations and substitutions → Common non-English dictionaries with capitalization and substitutions → Brute force, lowercase alphabetic characters → Brute force full character set

Summary: Guessing Passwords

[illegible]

Exhaustive search (brute force): Try all possible combinations of valid symbols up to a certain length.

Intelligent search: search through a restricted name space, e.g. passwords that are somehow associated with a user like name, names of friends and relatives, car brand, car registration number, phone number,..., or try passwords that are generally popular.

- Typical example for the second approach: dictionary attack trying all passwords from an on-line dictionary.

You cannot prevent an attacker from accidentally guessing a valid password, but you can try to reduce the probability of a password compromise.

weak

How to improve password security from

Password Guessing

Control: Choosing secure passwords

- Expected time (in seconds) to guess a password is

  N/R

  N: Size of the set of possible passwords.

  R: Number of passwords that can be tested in a second.

- For a randomly generated password of length 8, with each element with being a lower case character a-z, and 1 millisecond testing time, it takes nearly 3 years => Proof by yourself..and tell me if the statement is correct.
- A password chosen as above is secure but hard to remember.

Trying to improve passwords

Using pronounceable passwords makes remembering passwords easier. But …

- … this reduces the number of possible passwords, since the number of vowels is likely to be fairly high. Every third character could be a vowel.

Using a ``pass-phrase'' is another alternative.

- A string of characters usually is longer than password
  - Harder to do brute force
- …She sells see shell by the see shore..

BUT BUT Dictionaries can be used for either of these scenarios so …

- … we could use pass-phrases with intentional misspellings, odd capitalizations and symbol replacements. Or insert some numbers.
- She ell C shells ByE da c-shor

Hybrid attacks

- แต่ยังเจอ Hybrid attacks.
  - ซึ่งใช้ dictionary as เป็นฐาน ก่อน
  - แล้วค่อยทำ brute-force attack โดย
    - add prefix or suffix characters ลงไปในคำใน dictionary
      - □one-upped constructed password (a password where a single characters differ from its from in the dictionary),
      - □ a two–upped constructed password and so on.
    - ex. Password1 is a one-upped for password
  - or to replace each lower case “L” with 1, or “O” with 0, and so on.
- ดังนั้น the hybrid attack อยู่ระหว่าง dictionary and the brute force attack ในเรื่องของ
  - เวลาที่ใช้ (time consumed), จำนวนของพาสเวิร์ดในการลอง (the number of passwords tried) เป็นต้น

Protective mechanisms → 5 [illegible]

- 5 ครั้ง lock

Keep track of incorrect password attempts:

- Limit the number of account/passwords guesses per connect attempt.
- Or lock the account when a threshold is exceeded.
  - Although the attacker can use this to perform a DOS attack. ☹
- Or raise an alarm and try to trace the intruder.
  - Administrators on the system can observe when a limit is reached.

Slowly process passwords, it doesn't make much difference to a legitimate user (user tries for his own password) but it makes a lot to the processing speed of an attacker (attacker must try all)***

Strong passwords/password systems (รหัสที่ strong)

- Change passwords every 45 days.
- Minimum length of eight (or higher) characters.
- Must contain at least one alpha, one number, and one special character.
- Alpha, number, and special characters must be mixed up and not appended to the end.
  - For example, fg#g3s^hs5gw is good, abdheus#7 is not.
- Cannot contain dictionary words.
- Cannot reuse any of the previous five passwords.
- Minimum password age of ten days.
- After five failed logon attempts, password is locked for several hours.

วิธีการทั้งหมด ในหน้านี้เกิดจากมาตรการต่างๆ ไม่ได้ใช้ เทคนิควิธีการอะไรเลย เป็นตัวอย่างหนึ่ง ของการนำ Control แบบ Policy มาใช้เพื่อเพิ่ม Security ให้กับระบบ

BUT The flip-side

- As security managers add rules for passwords users resist.
- It is tempting to write the password down.
- It is tempting to simply rotate the password.
- It is more likely that structure will actually be contained in a password.

user → เซอร์ ขอบอกเขาด้วย

ปัญหา ของ การนำ Control มาใช้ แล้วไม่สำเร็จ ;) เพราะผู้ใช้ไม่ให้ความร่วมมือ

How can you remember anyway?

- คำแนะนำการสร้าง และ จำ password ที่ดี
  - choose a phrase
  - take the first letter from each word as your password.
- Choosing a well-known phrase is not such as good idea.
- A rolling stone gathers no moss.

  Arsgnm
- My cat Boris has a long tail and 16 teeth.

  McBhalta16t

“Online” กับ “Offline” guessing ต่างกันอย่างไร

- Online (“live”) guessing will usually face restrictions on the number of attempts.
- Offline ไม่มีปัญหานี้, และอาจเกิดขึ้น โดยการที่เจ้าของพาสเวิร์ด หรือ system administrator ไม่รู้เรื่อง.
- Offline attacks จะเกิดขึ้น ถ้าผู้บุกรุกสามารถเข้าถึง password file ได้
- หรือเกิดจากการที่ มีการดักจับ พาสเวิร์ดในการส่งข้อมูล (Or if the transmission of a password is intercepted)
  - The interception
    - may capture the password directly, which means the communication wasn’t adequately protected => plaintext
    - or it may be some function of the password, possibly an encrypted or a hashed version => หรือเป็นฟังก์ชันของพาสเวิร์ด เช่น แฮช

Online” versus “Offline” guessing Cont.

- ความแตกต่างระหว่าง “online” และ “offline” ไม่ใช่ประเด็นสำคัญ
- ความสำคัญอยู่ที่
  - The issue that really matters is whether the number of “guesses” is restricted or not. (จำนวนครั้งในการเดาว่ามีการจำกัดหรือไม่)
  - The distinction completely changes the way in which attackers are likely to operate.
- If you can guess without restriction it is probably worthwhile trying, at least a dictionary attack. เดากี่ครั้งก็ได้ไม่ว่า ก็ต้องเดาสิ ;)
  - If you cannot guess without restriction then another approach is probably more useful. (ถ้าไม่สามารถเดาแบบ ไม่มีกฎเกณฑ์จำนวนครั้งในการเดาได้ ก็หาวิธีอื่น ดีกว่า => Hack password file)

crypto graphic hash function

ใช้เก็บ password สุดท้าย

แล้วระบบ เก็บ Password ที่ไหน??? อย่างไร (ในมุมมองของผู้ดูแลระบบ)

Server Side

Plaintext System Password List

Store plaintext password in the file with the user ID

User ID	Password
Manee	_ad3d3%
Piti	123456
Chujai	asdfedg
. . .	. .

Problems

- Attacker targets the password file.
- Attacker can dump memory to access the password file
- Attacker can get the password file from the back up disk.
- Attackers sniff the user ID and pwds through the communication.
- ผู้ใช้บางคนใช้ พาสเวิร์ดเดียวกันในทุกระบบ ดังนั้นถ้า password file บนเครื่องใดเครื่องหนึ่งหรือระบบใดระบบหนึ่งถูก hack ยิ่งเป็น plaintext ด้วยแล้ว ....

Solution: Encrypt the password file

1-way-fn

$f(x) \leftarrow y$

$x \rightarrow f'(y)$

$\boxed{y = x+5} = m$

รู้ y หา x ได้
แต่ถ้ารู้ x หา y ไม่ได้
เพราะ

Offline Dictionary Attack

Example: Unix – The most of pupular Operating System in opening age..;)

- In the UNIX operating system, users passwords are not stored.
  - Hashes of the passwords are stored rather than the plaintext.
- The hash of a message is a “fixed length fingerprint” of the block of data.

One-way Functions and password file

- For cryptographic protection we can use one-way functions (hash function)

pwd

f(x)

hash Algo

- Definition: A one-way function *f* is a function that is relatively *easy to compute* but *hard to reverse*.
  - Given an input *x* it is easy to compute *f*(*x*), but given an output *y* it is hard to find *x* so that *y* = *f*(*x*)

Instead of the password *x*, the value *f*(*x*) is stored in the password file.

- When a user logs in and enters a password, say *x'*,
- the system applies the one-way function *f* and compares *f*(*x'*) with the expected value *f*(*x*).

hash

Hash**** Password file

User ID	Hash code
Manee	XcRqBAu2wfRQo DF
Piti	;yw9iDUld8iyh”mjmQ
Chujai	1=b’g’boik’;DF;s;peW
	. .

- 1. Manee enters her user id and her password:
  - Manee: _ad3d3%
- The system computes hash of Manee’s password
  - h(Manee_pwd) = h(_ad3d3%) = XcRqBAu2wfRQo DF
- The system looks up its password file to check whether Manee’s Hash code is the same.
  - If so, it allows this user to access the system.

Problems: 1. Duplicate password (users ที่มี password เหมือนกัน Hash จะเหมือนกัน จะเห็นได้เลย ว่าใช้ password เดียวกัน)

2. เกิดปัญหา Offline Dictionary Attack กับ password file นี้

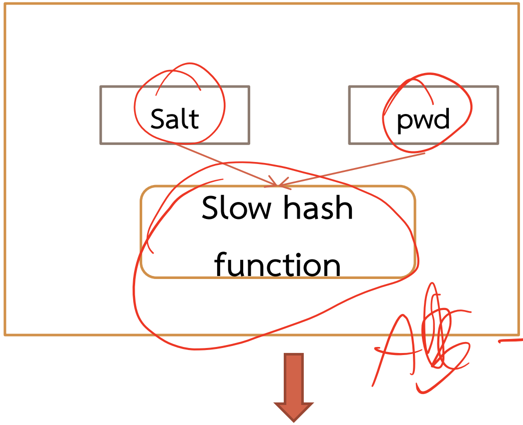
Example: Offline dictionary attack on plaintext file

User ID	Hash code
Manee	XcRqBAu2wfRQo DF
Piti	;yw9iDUld8iyh’’mjmQ
Chujai	1=b’g’boik’;DF;s;peW
	.

Dictionary words	Hash of this word
a	C9j57’i[dqedf
abaca	dfdfEF82SFW
aback	;YOouh1df,ie2
abacus	1S2Ded[.le43f
….	……
pretty	XcRqBAu2wfRQo DF
….	….
sunshine	;yw9iDUld8iyh’’mjmQ
….	….
target	1=b’g’boik’;DF;s;peW
….	….
zygote	Ded;,di,meo’2

This table is computed offline

Storing a new password : Hashing + Salting



User ID, h(pwd,salt)

Ab + 123 →

User ID	Salt value	Hash code
Manee	486	XcRqBAu2wfR Qo DF
		. .
		.

Password Salting

- Salt is a value that is randomly generated.
- The hash of the combination of the salt and the password, is stored, along with the salt (not necessarily in the same place).

user ID	salt value	password hash
Alice	3487	hash(3487||password_Alice)
Bob	8254	hash(8254||password_Bob)
Oscar	1098	hash(1098||password_Oscar)

How using salt slow down attacker

Salt

- If the whole password file is disclosed, the intruder can compute the hash of a password and compare it against all hashes (สามารถคำนวณ ตาราง Hash ไว้ใช้ดูกับทุกคนได้เลยทันที)
- Using salt the Duplicate attack isn’t possible, we can only check against one-user at a time. (เพราะแต่ละคน salt ไม่เหมือนกัน)
- Even if the salts are known
  - Not easy to link the passwords of users, either in the same system or between systems. (user มักใช้ password เดียวกับทุกๆ ระบบแต่การมี Salt ทำให้ hash ของ พาสเวิร์ดนั้นจะไม่เหมือนกัน ทั้งที่พาสเวิร์ดเหมือนกัน)
  - Without salt this linking is possible because two users with the same password would have the same stored password hash. (แกะได้หนึ่งระบบเข้าได้หมด)

Verifying

1. A user enters her user ID (Ex. Manee) and her pwd (p)

1. Manee, pretty

2. The system looks up the password file to get the salt value (s)

3. The system
   - computes the hash of (s,p) and
   - compares if h(s,p) equals Manee's hash codes that stored in the password file.

Authenticate

Login

id

pwd

Client

reg

Server

User ID	Salt value	Hash code
Manee	486	XcRqBAu2wfRQo DF
Piti	1690	;yw9iDUld8iyh''mjmQ
Chujai	4815	1=b'g'boik';DF;s;peW
		.

Example

User ID	Salt value	Hash code
Manee	486	XcRqBAu2wfRQo DF
Piti	1690	;yw9iDUld8iyh”mjmQ
Chujai	4815	1=b’g’boik’;DF;s;peW
		. .

Password file with salt hardens the offline dictionary attack

Dictionary words	Salt	Hash of this word
a	0	dfs57’i[dqedf
a	.....	...
a	65535	;;peeh1df,ie2
abacus	0	03lf0tr[.le43f
....	...	......
pretty	0	Drt;grtfg;;4o
pretty	...	...
pretty	486	XcRqBAu2wfRQo DF
....		....
pretty	65535	;i6j’iy93g;up’dfrok;yp
....		....
target		1=b’g’boik’;DF;s;peW
....		....
zygote	65535	0ydir[g;up’liurk;

สรุป Adding salt

- ค่าที่มีขนาดคงที่ (fixed length) called ‘salt value’.
  - เดิมใช้
    - Time at which the password is assigned to the user. (เวลาตอนที่ตั้งพาสเวิร์ด)
  - เดี๋ยวนี้ใช้
    - A pseudorandom or random number.
- It serves three purposes
  - Prevent duplicate passwords
  - Increase the difficulty of offline dictionary attacks.
    - For a b-bit salt,
      - □ Possible passwords is increased by a factor of 2^b
  - Nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them.

การป้องกัน มาตรฐาน ดาว password

*

- hash function - ✓
- salt (รหัสค่า hash)
- control ให้ผู้ควบคุม (Admin)

[เอาค่า เขียน บรรทัดเดียว]

security - รปภ

ใช้ Attack ยิ่งหา port ที่มี ช่องโหว่

Example: Unix=>How Did UNIX "Encrypt“*** Passwords?

- เดิม UNIX ใช้ hashing algorithm called crypt to protect its passwords.
  - This isn’t actually the same as simply running crypt from the command line in UNIX, that is an encryption algorithm. crypt() is a built-in Perl function.
- The protection is through the one-way transformation of the password by the one-way hash function.
- There is no way to obtain a password that has been “hash with crypt.
- จริงๆ สิ่งที่ Cryptographic function ที่ Unix ทำเพื่อ protect password ไม่ใช่ ~~encrypt~~ คือ Hash ***

Unix: Salting

- Each "encrypted" password is
  - 11 characters in length, and is combined with
  - a random 2-character salt to get a 13-character "stored" password.
- The salts must be random, and from a large enough space that the chance of two users having the same salt is low. For example, if create two users with a password of yellow, the stored passwords could be:

```
XcRqBAu2wfRQo
 5pjoJnbeVEUbw
```

- With newer versions of UNIX, other hash algorithm options rather than crypt are deployed.
- One option is MD5 , which provides stronger hashing and irreversibility.

Unix

Password storage in UNIX: Where?

- Early versions of UNIX contained a file /etc/passwd, which stored all of the user IDs and encrypted passwords in the same file.
- This file was a text file and contained the user ID, encrypted password, home directory, and default shell. The following is a sample passwd file:
-
```
root:6T1E6qZ2Q3QQ2:0:1:Super-User:/:/sbin/sh
John:.D532YrN12G8c:1002:10::/usr/john:/bin/sh
mike:WD.ADWz99Cjjc:1003:10::/usr/john:/bin/sh

. . .

 cathy:BYQpdSZZv3gOo:1010:10::/usr/cathy:/bin/sh
frank:bY5CQKumRmv2g:1011:10::/usr/frank:/bin/sh
tom:zYrxJGVGJzQL.:1012:10::/usr/tom:/bin/sh
karen:OZFGkH258h8yg:1013:10::/usr/karen:/bin/sh
```

encryp

The general format for the passwd file

- *Username:passwd:UID:GID:full_name:home directory:shell*
- *Username:* Stores the username of whom the account belongs to.
- *Passwd:* Stores the user's encrypted password.
  - If shadow files are used, an x appears in this location.
- *UID:* The user ID or the user identification number, generally chosen by the system.
- *GID:* The group ID or group identification number, which reflects the native group (base group of membership).
- *Full name:* This field usually contains the user's full names but is not mandatory.
- *Home Directory:* Stores the location of the user's home directory.
- *Shell:* Stores the user's default shell, which is what runs when the user first logs onto the system.
- **/etc/passwd is world readable ☹**

What do you think is the problem of this

Shadow Files

- A solution to the readability problem.
- UNIX splits the passwd file information into two files.
- The passwd file still exists and contains everything except the encrypted passwords.
- A second file, shadow file , was created.
  - This contains the encrypted password and is only accessible to the root user.
- This information is stored centrally.
  - /etc/passwd → ทุกคน
  - /etc/shadow → root

Shadow files: The fields

- *username:passwd:last:min:max:warning:expire:disable*
- *username:* The user's name of the account. There should be a corresponding line in the passwd file with the same username.
- *passwd:* Contains the encrypted password.

Only the first two fields are mandatory.

- *last:* Contains the date of the last password change.
- *min:* The minimum number of days until the password can be changed.
- *max:* The maximum number of days until the password must be changed.
- *warning:* The number of days that the user is warned that the password must change.
- *expire:* The number of days in which the password expires and the account is disabled.
- *disable:* The number of days since the account has been disabled.

/etc/passwd VS /etc/shadow

- Using "shadow passwords" is the preferred way of storing password hashes.
- You shouldn’t have any system that still stores password hashes in /etc/passwd. Update if you are
- Consider the following pair of /etc/passwd and /etc/shadow files:

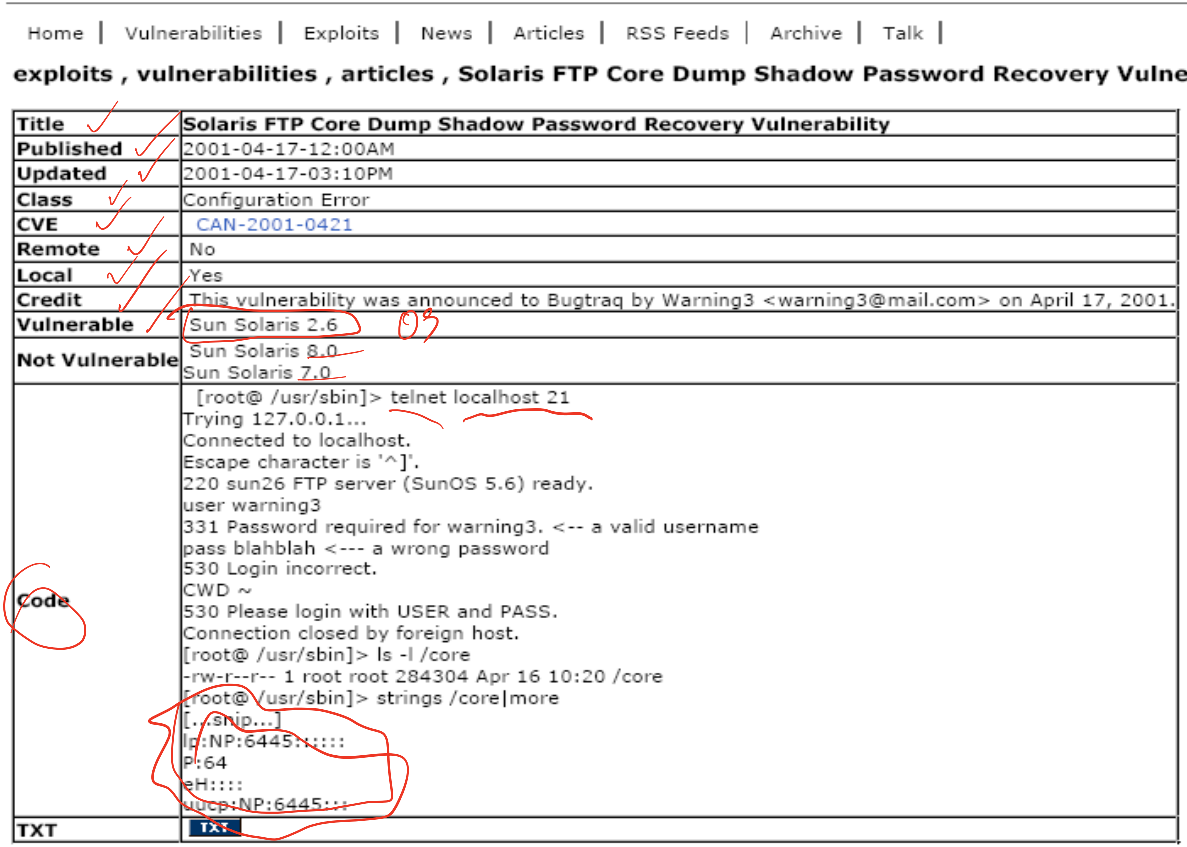
```
root:x:0:1:Super-User:/:/sbin/sh          root:6T1E6qZ2Q3QQ2:6445::::::
eric:x:1001:10::/usr/eric:/bin/sh         eric:T9ZsVMlmal6eA:::::::
John:x:1002:10::/usr/john:/bin/sh         John:.D532YrN12G8c:::::::
mike:x:1003:10::/usr/john:/bin/sh         mike:WD.ADWz99Cjjc:::::::
. . .                                     ...
tim:x:1009:10::/usr/tim:/bin/sh           tim:sXu5NbSPLNEAI:::::::
cathy:x:1010:10::/usr/cathy:/bin/sh       cathy:BYQpdSZZv3gOo:::::::
```

root

How safe are shadowed systems?

- Using shadow files is safer because require root access
- Although shadow files require root access, there were attacks that can be used to acquire a copy of the shadow file without obtaining root access directly.
  - For example imapd (a mail related server) and telnet were, at one time, both guilty of dumping core on occasion complete with the shadow file in the core where it was user-readable.
  - It is possible to recover information from the core.

CPU



Home | Vulnerabilities | Exploits | News | Articles | RSS Feeds | Archive | Talk |

exploits , vulnerabilities , articles , Solaris FTP Core Dump Shadow Password Recovery Vulnerability

Title ✓	**Solaris FTP Core Dump Shadow Password Recovery Vulnerability**
Published ✓	2001-04-17-12:00AM
Updated ✓	2001-04-17-03:10PM
Class ✓	Configuration Error
CVE ✓	CAN-2001-0421
Remote ✓	No
Local ✓	Yes
Credit ✓	This vulnerability was announced to Bugtraq by Warning3 <warning3@mail.com> on April 17, 2001.
Vulnerable ✓	Sun Solaris 2.6 03
Not Vulnerable	Sun Solaris 8.0 Sun Solaris 7.0

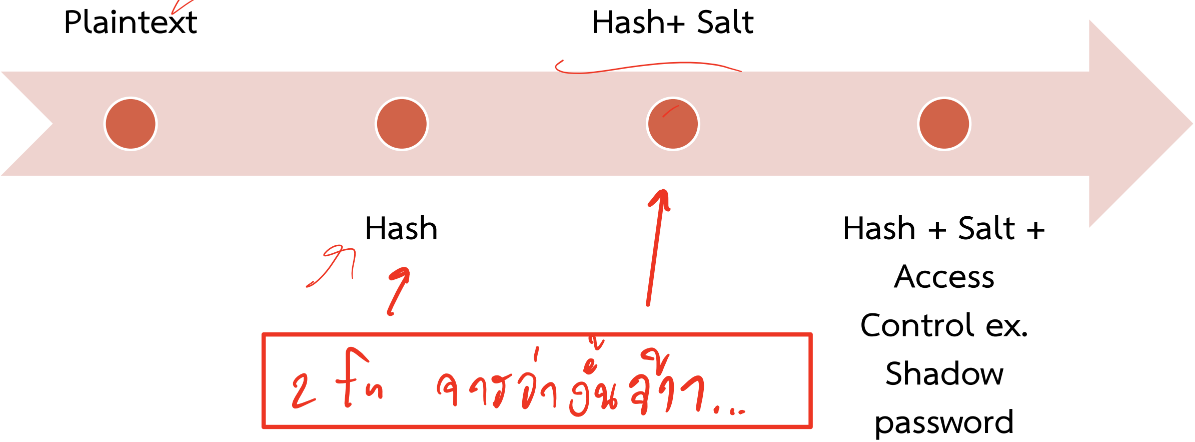
Code

```
 [root@ /usr/sbin]> telnet localhost 21
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 sun26 FTP server (SunOS 5.6) ready.
user warning3
331 Password required for warning3. <-- a valid username
pass blahblah <--- a wrong password
530 Login incorrect.
CWD ~
530 Please login with USER and PASS.
Connection closed by foreign host.
[root@ /usr/sbin]> ls -l /core
-rw-r--r-- 1 root root 284304 Apr 16 10:20 /core
[root@ /usr/sbin]> strings /core|more
[...snip...]
lp:NP:6445::::::
P:64
eH::::
uucp:NP:6445:::
```

TXT	TXT

CVE

Password file stories



Plaintext

Hash

Hash+ Salt

Hash + Salt + Access Control ex. Shadow password

2 fn จารย์จำอั้นลิ้วา...

- Do all these mechanisms stop an adversary to launch an dictionary attck to the password file or any password that he/she can eavesdrop?

Summary: Protecting passwords (file) using Access Control

- Password lists in a system must be well protected.
  - If back-ups have password files then they have to be protected too.
- The operating system maintains a file with user names and passwords
- An attacker could try to compromise the confidentiality or integrity of this password file. (how???)
- Options for protecting the password file:
  - cryptographic protection,
  - access control enforced by the operating system, ✓
  - a combination of cryptographic protection and access control, possibly with further measures to slow down dictionary attacks.

One time password: Intuitive Idea

- With a one-time password system the user and the system have a list of valid passwords such that
  - each one is valid only once. ใช้ password ใหม่ทุกครั้ง
- An observed password leaks no information about the other passwords.
- Provided the passwords are not obviously correlated, this system is immune to eavesdropping
  - This property means that even though the adversary can eavesdrop the password he cannot reuse it or even they can do offline dictionary attack with this eavesdropped password, there is no use.
- Problem:
  - Does this mean that the server has to store heaps of passwords since each of them is used only once.?
  - So, how these number of passwords are stored.???

Oh

One Time Password

OTP

Problems with the intuitive idea.

ไม่เวิร์ค

From the point of view of the server

- they need to store more information. ต้องเก็บหลายๆ พาสเวิร์ด

From the point of view of the user

- they are more likely to write down passwords and be less careful in choosing them.
- They are not going to be able to rely on repeated usage to reinforce their memory of a single password.
- ต้องจำหลายๆ พาสเวิร์ด

การป้องกัน

1. crytographic hash fn

2. ฟังก์ชันใช้ ตัวไดน P___ , Argon

PBKDF2

GSM

Examples:

54.00

- f(x) = x+1
  - System
    - prompts with the value of ‘x’. Ex. 1
  - User
    - computes x+1 in this case 2
    - Reply with 2
  - System
    - Check if equal allow access.

- f(x) = r(X), where r(x) is the functions to generate the random number
  - System
    - prompts with the value of ‘x’. Ex. 22
  - User
    - computes r(X) ex. 23456
    - Reply with 22
  - System
    - Check if equal allow access.

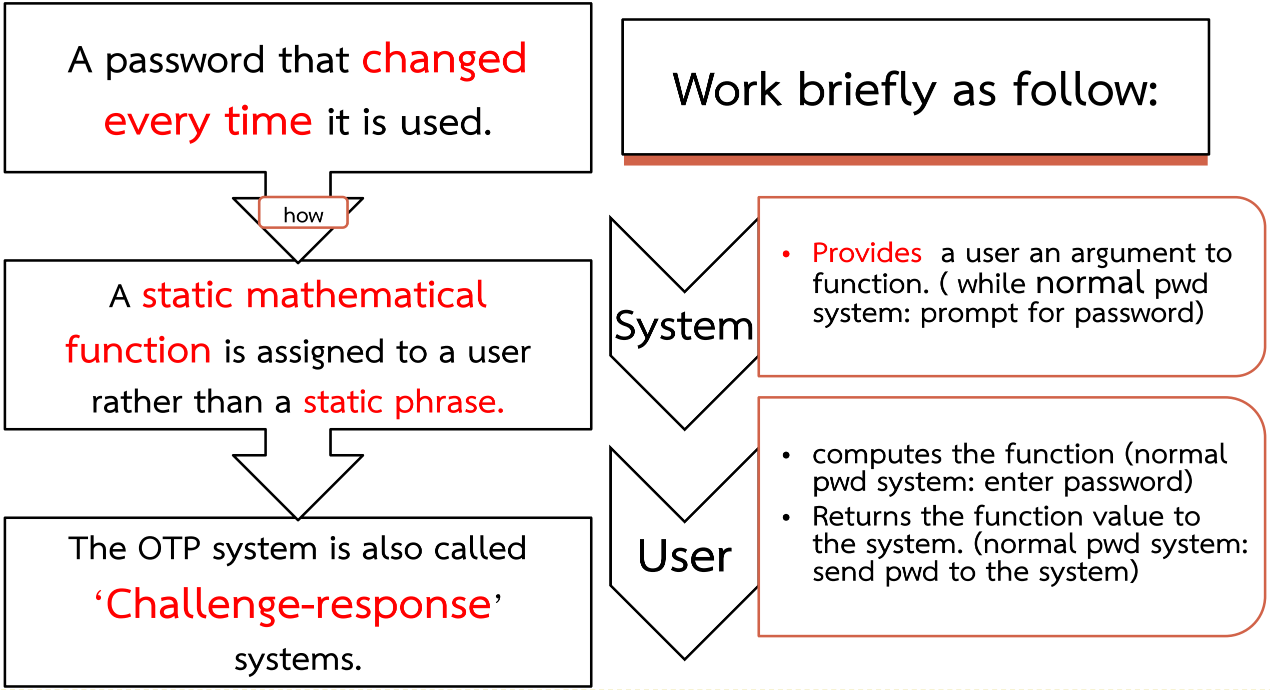
Note that …both the system and user/host must have the same **random generator.**

Examples

- f(E(x)) = E(D(E(x)) +1)
- System
  - Computer E(x)
  - Send E(x) to user
- User
  - decrypts (D(E(x)) to get x
  - computes E(x+1)
  - Send E(x+1) to System
- System
  - Computes E(x+1) and verifies…
    - If equals allow access

- ลองทำตัวเป็น System กับ User ในกรณีฟังก์ชั่นข้างล่างค่ะ
- f(a1,a2,a3,a4) = (a2,a4,a1,a3)

One time password (OTP)



A password that changed every time it is used.

how

A static mathematical function is assigned to a user rather than a static phrase.

The OTP system is also called ‘Challenge-response’ systems.

Work briefly as follow:

System

- Provides a user an argument to function. (while normal pwd system: prompt for password)

User

- computes the function (normal pwd system: enter password)
- Returns the function value to the system. (normal pwd system: send pwd to the system)

Lamport’s one time password

telnet : port 23

SSH : port 22

- One time passwords based on one-way functions
- Credited to Leslie Lamport work in[2]
- Used in S/KEY, a one-time password system used by Unix like OS (before SSH)

Alice, the user, remembers a password. สำหรับฝั่ง user จำพาสเวิร์ดเดียวเหมือนเดิม

Bob, the server (computer), has a database in which it stores, for each user แต่ฝั่ง server bob บางสิ่งเปลี่ยนไป

The username U_i.

A counter n

that decrements each time Bob authenticates the user.

The hash value $x_n=h^n(\text{password})$, for some specified hash function $h^i(X)=h(h^{i-1}(X))$ and $h^0(X)=X$

How does it work?

- Alice has a workstation, and Bob is the server.
- To authenticate we use the following protocol:

  Workstation → Bob : Alice // User พิมพ์ลงไปที่ Work station บอกว่า ฉันชื่อ Alice

  Bob → Workstation : n // Bob (Server) ส่ง ค่า n กลับมาให้เครื่อง Workstation ที่ Alice ใช้อยู่

  Workstation → Bob : h^{n-1}(password) // WorkStation นำค่า password ที่ alice พิมพ์เข้าไปทำการ ใส่ hash function จำนวน n-1 ครั้ง แล้ว ส่งค่านี้ กลับไปให้ bob

- Bob checks if หลังจาก bob ได้รับค่า จะทำการตรวจสอบโดย นำค่าที่ได้ h^{n-1}(password) ไปเข้าฟังก์ชั่น hash อีกครั้ง แล้วเช็คกับ ค่า hash จำนวน n ครั้ง (h^n ที่ตัวเองเก็บไว้)

  $h(h^{n-1}(\text{password})) = h^n(\text{password})$

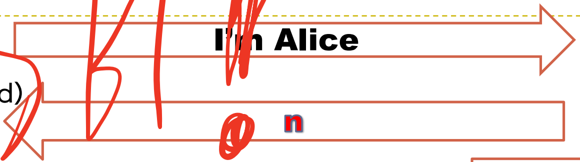
- If it does ถ้าเท่ากัน แสดงว่า Alice ใส่ พาสเวิร์ดถูกต้อง
  - then Bob accepts the communicating party as Alice.
  - If it doesn’t Bob rejects the communication.

ออกข้อสอบแล้ว

HASH!

Alice (work station)

Bob (server)


I'm Alice

n

- Enter password (pwd)
- Compute
  - 0 -> h^0(pwd)= pwd (ไม่ได้ทำอะไร ;)
  - 1 -> h^1(pwd)= h(pwd) เข้า hash 1 ครั้ง
  - 2-> h^2(pwd) = h(h(pwd)) เข้า hash อีกครั้ง …
  - n-1->h^{n-1}(pwd) = h(h^{n-2}(pwd) = h(…(h(h(pwd)…))

n = 100

alice = 15 รอบ

alice → 16 แทน n

h^{100-1}

- Bob stores (สิ่งที่ Bob ต้องเก็บ)
  - Alice id
  - Alice counter –n 100 → 99 …
  - Current Alice Hash value
    - h^n(pwd)

h^{99}(pwd) = h^{98}(pwd)

h^{n-1}(pwd)

- checks if

  h(h^{n-1}(pwd)) = h^n(pwd)

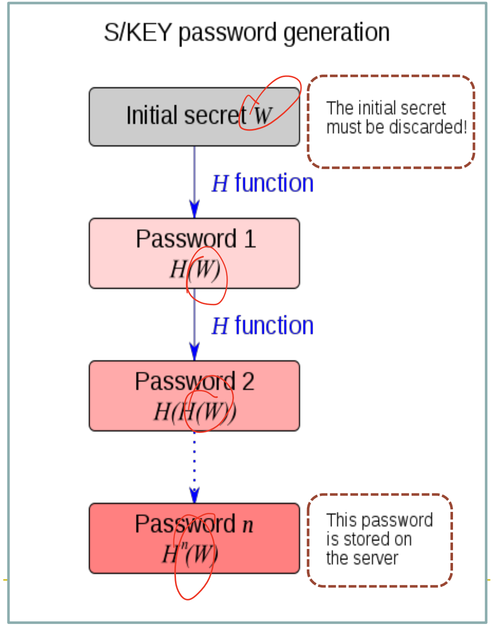
- If so accept the communication and update the hash value to …….h^{n-1}(pwd)

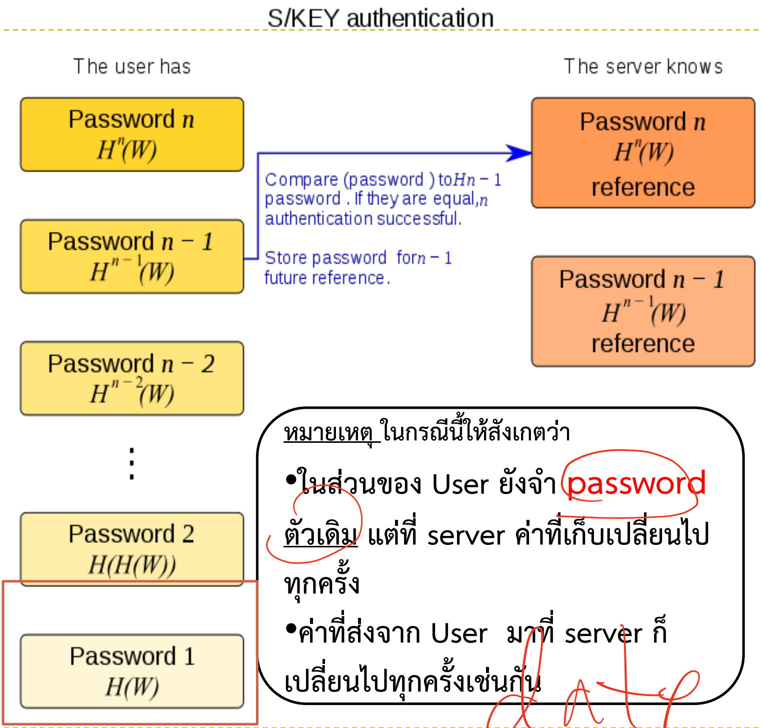
- OTP → ไม่ใช่โทรศัพท์

ลองไป Hash!

S/KEY หลักการทำงาน

เหมือน password ทั่วไป ต้องมีการ init เช่นกัน โดย server จะนำค่า ของ password ของ user ไป คำนวณ ตามนี้


S/KEY password generation
Initial secret W
The initial secret must be discarded!
H function
Password 1
H(W)
H function
Password 2
H(H(W))
Password n
H^n(W)
This password is stored on the server



<u>หมายเหตุ</u> ในกรณีนี้ให้สังเกตว่า

- ในส่วนของ User ยังจำ password <u>ตัวเดิม</u> แต่ที่ server ค่าที่เก็บเปลี่ยนไปทุกครั้ง
- ค่าที่ส่งจาก User มาที่ server ก็เปลี่ยนไปทุกครั้งเช่นกัน

date

http://en.wikipedia.org/wiki/S/KEY

Single-sign-on ใช้ google Authen ทุก login

- Single-sign on is similarly designed to reduce the volume of authentication information, in other words the number of passwords, that need to be remembered.
- What is the idea?
  - Sign in once
  - Access lots of resources.

- How does it work? (Very roughly)
  - Users are registered with multiple entities which share information.
  - Centralised authentication generates behind the scene tokens for passing authentication at other locations without explicit subsequent input by the user.
- What are the main issues?
  - The single-sign on has to be very well protected.
  - Scalability to work across multiple domains, multiple platforms and with multiple types of application authentication is tricky.

- Cyberos - 1st The Founder of single-sign-on

References:

- [1] Lecture slides prepared by Dr Lawrie Brown (UNSW@ADFA) for "Computer Security: Principles and Practice", 1/e, by William Stallings and Lawrie Brown, Chapter 1 "Overview".
- [2] CSCI262 Lecture Notes by Dr. Luke McEvan, University of Wollongong Australia.
- [3] Computer Security: Principles and Practice, W. Stalling and L. Brown, 1st edition, Pearson Education, 2008.
- [4] Security in Computing, C.P. Pfleeger and S.L. Pfleeger, 4th edition, Prentice Hall, 2007.
- [5] Computer Security, D. Gollman. 2nd edition, John Wiley & Sons, 2006.

แบบฝึกหัด: ข่อนี้ ออก

- ข้อ 1 สมมติว่า ใน กฎเกณฑ์ของ password คือ password ประกอบไปด้วย ตัวอักษร ตัวใดตัวหนึ่ง ใน a,b,c เรียงต่อกันไม่เกิน 4 ตัว

  คำถาม จำนวน password ที่เป็นไปได้ทั้งหมด มีกี่ตัว

- ข้อ 2 สมมติว่า ใน กฎเกณฑ์ของ password คือ password ประกอบไปด้วย ตัวอักษร ตัวใดตัวหนึ่ง ใน a,b,c เรียงต่อกันจำนวน 4 ตัว (ต้อง สี่ตัว)

  คำถาม จำนวน password ที่เป็นไปได้ทั้งหมด มีกี่ตัว

- ข้อ 3 สมมติว่า ใน กฎเกณฑ์ของ password คือ อนุญาตให้ใช้ตัวอักษรภาษาไทย เท่าที่มีบนแป้นพิมพ์ ตัวอักษรภาษาอังกฤษได้ทั้งตัวใหญ่และตัวเล็ก ความยาวอย่างน้อย 8 ตัว แต่ไม่เกิน 12 ตัว
  - ในกรณีเลวร้ายที่สุด แฮกเกอร์ต้องใช้เวลานานเท่าไหร่ ในการแฮกจนพบพาสเวิร์ดของมานี ถ้าในการทดสอบพาสเวิร์ด 1 ตัว แฮกเกอร์ใช้เวลา 1 microsecond.

ตอนที่ 2 [illegible]

password

user

bruteforce

hash

hash + salt