Lecture #5 – 9/12/2022

Ex 5: Due in 6 days

Resubmit #1 and #2 to the English boxes

Continuing authentication and authorization

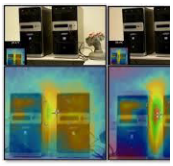
- Access control
  - Subject vs object
  - File permissions (R/W/X)
  - Bibas model – integrity
    - No read down
    - No write up
  - Bel lapedula – confidentiality
    - No write down
    - No read up
    - Classified material writing down to not classified info – prevents that issue

Now:

Breaking these models via covert – side channel attacks

Covert Channel

- MLS are designed to restrict legitimate channels of communication
- Other ways for information to flow
- **Covert channel**: a communication path not intended as such by system's designers
- For example, resources shared at different levels could be used to "signal" information



Air gapped computers

- Heat up one processer
- Induce electric current
- Heat = 1 , cool = 0 bit by bit
- Other computer gets the info

Need someone from the inside to pass the info

- Slow but doable

Pass – send short passwords etc…

Shared resources with different levels creates ways to pass information

Covert channels require an inside man

Covert Channel Example

- Alice has **TOP SECRET** clearance, Bob has **CONFIDENTIAL** clearance
- Suppose the file space shared by all users
- Alice creates file FileXYzW to signal "1" to Bob, and removes file to signal "0"
- Once per minute Bob lists the files
  - If file FileXYzW does not exist, Alice sent 0
  - If file FileXYzW exists, Alice sent 1
- Alice can leak **TOP SECRET** info to Bob!

Example:

- Info is that file either exists or not
- Monitor the folder to get the info
- Two insiders
- Alice – write permission
- Bob – read permission
- Both parties need to synchronize (timing)
- How to pass more information?
  - Multiple files
  - Change filenames

Idea is there are ways to break/crack the system and find ways to deliver information

Lecture #5 – 9/12/2022

Time isn't an issue in this case

How to eliminate covert channels

- Cooldown for files

Eliminate shared resources

- Price to pay: –…hurting communication and usability

Reduce covert channels by limiting capacity – for writing

Hard to remove entirely

Covert Channel

- Covert channels are everywhere
- "Easy" to eliminate covert channels:
  - Eliminate all shared resources…
  - …hurting communication and usability
- Virtually impossible to eliminate covert channels in any useful system
  - DoD guidelines: **reduce covert channel capacity** to no more than 1 bit/second
  - May be still not good enough to protect a key from leaking …

3 conditions

- Sender change
- Recieiver read them
- And need to synchronize

If we remove 1 of them then no covert channels

Add random noise to the channel

- Update mode of OS randomly
  - Then we get the processer heats up
- Erase random things from shared folders
  - Use your own folders, not shared ones

Removing covert channels

- Covert channel requires:
  - A sender can change
  - A receiver can read
  - They can synchronize
- Remove either one from the 3 requirements
- Add random noise to the channel
- Decrease bandwidth of time-based channel (no precise timers)
- Audit the channel

https://flic.kr/p/hwC4He

Audit – see if info is being passed

Compares between two strings

Check if password is correct

Related to sidechannel attacks – no one on the inside is needed

Execute attack just by exploring fns in specific examples

Physical attack on system

- Measure different computer metrics

Comparing string

Proper vs wrong password

Side channel attacks

- Help an **external** enemy to learn a secret
- Based on (physical) implementation weaknesses
  - Timing, power consumption, electromagnetic waves, cache attacks
- Prevention: reduce the information or the relation to the data
  - Bad for performance

```
static int strcmp2(p1, p2)
  const char *p1;
  const char *p2;
{
  const unsigned char *s1 = (const unsigned char *) p1;
  const unsigned char *s2 = (const unsigned char *) p2;
  unsigned char c1, c2;

  do
  {
     c1 = (unsigned char) *s1++;
     c2 = (unsigned char) *s2++;
     if (c1 == '\0')
        return c1 - c2;
  }
  while (c1 == c2);

  return c1 - c2;
}

void main(){
  ...
  return (0 == strcmp2(entered_pwd, correct_pwd))
}
```

Good Meltdown & Spectre lecture:
https://www.youtube.com/watch?v=UTHkYa3YQjA

Lecture #5 – 9/12/2022

Closer we are – stay in loop longer – longer to exit

More Wrong password will exit loop faster

Measure differences between times

First char wrong, exit – measure time and then see difference between time period for each character and build correct password

Joe Grand - https://www.youtube.com/watch?v=2-zQp26nbY8

Back to auth:

What do we use?

- Passwords
  - Hard to fake
  - Easy to remember

Methods:

- **Know = password or secret or answer to question**
- **Have = token**
  - **Key to door**
- **Are = biometrics**
  - **Fingerprint**
  - **Iris scan**
- **2FA**
  - **Atm card and pin**

Combine them

- What passwords have

Strong password

- Sentence
- Length
- Password manager
- Sequences which are random for each site

Passwords are hard, more accounts, etc.

What's your password …

Authentication

- Who are you? (Prove it!)
- Hard for others to fake
- Basis for deciding what you're allowed

Authentication methods

- Something you know
  - Password
- Something you have
  - Token
- Something you are
  - Biometrics
- Two factor authentication
  - ATM card and pin

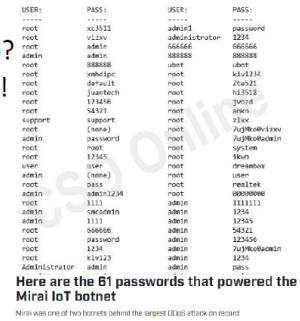
Passwords

- Most popular approach
- Free, easy to distribute and change
- Needs to be:
  - Easy to remember to be useful
  - Hard to guess (random) to be secure

Passwords practical problems

https://goo.gl/UFf9ba

- 8 letters and 256 characters give 2^{64}, no?
- NO, dictionary attacks!
- Try:
  - Common
  - Dictionary
  - All possible
- John The Ripper will automate this for you



USER:	PASS:	USER:	PASS:
root	xc3511	admin1	password
root	vizxv	administrator	1234
root	admin	666666	666666
admin	admin	888888	888888
root	888888	ubnt	ubnt
root	xmhdipc	root	klv1234
root	default	root	Zte521
root	juantech	root	hi3518
root	123456	root	jvbzd
root	54321	root	anko
support	support	root	zlxx.
root	(none)	root	7ujMko0vizxv
admin	password	root	7ujMko0admin
root	root	root	system
root	12345	root	1001
user	user	root	dreambox
admin	(none)	root	user
root	pass	root	realtek
admin	admin1234	root	00000000
root	1111	admin	1111111
admin	smcadmin	admin	1234
admin	1111	admin	12345
root	500606	admin	54321
root	password	admin	123456
root	1234	admin	7ujMko0admin
root	klv123	admin	1234
Administrator	admin	admin	pass

Here are the 61 passwords that powered the Mirai IoT botnet

Mirai was one of two botnets behind the largest DDoS attack on record

400,000 accounts

Common passwords – easy to crack and then go to dictionary attack

Then brute force

John the reaper – automated cracker via dictionary attack

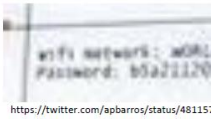
Cain – windows password recovery

World cup security system:

- Large and random passwords
  - So need to write it down
    - Was on the computer screen at the side
    - https://twitter.com/apbarros/status/481157619261116416

Passwords, good and bad

- Users will pick a weak password
- Generate a strong one and they will stick it to the monitor
- Force them to change it and they will use the old one
- And use the same one for many apps!
- And will share good ones with colleagues!



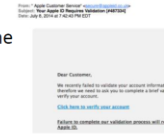
https://twitter.com/apbarros/status/481157

Weak password -> strong forced to write it -> change password (use old one)

And use for many accounts

Attacks on passwords

- One weak account is enough to penetrate the network
- Phishing attacks are effective
- Default passwords
- Key loggers
- Passwords reuse
- Social engineering



Email with login page – **phishing** – email from trusted company like a bank or apple

Real attack is MITM – to extract password – enter credentials …

Default passwords

Key loggers – record what is typed

Kevin Mitnick – pretended to be IT member of company -> asked people about their passcodes etc…

How to store and verify passwords

How to protect them

- Put them in file and limit access
- Dependent on 1 person

Solution: store hashes instead of the plain-text

- Generate hash of common passwords from dictionary and then compare
- Pre-computed hashes
    - One time automated job
- Which hash method do they use?
    - Can be easily accessed
        - We use specific methods
            - Not inventing new hashes
            - Takes time though

Passwords verification

Store the password in the clear?

- Protected by a superuser password …
- Single Point Of Failure!

Storing **hashes** instead

Attack: precomputed hashes of dictionary!

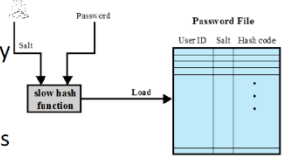
– One-time job automates the attack

Make life harder for attacker

- Use salt
    - Concatenate it to the passcode
        - Compute hash of password with salt and keep the hash
    - Random sequence of bits to password
    - 128 bit salt like in openbsd
        - Long and hard to crack but still doable
    - Another bonus
        - 2 people with same password will have different passwords via salt
            - Site stores hash with the salt for re-access
    - Use slow hash fns – procedure – more than 1 hash fn – takes time
        - So attacker can still guess but takes way longer

Adding a grain of salt

Choose random salt s and compute
$y = h(\text{password} \mid s)$
and store (s, y) in the password file
No offline dictionary attacks!
– 128 bits of salt in Open BSD
Salt hides duplicates
Slow hash function

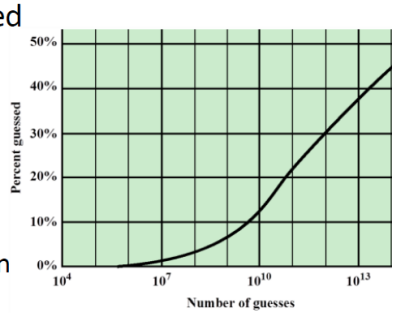


Password cracking

Huge "rainbow tables" (precomputed small salts)
- Win 2003: use 1.4G, break in 13sec

Using GPU (and the cloud)

Beyond the dictionary:
- Password generation algorithms
- Leaked passwords analysis



-
- https://kestas.kuliukas.com/RainbowTables/
- Password rules for students but still 10^13 combos crackers got about 0% of the passwords
- Windows 2003, password use 1/4 gb to crack it – broken in 13 seconds

Breaking passwords isn't so hard

- More sophisticated analysis of passwords
- Algorithm learns how people use and write passwords
    - Combos etc
- Patterns exist
- Then generate passwords based on constraints of what he learned

Preventing passwords attacks

- Lock-up or slow down after few failed password attempts
  - Better to cause inconvenience than to let guessing
- Non-computer interface prevents brute force
  - You can't automate it
  - (Make sure there's no API to do that)

Worst-Case Passcode Guessing Time (iPhone4)

Passcode Length	Complexity	Time
4	Numeric	18 minutes
4	Alphanumeric	51 hours

Nothing stops the attacker from continuing to guess passwords

- Guess via brute force via automated tool
- Prevent crackers from guessing passwords
    - Slow down processes to make it longer to guess
    - Close the system after X guesses
    - Inconvenience users < protects system

Phone passcodes – use every day without computers

- Physical

Or find a sophisticated way to do an automated attack

Biometric

- We are
- Face recognition is used often

Is it unique

- Hard to compare or artificial (hair color)

Accuracy vs cost for verification

Personal – unique

Cant change it but if someone steals it , you are stuck

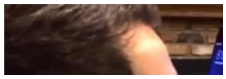
Passwords are quite accurate

Fingerprint depends on humidity

- Scanner type
- Type 1 vs type 2 arrows
  - https://www.scribbr.com/statistics/type-i-and-type-ii-errors/#:~:text=What%20are%20Type%20I%20and,hypothesis%20when%20it's%20actually%20false.
- Genuine vs non genuine user
- Via distributions, options exist for mistakes

Biometry challenges

- Spoofable
  - Need to verify it came from person
- Securely storing the measurement
- Can't change!



Link there to see

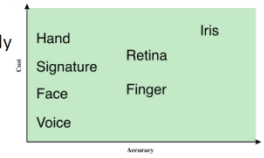
We have – tikens

- People use this for a long time:
  - Mother's voice
  - Passport picture
  - (Cats use this too ☺)
- You're your authenticator
  - Face recognition
  - Fingerprint
  - Form of palm
  - Retina/iris scan
  - Voice
  - Activity patterns: handwriting, typing, walk

Biometry requirements

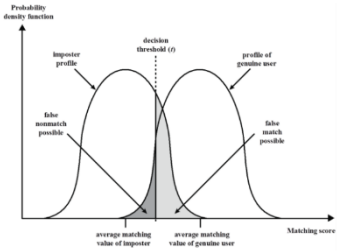
- Universal
  - Guitar player's fingerprints?
- Distinguishing
- Permanent
- Usable
  - Reliable & Robust
  - Collectable/User friendly
- Identification vs. verification

https://flic.kr/p/7zhv3Q



Biometry precision

- Unlike passwords: probability of matching
- Precision: detecting false positives and negatives



- Key is the classical token
- King's seal is another
- Passive - Magnetic cards
  - If it gets stolen?
    - Combine with PIN
  - Needs a reader
  - Can be reprogrammed

King achaz ben yotam

Keys reprogrammed and used

Microprocessors follow protocols

Smart card gets in system

Prevents us from typing passcode

RSA card with number, period changes every minute or so

Add sequence to passcode and another short pin # to combine it and then login

Now its google authenticator – OTP random # generator

Sequence of #s only client and server know the current sequence

- Each user gets specific set of sequences

Algo but can be reverse engineered

challenge response with token protocol

Client and server

Client connect to host get 2 hash fns and 1 random #

Now user uses token and generate from password a passcode

- Now user sends passcode not password
- After gets it from token, hashes it with 1 of hash fns from host (server)
- H – concatenates random # or random sequence to hash passcode
- And hash everything with f hash fn then sends it all to host
- Compares with db and if it matches proper user… gets clearance to system

Smart cards

- Embedded microprocessor
- Contact/contactless interface to the reader
- Authentication protocol
  - Static
  - Periodic password generator
  - Challenge-Response
- Can be reversed
- Relatively expensive

Challenge Response with Token

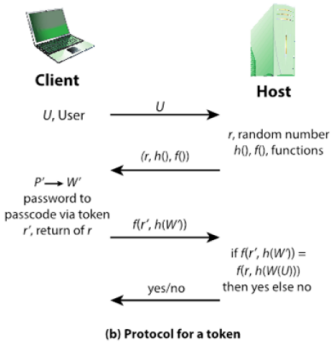
Protects from:
- Server-side breach
- Eavesdropping
- Replay

User sends identity

User password transformed by token – is never exposed

Hash never exposes the password to the server

Random (nonce) prevents replay attack



(b) Protocol for a token

Random # needed so each communication is unique and different – protect us from **replay attack**

- Attacker doesn't know which si real passcode or key but can record anything and use it in new session
- Host -> get clearance
- Random sequence, not gonna use same sequence twice
- New session each time

Eavesdropping via hash

Server side breaches – everything is hashed and using passcode not password

Hash never exposes passwords

Meltdown and spectre

- Relates to side channel attack
- Both relate to cache memory

Two videos on these attacks:

https://www.youtube.com/watch?v=I5mRwzVvFGE

https://www.youtube.com/watch?v=bs0xswK0eZk

cpu cache and main memory cache

amd doesn't speculate memory with meldown (intel and arm)

- Program read protected mem

spectre – all modern cpus

- Program read from any other program