AUTHENTICATION I: PASSWORDS & CO

Introduction to Computer and Network Security

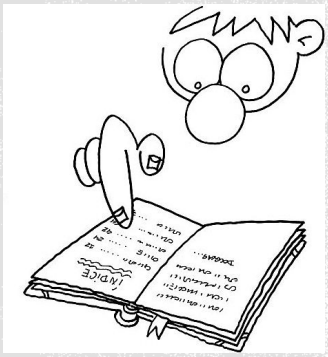
Silvio Ranise [silvio.ranise@unitn.it or *ranise@fbk.eu*]

UNIVERSITÀ DI TRENTO

FONDAZIONE BRUNO KESSLER

CONTENTS

- User authentication and digital identity lifecycle
  - Enrollment and identity assurance levels
- Authentication with passwords
  - History of passwords and lessons learnt
  - Passwords: attacks and mitigations
    - Protecting the password file: hashing & salting
- Multi Factor Authentication
- Password-less authentication
- Outsourcing identity management
  - Digital identity infrastructures



2

USER AUTHENTICATION & DIGITAL IDENTITY

Digital Identity definition (ISO/IEC 24760)

- Identity is a set of attributes related to an entity
- Attribute is a characteristic or property of an entity that can be used to describe its state, appearance, or other aspect

A person

An organization

Digital Identity is an identity whose attributes are stored and transmitted in digital form

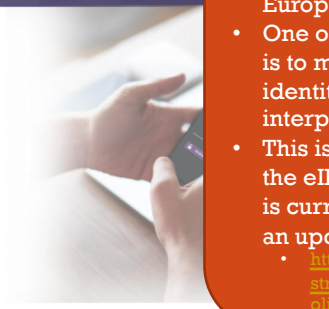
Very broad definition, independent of particular technology such as SAML, OpenID Connect, Blockchain, ...

PROMISES OF DIGITAL IDENTITY (1)

Unlocking the value of the digital economy globally

Today, digital identity is seen as an opportunity to drive **transformational change for citizens, businesses and public administrations**

The adoption of **mobile technology** for digital identity solutions across governments **support and enhance government performance and delivery and promote a more connected society both at national and cross-border level**



Enabled by trusted digital identity	$50bn	Global annual savings by 2020	»	$522bn Wider economic benefits by 2020
Through digital and technology transformation	$2.5bn	annual savings in 2014-15 in the UK		

- The theme of a unique digital marketplace is particular relevant in Europe
- One of the basic enabler is to make national digital identity system interporable
- This is one of the goals of the eIDAS framework that is currently undergoing an update
  - https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation

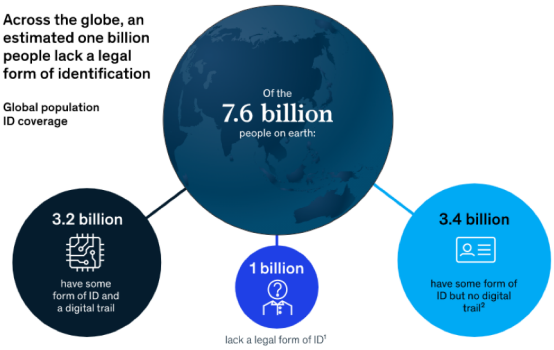
https://www.gsma.com/identity/wp-content/uploads/2016/02/MWCB16-Digital-Identity-as-a-Key-Enabler-for-eGovernment-Services-Marta-Ienco.pdf

4

PROMISES OF DIGITAL IDENTITY (2)

Across the globe, an estimated one billion people lack a legal form of identification

Global population ID coverage



[1]Legal ID coverage figures are based upon World Bank ID4D reporting of the latest registration levels for national ID, with voter registration used as a proxy where national ID does not exist or data are not available.
[2]Calculated as population with active social-media use, as reported in the *Global Digital Report 2018* from We Are Social. These social-media users are presumed to be within the population that has some form of legally recognized ID.

McKinsey & Company

S. Ranise - Security & Trust (FBK)

- A possible solution to the problem of providing 1 billion people in the world without a legal identity?
- What is the impact of not having a legal identity?
  - Difficulties in accessing basic services such as those of the public administration or financial/banking services

https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/digital-identification-a-key-to-inclusive-growth

5

DIGITAL IDENTITY LIFECYCLE

Relationship starts

Enrollment/on-boarding

The process through which an applicant applies to become a subscriber of an identity system and the identity system validates the applicant's identity.

The user receives a credential or authenticator from a *Credential Service Provider* (CSP).

Authentication

The process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to a system's resources.

Authorization/Access control

The process of checking user's permissions to access data, typically automated by evaluating a subject's attributes.

Relationship ends

Deregistration

DIGITAL IDENTITY LIFECYCLE

Relationship starts

Enrollment/on-boarding

The process through which an applicant applies to become a subscriber of an identity system and the identity system validates the applicant's identity.

The user receives a credential or authenticator from a *Credential Service Provider* (CSP).

Authentication

The process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to a system's resources.

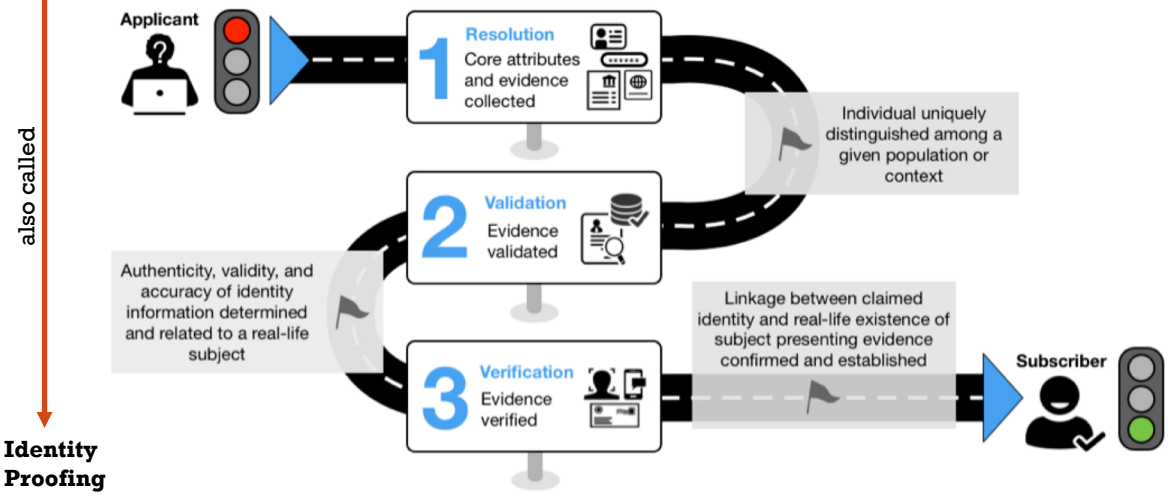
Authorization/Access control

The process of checking user's permissions to access data, typically automated by evaluating a subject's attributes.

Relationship ends

Deregistration

ENROLLMENT / ON-BOARDING (1)



(source NIST) https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63a.pdf

8

ENROLLMENT / ON-BOARDING (2)

- **Resolution**

Individual uniquely distinguished among a given population or context

  - Collection of Personal Identifiable Information (PII) from the applicant
    - Examples: name, address, date of birth, email, and phone number
  - Collection of two forms of identity evidence
    - Example: driver's license and passport by using the camera of a laptop

Minimum set of PII:
- Full name
- Date of birth
- Home address

- **Validation**

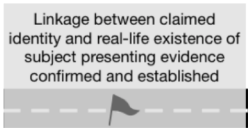
Authenticity, validity, and accuracy of identity information determined and related to a real-life subject

  - Validation of the information supplied above by checking an authoritative source
  - Check of the images of the license and the passport for alterations, coherence of the information on the documents and compliance against standard
  - Check with the issuing sources for the license and passport and validates the information matches

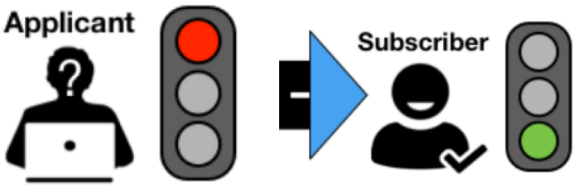
S. Ranise - Security & Trust (FBK)

9

ENROLLMENT / ON-BOARDING (3)



- **Verification**
  - Ask the applicant for a photo of themselves to match to the license and passport
  - Matching the pictures on the license and the passport to the applicant picture
  - Sending an enrollment code to the validated phone number of the applicant, the user provides the enrollment code, and check if they match, verifying the user is in possession and control of the validated phone number
  - The applicant has been successfully identity proofed



ENROLLMENT: IDENTITY ASSURANCE LEVELS

- Identity Assurance Levels (IALs)
  - A category that conveys the degree of confidence that the applicant's claimed identity is their real identity
- **IAL 1**: Attributes, if any, are self-asserted or should be treated as self-asserted
- **IAL 2**: Either remote or in-person identity proofing is required
- **IAL 3**: In-person identity proofing is required. Identifying attributes must be verified by an authorized respresentative through examination of physical documentation

"On the Internet, nobody knows you're a dog."

DIGITAL IDENTITY LIFECYCLE

Relationship starts

Enrollment/on-boarding

The process through which an applicant applies to become a subscriber of an identity system and the identity system validates the applicant's identity.

The user receives a credential or authenticator from a *Credential Service Provider* (CSP).

Authentication

The process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to a system's resources.

Authorization/Access control

The process of checking user's permissions to access data, typically automated by evaluating a subject's attributes.

Relationship ends

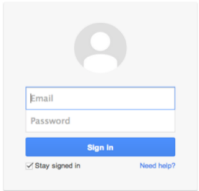
Deregistration

S. Ranise - Security & Trust (FBK)

12

Digital Identity Lifecycle
Authentication

- **Authentication** is the process of verifying the identity of a user, process, or device.
  - the **claimant** must demonstrate to the **verifier** that is indeed the one that claims to be
- When logging on, you enter
  - *username* to announce who you are
  - *password* to prove that you are who you claim to be



- *Password-based authentication* is widely accepted and not too difficult to implement…

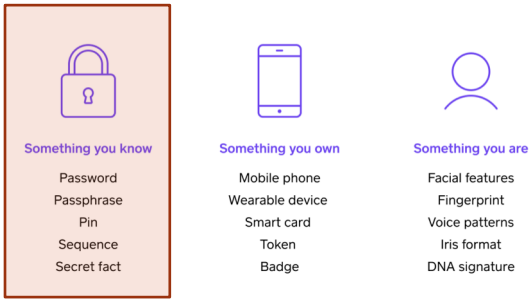
… still it is tricky to implement it in a secure way as we will see…
additionally passwords are well-known to **provide an insufficient level of protection**…

page 013

14

AN INTRODUCTION TO PASSWORDS

USER AUTHN



- When logging on to a computer you enter
  - **user name** and
  - **password**
- Entering user name:
  - You announce who you are
- Entering password:
  - You prove that you are who you claim to be
- This type of 'authentication' is called *user authentication*: the process of verifying a claimed user identity
- *Authentication by password* is widely accepted and not too difficult to implement (although it may be tricky)

AUTHN: ENROLLMENT

Similar problems occur when resetting passwords…

- **Passwords** should be ***secrets*** *shared between the user and the system*
- How do you bootstrap a system so that the password ends up in the right hands (better head), but nowhere else?
- *In a small enterprise*, users can collect their password personally
- *Otherwise*, the password could be sent by mail, email, or phone, or entered by the user on a web page or…
  - Who might intercept the message and who might actually pick it up?
    - Ex: a letter containing the password for an online bank account might be stolen or an impersonator may phone in asking for another user's password

PRACTICE VS RESEARCH DUALITY

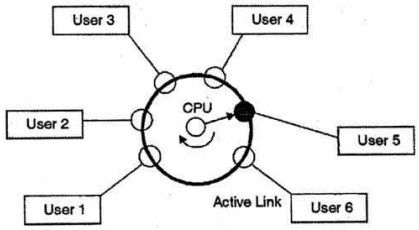
- Passwords dominated human computer authentication for half a century
- Despite this, rising consensus that a **more**
  - **secure** and
  - **user friendly**

  alternative should be found
- Duality between security
  - **researchers** focusing on authentication processes
  - **practitioners** aiming to protect user accounts and sensitive data

- A secure system might have to track the identities of the users requesting its services
  - The user **identity** is recorded when **logging** security relevant events in an audit trail
  - The user **identity** is a parameter in **access control** decisions
- **Authentication**: process of verifying a user's identity

Example
- researchers recommend strict password composition policies
- in practice, little evidence that such policies are effective in reducing security issues

A BIT OF HISTORY



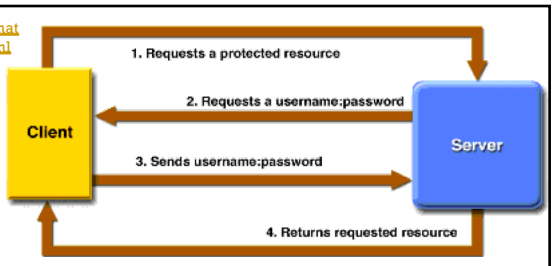
https://ecomputernotes.com/fundamental/disk-operating-system/time-sharing-operating-system

- Passwords originally used in the sixties with **time shared** mainframes
  - Quite different from today situation in the web
- Practices used today in the web are rooted in those of the original mainframes
  - Even though they are no more adequate to the current situation
  - Reasons: inertia and failure of research to come up with alternatives
- For instance, the following two models proposed by researchers are outdated
  - **Random user** drawing passwords uniformly and independently from a given set
  - **Offline attack against password files** that overemphasizes unthrottled guessing with respect to other threats including client malware, phishing, and network eavesdropping that are easier to mount

The use of these models have given rise to suggestions that are extremely difficult to put in practice by humans such as
- use long and randomly generated passwords without writing them down!

http://lia.deis.unibo.it/Courses/TecnologieWeb0708/materiale/laboratorio/guide/j2ee14tutorial7/Security5.html

CURRENT SITUATION



- In the context of **web authentication**
  - no single technology is likely to solve the authentication problem perfectly in all use cases
  - a synergistic combination of several different techniques is more likely to succeed and cope with the heterogeneous requirements arising in different use cases such as
    - social networks
    - financial services
    - healthcare services
    - …
- In practice, several different authentication processes use passwords in combination with other techniques including additional authentication factors (e.g., biometrics or One Time passwords) and Machine Learning for risk based authentication
  - we will see a hint on these later in this slide deck…

20

HISTORY OF PASSWORDS AND LESSONS LEARNT

ORIGINALLY A BAND AID

- In the sixties, passwords were originally added to time sharing operating systems to protect against jokes or **abuse of resources**
- Security issues were immediately reported
  - cases of guessing passwords
  - leak of the master password file stored in clear
- In the seventies, passwords were used to protect both resources and sensitive data
  - passwords were protected by using **hash** functions (more on this later)
  - afterwards, not only hashing but also **salting** was used (again more on this later)
  - For a historical overview, refer to the following paper
    https://rist.tech.cornell.edu/6431papers/MorrisThompson1979.pdf

21

MORRIS WORM

- A **computer worm** is a standalone malware program that replicates itself in order to spread to other computers
- It often uses a computer network to spread itself, relying on security failures on the target computer to access it

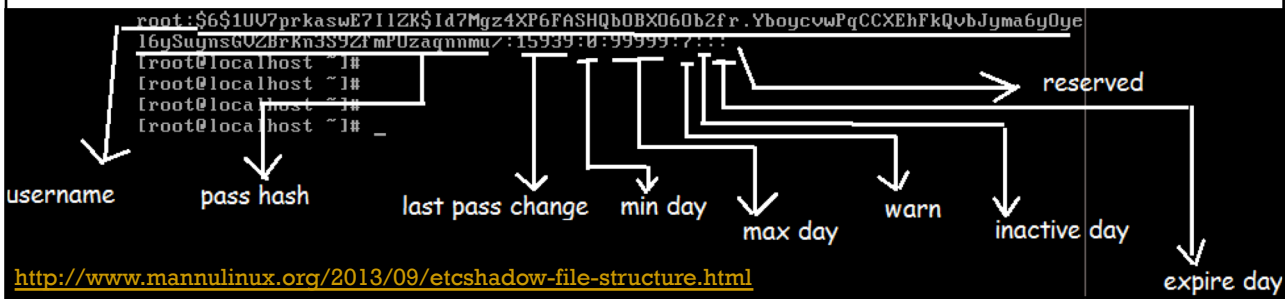
- In 1988, it was one of the first computer worms distributed via the Internet
- Among the vulnerabilities, it also **exploited easy to guess passwords** to spread
- By design, the worm was intended not to do any malicious activity but only to understand the possibility of writing such a program
- Unfortunately, **a coding error allowed the worm to infect multiple times the same computer thus potentially exhausting its resources**
  - In other words, the Morris worm was able to mount a Denial of Service attack
- As a result of the error, the **Internet was partitioned for several days**, as regional networks disconnected from the backbone and from each other to prevent recontamination whilst cleaning their own networks
- Morris was sentenced to pay a fine and some hours of community services
- For a detailed reconstruction, see the following technical report

https://spaf.cerias.purdue.edu/tech-reps/823.pdf

The U.S. General Accounting Office estimated that between $100,000 and $10 million was lost due to the internet inaccessibility that resulted from the attack.

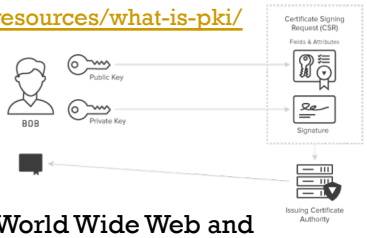
AFTER MORRIS WORM (1)

- In the first half of the nineties, passwords were stored in **shadow** files
- This means that instead of storing the (hashed and salted) passwords, they are moved to another file, called shadow, and making **this file readable only by those users who have access to the system root directory**
  - In other words, we are **protecting the shadow file** in which the passwords are stored **by using** another fundamental security mechanism called **access control** (we will see the details on the latter later in the course)
  - This is an example of using a combination of security mechanisms to protect a sensitive resource (the file containing the passwords) that provides for some redundancy in case of failure of one mechanism



http://www.mannulinux.org/2013/09/etcshadow-file-structure.html

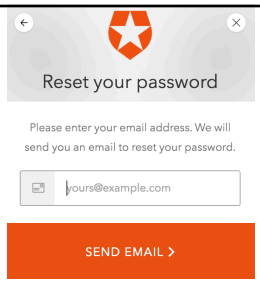
https://www.keyfactor.com/resources/what-is-pki/



AFTER MORRIS WORM (2)

- In the second half of the nineties, there was the advent of the World Wide Web and eCommerce
- This period has seen the first attempt to replace passwords with **certificates based on public key cryptography** (we will study this and other cryptographic techniques later in the course)
- This failed because **managing both client and server side certificates proved to be too cumbersome**
- Thus **certificates are used only by servers while users at the client side are typically asked to enter passwords or other credentials to authenticate at servers**
  - Here we are referring to the Transport Layer Security (TLS) that will also be discussed later in the course

24



PASSWORDS AND THE WEB

- Since the second half of the nineties, web services proliferated and usability problems for passwords arose that did not exist before
- **Resetting forgotten passwords**, previously done manually by system administrators of organizations, was automated **by email**… this created a security loophole for several reasons including
  - In many cases, users can click on a password reset link and get their old password delivered to their inbox. For this to work
    - the passwords are retrievable which means that they are either being stored in the site's database as plaintext or encrypted with a reversible algorithm
  - Additionally, passwords are sent without encryption so hackers sniffing network traffic could steal them
- The increased number of accounts per user stimulates the **reuse of the same password for multiple services** to help them memorize a reasonable number of passwords
- **Phishing** has emerged as a major concern with statistical approaches to mitigate it including blacklisting of known phishing site or machine learning classifiers to identify phishing messages such as those used by Google Mail

25

26

PASSWORDS: ATTACKS & MITIGATIONS

THREAT MODEL FOR CRACKING PASSWORDS

- **On-line**
  - It can be easily mitigated, it is sufficient to set a maximum number of attempts and then to block the authentication procedure for a certain interval of time to make brute-forcing practically impossible
  - This case is thus not very interesting…
- **Off-line**
  - Attackers is able to steal the file / database where passwords are stored
  - The attacker can then be able to carry a brute-force attack
  - It becomes crucial to adequately protect the passwords in storage…
  - … unfortunately, still today, there are several cases in which the protection is not implemented (i.e. passwords are stored in clear) or implemented in the wrong way (e.g., the hash functions used are weak or vulnerable to attacks)

27

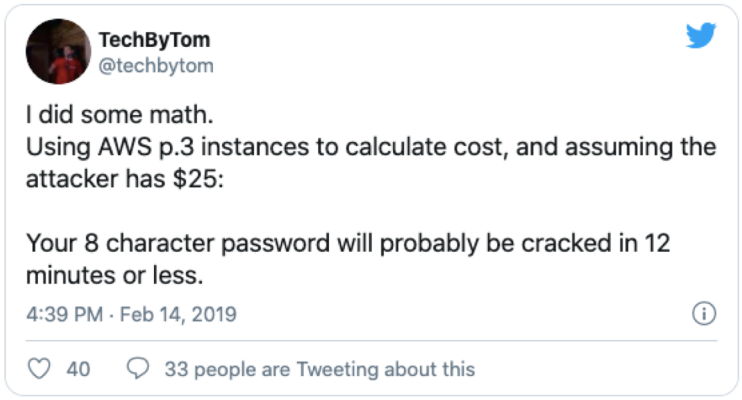
THREAT MODEL: FIRST ATTEMPT

- **On-line**
  - It can be easily mitigated, it is sufficient to set a maximum number of attempts and then to block the authentication procedure for a certain interval of time to make brute-forcing practically impossible
  - This case is thus not very interesting…
- **Off-line** we focus on this case
  - Attackers is able to steal the file / database where passwords are stored
  - The attacker can then be able to carry a brute-force attack
  - It becomes crucial to adequately protect the passwords in storage…
  - … unfortunately, still today, there are several cases in which the protection is not implemented (i.e. passwords are stored in clear) or implemented in the wrong way (e.g., the hash functions used are weak or vulnerable to attacks)

GUESSING PASSWORDS

- **Brute force** (exhaustive search): try all possible combinations of valid symbols up to a certain length
- The problem is that nowadays a lot of computational power is available at a reasonable cost
- This is so for several different reasons including
  - **Moore's law**: the number of transistors that could be housed in a dense integrated circuit doubles about every two years
    - this phenomenon suggests that **computational progress will become significantly faster, smaller, and more efficient over time**
  - possibility to **rent hw from cloud service providers** (e.g., Amazon)

Combination	Combinations per character	Example	Password Space
Lower case only	26	pass	456,976
Lower case only	26	password	208,827,064,576
Lower+upper case	52	Password	53,459,728,531,456
Lower+upper case + digits	62	Pa55w0rd	218,340,105,584,896
Lower+upper case + digits + symbols	92 (approx.)	Pa$$w0rd	5,132,188,731,375,616



30

https://turingpoint.de/en/blog/how-much-does-it-cost-to-crack-your-password/

#	Numbers	Lowercase	Mixed Case	Mixed Case + Numbers	Mixed Case + Numbers + Special
1	0 $	0 $	0 $	0 $	0 $
2	0 $	0 $	0 $	0 $	0 $
3	0 $	0 $	0 $	0 $	0 $
4	0 $	0 $	0 $	0 $	0 $
5	0 $	0 $	0 $	0 $	0 $
6	0 $	0 $	0 $	0 $	0 $
7	0 $	0 $	0 $	2 $	10 $
8	0 $	0 $	6 $	155 $	965 $
9	0 $	1 $	315 $	12,118 $	94,536 $
10	0 $	16 $	16,391 $	945,165 $	9.3 M$
11	0 $	416 $	852,312 $	73.7 M$	907.9 M$
12	0 $	10,820 $	44.3 M$	6 B$	89 B$
13	1 $	281,330 $	2 B$	449 B$	8.7 T$
14	11 $	7.3 M$	120 B$	34 T$	854 T$
15	113 $	190.2 M$	6.2 T$	-	-
16	1,134 $	5 B$	324 T$	-	-
17	11,339 $	129 B$	-	-	-
18	113,387 $	3.3 T$	-	-	-
19	1,1 M$	86 T$	-	-	-
20	11.3 M$	-	-	-	-

- monetary effort required to break the given password complexity with an **offline brute force attack by simple trial and error**
- expected costs an Amazon EC2 instance (p3.16xlarge - 8xNVIDIA Tesla V100 GPUs) and SHA256 hash algorithm

31

GUESSING PASSWORDS RELOADED

- Brute force (exhaustive search): try all possible combinations of valid symbols up to a certain length
- **Being smarter**: search through a restricted name space
  - Ex: passwords associated with a user like name, names of friends and relatives, car brand, car registration number, phone number,…, or try popular passwords
  - Typical example: **dictionary attack**, i.e. 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

32

DICTIONARY: AN EXAMPLE

- Trying a large number of
  1. commonly used names as possible account names and then
  2. try a large number of commonly used passwords for the account
- Commonly used account names
  - root, webmaster, admin, mysql, oracle, guest, test, sales, staff, …
- A password file embedded in the Mirai botnet (see right)
  - 61 pairs (name, password)

https://www.csoonline.com/article/3126924/here-are-the-61-passwords-that-powered-the-mirai-iot-botnet.html

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	ikwb
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	666666	admin	54321
root	password	admin	123456
root	1234	admin	7ujMko0admin
root	klv123	admin	1234
Administrator	admin	admin	pass
service	service	admin	meinsm
supervisor	supervisor	tech	tech
guest	guest	mother	fucker
guest	12345		
guest	12345		

WHY DICTIONARY ATTACKS ARE SO EFFECTIVE?

- Prior to 2010, guessing passwords was not very relevant as a threat
- In 2009, a large site, rockyou.com, with more than 32 million users, got hacked and the website stored all its users' passwords in **plain text** and all the passwords were leaked
  - This gave hackers the single biggest insight into how humans construct passwords globally
- Every password leak that has happened since just widens hackers knowledge into how people create passwords
  - There are dozens of major password leaks every year
  - For instance, in 2016 there was the LinkedIn breach: 96 percent of passwords were cracked over the course of a week!
- Compiling RockYou, LinkedIn, Ashley Madison, and other big leaks, one gets a really detailed picture of people's passwords.
- It's a **feedback loop**; passwords are cracked, added to word lists, and used to crack new passwords

CRACKING PASSWORDS IN PRACTICE (1)

- Popular password crackers: https://hashcat.net/hashcat/ and https://www.openwall.com/john/
- An example with hashcat
- Target: MD5 hashes
- Create a file containing digests of passwords
  - echo -n "Password" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "HELLO" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "MYSECRET" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "Test1234" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "P455w0rd" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "GuessMe" | md5sum | tr -d " -" >> target_hashes.txt
  - echo -n "S3CuReP455Word" | md5sum | tr -d " -" >> target_hashes.txt

CRACKING PASSWORDS IN PRACTICE (2)

- An example with hashcat (contd)
- The content of the file is the following
  1. dc647eb65e6711e155375218212b3964
  2. eb61eead90e3b899c6bcbe27ac581660
  3. 958152288f2d2303ae045cffc43a02cd
  4. 2c9341ca4cf3d87b9e4eb905d6a3ec45
  5. 75b71aa6842e450f12aca00fdf54c51d
  6. 031cbcccd3ba6bd4d1556330995b8d08
  7. b5af0b804ff7238bce48adef1e0c213f
- Select one of the available wordlists…
  - For instance: rockyou.txt
  - Available at, e.g., https://www.kaggle.com/datasets/wjburns/common-password-list-rockyoutxt

123456
12345
12345678
9
password
iloveyou
princess
1234567
rockyou
12345678
abc123
nicole
daniel
babygirl
monkey
lovely
jessica
654321
michael
ashley
qwerty
111111
iloveu
…

CRACKING PASSWORDS IN PRACTICE (3)

- An example with hashcat (contd)
- Invoke the tool: hashcat -m 0 -a 0 -o cracked.txt target_hashes.txt /usr/share/wordlists/rockyou.txt
  - -m 0 designates the type of hash we are cracking (MD5)
  - -a 0 designates a dictionary attack
  - -o cracked.txt is the output file for the cracked passwords
  - target_hashes.txt is our input file of hashes
  - /usr/share/wordlists/rockyou.txt is the absolute path to the wordlist file for this dictionary attack

Results

dc647eb65e6711e155375218212b3964:**Password**
eb61eead90e3b899c6bcbe27ac581660:**HELLO**
75b71aa6842e450f12aca00fdf54c51d:**P455w0rd**
2c9341ca4cf3d87b9e4eb905d6a3ec45:**Test1234**
958152288f2d2303ae045cffc43a02cd:**MYSECRET**

5 out of 7 passwords are quickly cracked (few seconds on standard machine)

37

MITIGATIONS

- Change default passwords!
  - Often passwords for system accounts have a default value (e.g., admin)
- Avoid guessable passwords:
  - Prescribe a minimal password length
  - Password format: mix upper and lower case, include numerical and other non-alphabetical symbols
  - Today **on-line dictionaries** for almost every language exist

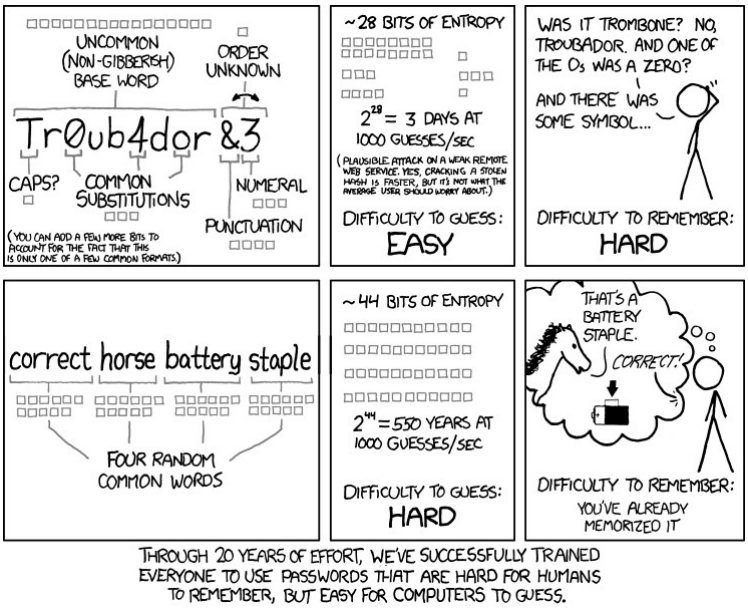
38

MITIGATIONS

- Change default passwords!
  - Often passwords for system accounts have a default value (e.g., admin)
- Avoid guessable passwords:
  - Prescribe a minimal password length
  - Password forma… and … n-alphabetical symb…
  - Today on-line d… …or aln…

OUTDATED!!!

39



Old days

Future…

40

A-HED

The Man Who Wrote Those Password Rules Has a New Tip: N3v$r M1^d!

Bill Burr's 2003 report recommended using numbers, obscure characters and capital letters and updating regularly—he regrets the error

By *Robert McMillan*

Aug. 7, 2017 12:41 p.m. ET

The man who wrote the book on password management has a confession to make: He blew it.

41

NEW NIST PASSWORD LIMITATIONS

- Forbid commonly used passwords
- Don't use password hints or knowledge-based authentication
- Limit the number of password attempts

42

PASSWORD GENERATORS & MANAGERS

- Websites and apps offer to create randomly generated password
  - Based on **pseudo random** number **generators** (e.g., linear congruential generators) to create a random string of symbols often of **length between 10 and 16** characters
- If only generators are used, then burden of remembering passwords is on user
  - Notice that generated passwords are **typically longer than 8 characters** because it is becoming cheaper and cheaper to brute passwords of up to that length…
  - … this makes the task of remembering them even more difficult
- If managers are used, then they may be vulnerable to attacks especially if they are on-line

43

PROTECTING THE PASSWORD FILE

- Operating system maintains a file with user names and passwords
- Attacker could try to compromise the confidentiality / integrity of password file
- Options for protecting the password file:
  - cryptographic protection
  - access control enforced by the operating system
  - combination of cryptographic protection and access control, possibly with further measures to slow down dictionary attacks

PROTECTING THE PASSWORD FILE: HASHING & SALTING

Secure storage of login and password

HASH: 1-WAY FUNCTION

- Cryptographic hash functions
- A 1-way function *f* is a function that is **relatively easy to compute but hard to reverse**, i.e.
  - 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 entering a password *x'*, the system applies the one-way function *f* and compares *f*(*x'*) with the expected value *f*(*x*)

46

WHY HASHING AND NOT ENCRYPTION?

- **Hashing is a one-way function** (i.e., it is impossible to "decrypt" a hash and obtain the original plaintext value)
  - Hashing is appropriate for password validation
  - Even if attackers obtain the hashed password, they cannot enter it into an application's password field and log in as the victim
- **Encryption is a two-way function**, meaning that the original plaintext can be retrieved
  - Encryption is appropriate for storing data such as a user's address since this data is displayed in plaintext on the user's profile. Hashing their address would result in a garbled mess.

47

HASH FUNCTION

- Requirements on a hash function h
  - Ease of computation: given x, it is easy to compute $h(x)$
  - Compression: h maps inputs x of arbitrary bit-length to outputs $h(x)$ of a **fixed bit-length** n
  - One-way: given a value y, it is computationally infeasible to find an input x so that $h(x) = y$
  - Weak collision resistance: given an input x and $h(x)$, it is computationally infeasible to find another input x', $x \neq x'$, with $h(x) = h(x')$
  - Strong collision resistance: it is computationally infeasible to find any two inputs x and x', $x \neq x'$, with $h(x) = h(x')$

Collision = the situation in which two inputs x and x' map to the same hash, i.e. $h(x) = h(x')$

48

HASHING IN PRACTICE

- Just changing one single letter in the input string makes the resulting hashes look very different
- This is called the **avalanche effect**, i.e. small changes to the input result in significantly different output
- This feature makes finding the input given the output much harder as the output looks "random"

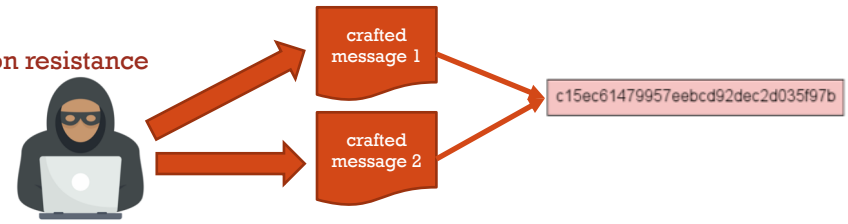
Here I use *python3*

- >>> import hashlib
- >>> str1 = "**homely**"
- >>> hash1 = hashlib.sha256(str1.encode('utf-8'))
- >>> hash1.hexdigest()
  - '**423cb2dd7078b1dac9d67c89b5bb7292f263041abf60455a443664095b47f7b4**'
- >>> str2 = "**comely**"
- >>> hash2 = hashlib.**sha256**(str2.encode('utf-8'))
- >>> hash2.hexdigest()
  - '**6f804dfb2cd3a13c1cc93f6dfea34ea436dc6da85b164697a3ec1e4f135c4285**'

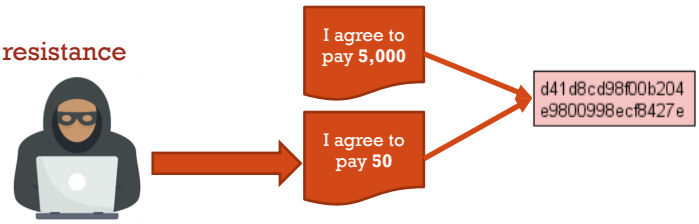
49

ON COLLISION RESISTANCE

- Strong collision resistance



- Weak collision resistance



50

HASH FUNCTIONS & COLLISIONS

hashing is the mathematical version of briskly stirring a pot containing 0s and 1s

- Theoretically, it is indeed **possible** to find **collisions** for any given hash function
  - This is obvious as soon as we consider the cardinality of the input set and the output set of a hash function where the former contains potentially unbounded messages and the latter only messages of size *2^n* for a given number *n* of bits
- The point that makes hash function useful in practice is that the **likelihood** that such a **collision** can **happen** should be **negligible**
  - Example
    - when n=128, we have 2^128 = 3.4 * 10^38 possible distinct hashes
    - **under the assumption that the hash function distributes uniformly the outputs among all possible distinct hashes**, it is more likely that the same person repeatedly wins the lottery…
    - For the assumption above to be satisfied, the design of the hash function should be performed in the appropriate way so that its output looks random…
    - … this is not always trivial and it was the source of attacks to widely adopted hash functions in the past…

51

HASH FUNCTION IMPLEMENTATIONS

- MD4: weak, it is computationally feasible to find meaningful collisions
- MD5: standard choice in Internet protocols, now broken and no longer recommended
- Secure Hash Algorithm (SHA-1): designed to operate with the US Digital Signature Standard (DSA); 160-bit hash value; collision attacks reported
- RIPEMD-160: hash function frequently used by European cryptographic service providers
- SHA-256: when longer hash values are advisable

52

A COLLISION ON MD5

Violation of strong collision resistance

```
d131dd02c5e6eec4 693d9a0698aff95c 2fcab58712467eab 4004583eb8fb7f89
55ad340609f4b302 83e4888325 71415a 085125e8f7cdc99f d91dbdf280373c5b
d8823e3156348f5b ae6dacd436c919c6 dd53e2b487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080a80d1e c69821bcb6a88393 96f9652b6ff72a70
```

```
d131dd02c5e6eec4 693d9a0698aff95c 2fcab50712467eab 4004583eb8fb7f89
55ad340609f4b302 83e4888325 f1415a 085125e8f7cdc99f d91dbd7280373c5b
d8823e3156348f5b ae6dacd436c919c6 dd53e23487da03fd 02396306d248cda0
e99f33420f577ee8 ce54b67080280d1e c69821bcb6a88393 96f965ab6ff72a70
```

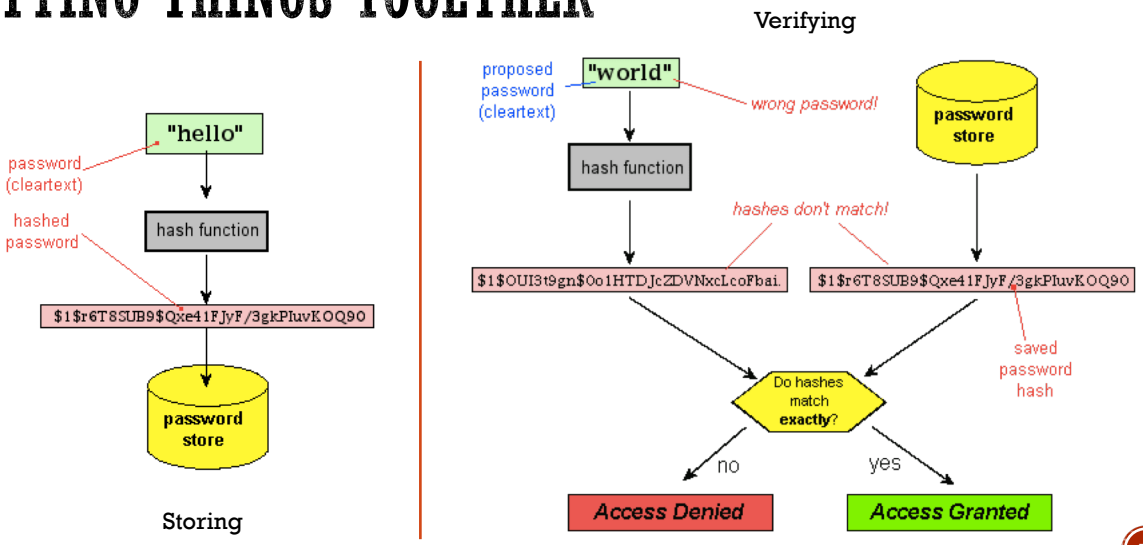
```
79054025255fb1a26e4bc422aef54eb4
```

Two 128 bytes block yielding the same MD5 digest differing for only 6 bits

https://en.wikipedia.org/wiki/MD5

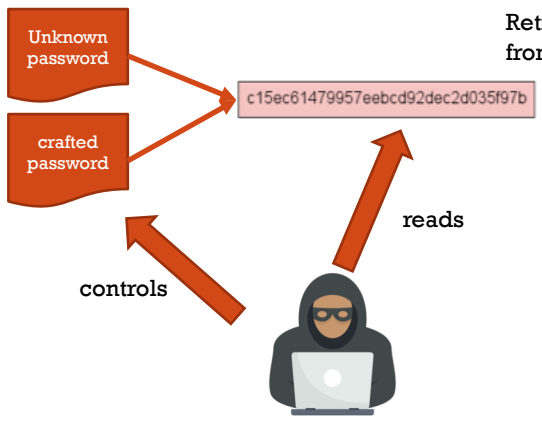
53

PROTECTING PASSWORD FILES PUTTING THINGS TOGETHER



54

VIOLATION OF ONE WAY PROPERTY



Retrieving a password from a password file

Notice that it is not strictly necessary to guess the password, it is "sufficient" to come up with a string yielding the same digest to authenticate as the user…

55

PROBLEMS WITH HASHED PASSWORDS

- Dictionary attacks
  - Use one of the many available on-line
  - Notice that hashes can be **pre-computed** and stored in databases whose key is the hash
  - Huge amount of storage space that is becoming cheaper (because of cloud services)
  - Effective when a large number of passwords are to be cracked
    - Pre-computed hashes of the content of the dictionary needs be generated only once
    - When completed, password hashes can be looked up almost instantly to find the password

- Rainbow tables
  - Data structure that aims to reduce storage requirements at the cost of slightly longer lookup-times

Have a look at the following site
https://hashtoolkit.com/
for an example of a searchable database of hash digests…

56

SALTING

Salt = fixed-length random value

- We refer to $f(x)$ as the hashed password, i.e. to the hash/digest of x

- To slow down dictionary attacks, a salt is appended to the password before hashing and stored with the hashed password

  - If two users have the same password, they will now have different entries in the file of hashed passwords

  - Ex: Unix uses a 12 bit salt

57

SALTING IN PRACTICE

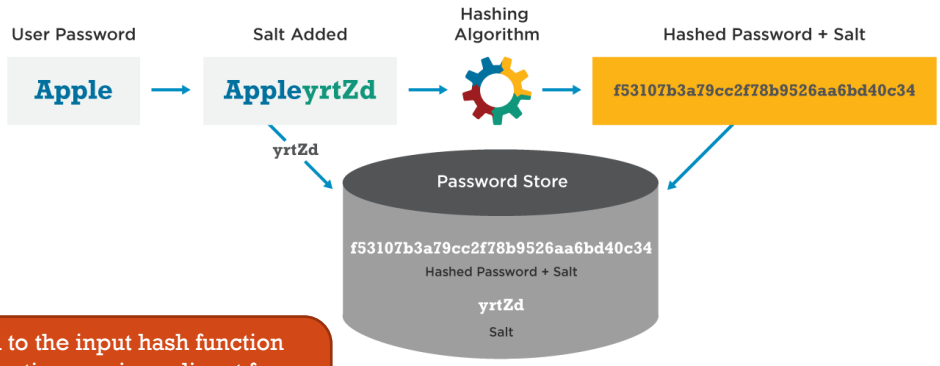
- A UUID (Universal Unique Identifier) is a 128-bit number used to uniquely identify some object or entity on the Internet
- uuid4() creates a random UUID

Again I use *python3*

- >>> import hashlib, uuid
- >>> str = "Tyger Tyger, burning bright, In the forests of the night; What immortal hand or eye, Could frame thy fearful symmetry?"
- >>> salt = uuid.uuid4()
- >>> salt
  - UUID('f4c7d2be-59d6-45ea-a34b-81e02c88ebbb')
- >>> salt.hex
  - 'f4c7d2be59d645eaa34b81e02c88ebbb'
- salted_hash = hashlib.**sha512**(str.encode('utf-8') + salt.hex.encode('utf-8'))
- >>> salted_hash.hexdigest()
  - '96d3e17abc98249c75daadf8ffd96cb4b0719b95fd2e328c151c215b0394b750d195790f068e97a7983516d75318ba70870f82b376f15954699ff0fe9ade2b3c'

58

Password Hash Salting



A salt added to the input hash function allows for creating a unique digest for every password (regardless of the input not being unique as in the case of two users with the same password)

https://cyberhoot.com/cybrary/password-salting/

59

SALTING AS A MITIGATION TO DICTIONARY ATTACKS

- When the salt is unique for each hash, an attacker must re-compute a rainbow table for each user hash
- This greatly reduces the capability of an attacker to crack a lot of passwords as it was the case without salting
- Notice that an attacker may still be able to crack a single password once he or she knows the salt that is not necessary to be kept secret
  - For simplifying the verification of hashes, salts are usually stored in clear with the user name and hash of the password
- Typical good sizes of salts are 32 or 64 bytes (notice the increase in the length of the password)
- **Final advice**: avoid to implement your own version of hashing and salting but use well—engineered available functions such as bcrypt, scrypt or even better argonid

HASH FUNCTIONS FOR PASSWORDS (1)

- The longer it takes to compute a digest, the longer a brute force attack will take
- The **work factor** is the number of iterations of the hashing algorithm that are performed for each password (usually, 2^work iterations).
  - The **purpose** of the work factor is to **make calculating the hash more computationally expensive**, which in turn reduces the speed and/or increases the cost for which an attacker can attempt to crack the password hash
- When choosing a work factor, a **balance** needs to be struck **between security and performance**
  - Higher work factors will make the hashes more difficult for an attacker to crack but will also make the process of verifying a login attempt slower.
  - If the work factor is too high, this may degrade the performance of the application and could also be used by an attacker to carry out a denial of service attack by making a large number of login attempts to exhaust the server's CPU.
- There is no golden rule for the ideal work factor; determining the optimal work factor will require experimentation on the specific server(s) used by the application
  - As a **general rule**, calculating a digest should take **less than one second**

HASH FUNCTIONS FOR PASSWORDS (2)

- There are a number of modern hashing algorithms that have been specifically designed for securely storing passwords
  - This means that they should be slow (unlike algorithms such as MD5 and SHA-1, designed to be fast), and how slow they are can be configured by changing the work factor
- 4 main algorithms
  - **Argon2** is the winner of the 2015 **Password Hashing Competition** (https://www.password-hashing.net/) There are three different versions of the algorithm, and the **Argon2id** variant should be used, as it provides **resistance to GPU-based attacks**
  - **scrypt** is a *password-based key derivation function*. While new systems should consider Argon2id for password hashing, scrypt should be configured properly when **used in legacy systems**
  - The **bcrypt** password hashing function should be the second choice for password storage if Argon2id is not available or PBKDF2 is required to achieve FIPS-140 compliance
  - **PBKF2** is a *password-based key derivation function*, with a sliding computational cost; it requires the selection of an internal hashing algorithm

- **pseudorandom function**: an efficiently-computable functions which emulate a random oracle

- cryptographic algorithm that derives one or more secret keys from a secret value such as a main key, a password, or a passphrase using a **pseudorandom function**

YET ANOTHER ATTACK: CREDENTIAL STUFFING

- Attackers take millions or even billions of email addresses and corresponding cracked passwords from compromised databases and see how many of them work at other online services

- Mitigation
  - Company first extracts from leaked credential lists any email addresses that correspond to their current user base
  - Feed passwords according to standard verification process (hashing & salting) and obtain hash code C
  - Compare stored hash code with C: if they are different, then do nothing
  - If they are equal, then force password reset procedure for the user
  - More info at https://krebsonsecurity.com/2019/08/forced-password-reset-check-your-assumptions/
  - See also at https://www.troyhunt.com/ive-just-launched-pwned-passwords-version-2/ for a list of half a billion passwords to download

THREAT MODEL: A REFINEMENT

- On-line
- Off-line

- **Phishing**
  - Identification and authentication through username and password provide **unilateral authentication**
  - Computer verifies the user's identity but the user has no guarantees about the identity of the party that is going to receive the password
  - A user voluntarily sends the password over a channel, but is misled about the end point of the channel

64

PHISHING: SOME REMARKS

Passwords are the root cause of > 80% of data breaches

- Attacker impersonates the system to trick a user into releasing the password to the attacker
  - Ex: a message could claim to come from a service you are using, tell you about an upgrade of the security procedures, and ask to enter username and password at the new security site that will offer stronger protection
- Take care to enter your passwords only at the "right" site (but how do you know?)
- **Social engineering**
  - Attacker impersonates the user to trick a system operator into releasing the password to the attacker

Phishing is the main cause of password leakage!!!

How can we mitigate this type of attacks*?*

Educate / train users and increase their awareness…

This is not enough…

65

66

EXTENSIONS OF PASSWORD BASED AUTHENTICATION

An attempt to mitigate phishing attacks…

Authentication
Multi-factor authentication

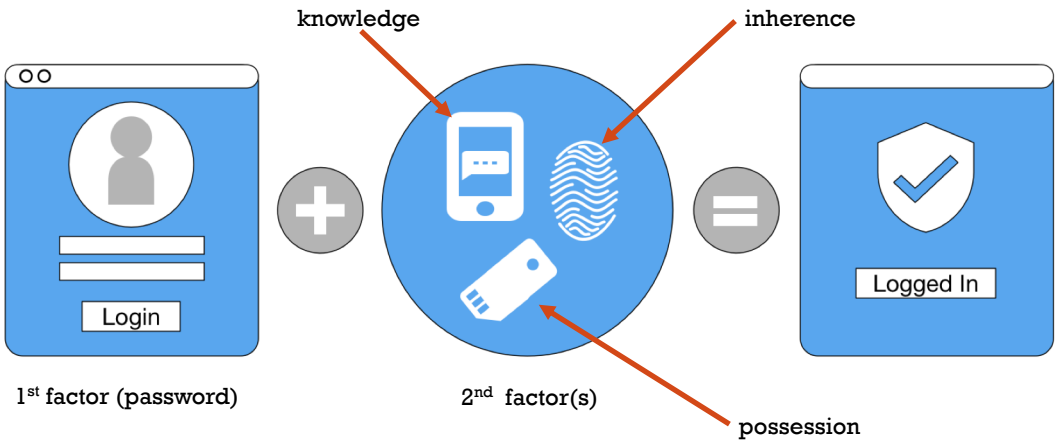
A procedure based on the use of two or more of the following factors:

knowledge, something only the user knows, e.g., static password, personal identification number;

ownership, something only the user possesses, e.g., token, smart card, mobile phone; and

inherence, something the user is, e.g., biometric characteristic, such as a fingerprint.

MULTI FACTOR AUTHENTICATION (MFA)



68

MFA: A (COMMON) SCENARIO

69

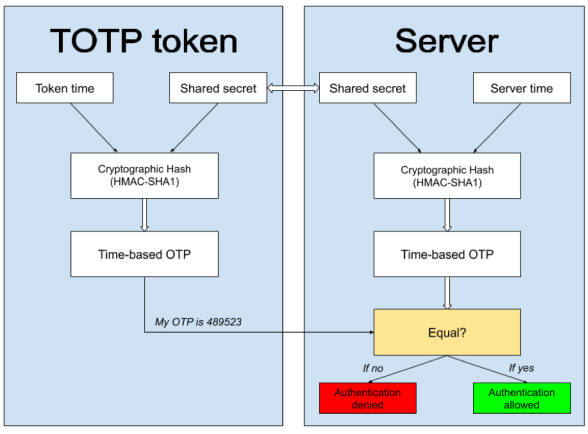
MULTI FACTOR AUTHENTICATION (MFA)

- MFA = authentication method in which a computer user is granted access only after successfully presenting **two or more factors** to an authentication mechanism:
  - knowledge (something the user and only the user knows)
  - possession (something the user and only the user has)
  - inherence (something the user and only the user is)
- Two-factor authentication (2FA) is a type of MFA confirming users' claimed identities by using a combination of two different factors
  - Example 1: withdrawing of money from an ATM; only the correct **combination of a bank card** (something the **user possesses**) and a **PIN** (something the **user knows**) allows the transaction to be carried out
  - Example 2: **supplement** a user-controlled **password with** a **One-Time Password** (**OTP**) or code generated or received by an authenticator (e.g., a security token or **smartphone**) that only the **user possesses**

70

https://datatracker.ietf.org/doc/html/rfc6238

TIME BASED ONE TIME PASSWORD



https://protectimus.medium.com/totp-algorithm-explained-ee4934b91ee3

71

TIME BASED ONE TIME PASSWORD (TOTP)

- Parameters
  - T0: Unix time from which to start counting time steps (default is 0)
  - Tx: interval which will be used to calculate the value of the counter (default is 30 seconds)
- Algorithm idea
  - Authenticator and authenticatee compute the TOTP value then the authenticator checks if the TOTP value supplied by the authenticated matches the locally-generated TOTP value
  - Some authenticators allow values that should have been generated before or after the current time in order to account for slight clock skews, network latency and user delays
- Weaknesses
  - **OTP values** can be **phished** just as passwords can and **replayed**, though they require attackers to proxy the credentials in real time rather than collect them later
  - **An attacker who steals the shared secret can generate new, valid TOTP values at will**. This can be a particular problem if the attacker breaches a large authentication database; this was the case the in the past…

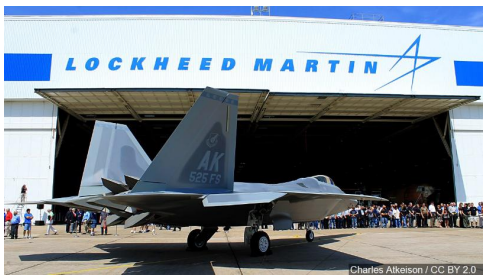
72

ATTACK TO TOTP: LOCKHEED MARTIN

- RSA was hacked in 2007
- **Secret key for OTP tokens stolen**

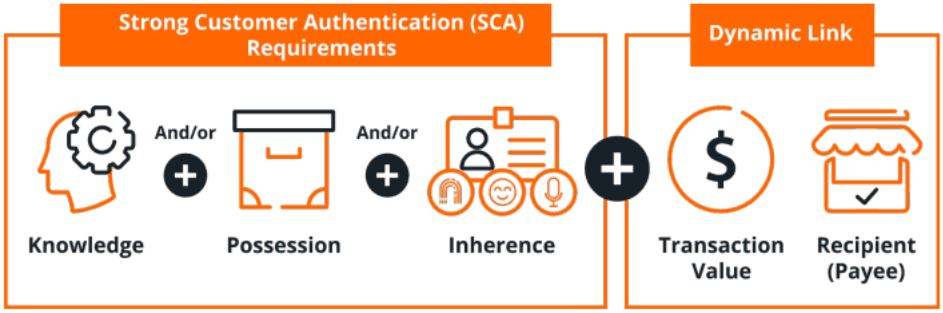
- Hackers could generate OTP and spoof users
- Companies using RSA SecureID were vulnerable
- Lockheed Martin used RSA SecureID

- **Chinese attackers spoofed Lockheed Martin staff–Stole plans for F-35 fighter jet**



73

BEYOND TOTP: REVISED PAYMENT SERVICE DIRECTIVE (PSD2)



74

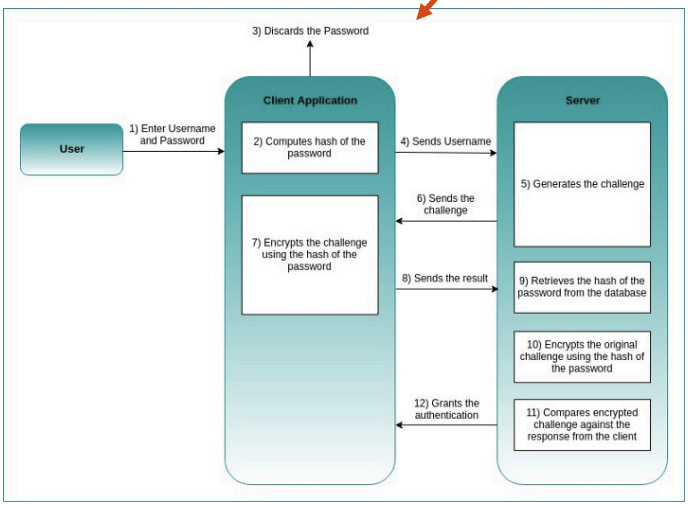
EFFECT OF PSD2: NO MORE TOTP HW TOKEN!

Can you explain why?
Do you see a way to overcome the problem?

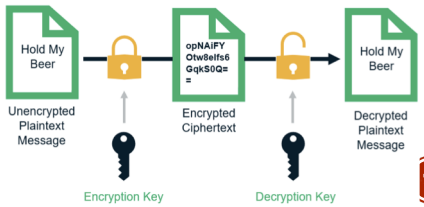
75

Password in clear never leaves the client!

CHALLENGE RESPONSE PROTOCOLS (1)



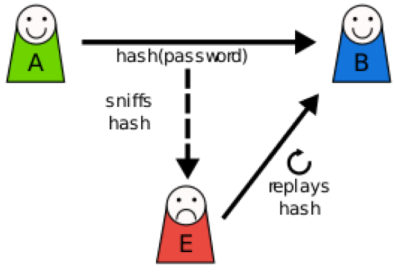
- Encryption convers the original representation of the information (**plaintext**) into an alternative form (**ciphertext**)
- Only authorized parties can decipher a ciphertext back to plaintext and access the original information
- Encryption denies the intelligible content to a would-be interceptor



76

CHALLENGE RESPONSE PROTOCOLS (2)

- Offer **protection** from **replay attacks**
  - Suppose Alice wants to prove her identity to Bob
  - Bob requests her password as proof of identity, which Alice dutifully provides (possibly after some transformation like hashing, or even salting, the password
  - Meanwhile, Eve is eavesdropping on the conversation and keeps the password (or the hash)
  - After the interchange is over, Eve (acting as Alice) connects to Bob; when asked for proof of identity, Eve sends Alice's password (or hash) read from the last session which Bob accepts, thus granting Eve access



Since challenge changes at every connection, Eve cannot reuse previously intercepted hashed passwords…

77

BACK TO PSD2 COMPLIANT OTP

- Idea: include in the challenge the data that are unique to the particular transaction such as
  - to identifier of the destination of the transaction
  - the amount of the transaction
  - the instant in which the transaction has been initiated
  - …
- This is not enough… why?
- It is necessary to add some random data, i.e. some unpredictable string to the challenge…

78

BACK TO PSD2 COMPLIANT OTP

- Idea: include in the challenge the data that are unique to the particular transaction such as
  - to identifier of the destination of the transaction
  - the amount of the transaction
  - the instant in which the transaction has been initiated
  - …
- This is not enough… why?
  - **Answer**: because the data above can be inferred from contextual information and makes the challenge too predictable…
- It is necessary to add some random data, i.e. some unpredictable string to the challenge…

79

USE OF SMARTPHONES AS PART OF MFA

- **Advantages**
  - No additional tokens are necessary
  - Dynamically generated passcodes are safer to use than fixed (static) log-in information
  - Protection to replay attacks as OTP values are uniquely associated to given transaction
- **Disadvantages**
  - A mobile phone is not always available - they can be lost, stolen have a dead battery or otherwise not working
  - Mobile phone reception is not always available - large areas, particularly outside of towns, lack coverage
  - SIM cloning gives hackers access to mobile phone connections
  - Text messages to mobile phones using SMS are insecure and can be intercepted by third parties that can steal and use the token
  - Modern smartphones are used both for receiving email and SMS. So if the phone is lost or stolen and is not protected by a password or biometric, all accounts for which the email is the key can be hacked as the phone can receive the second factor

80

ASSURANCE LEVEL (NIST)

Authenticator = the means used to confirm the identity of a user, process, or device (e.g., user password or token)

Authentication

1. Provides some assurance that the claimant **controls** the **authenticator**; requires at least **single-factor authentication**
2. Provides high confidence that the claimant **controls authenticators**; **two different authentication factors** are required; approved cryptographic techniques are required
3. Provides very high confidence that the claimant **controls** the **authenticator**; authentication based on **proof of possession** of a key through a cryptographic protocol; requires a "hard" cryptographic authenticator

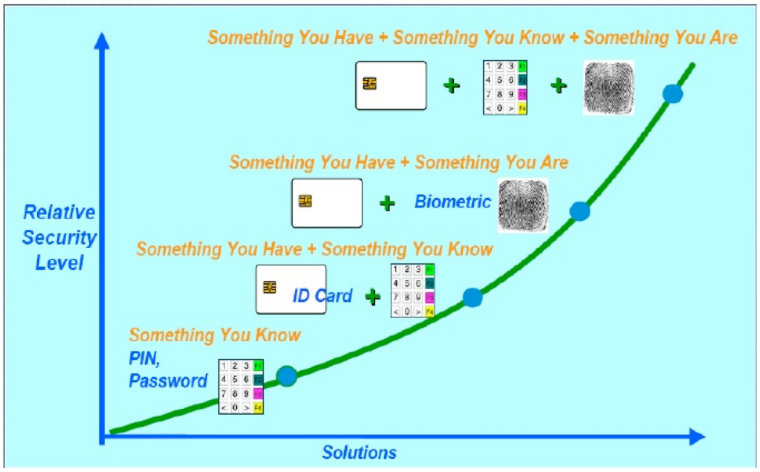
Memorized secrets such as passwords

Cryptographic software and tokens such as apps or tokens

Cryptographic hardware such as smartcards

81

MFA & ASSURANCE LEVELS



82

UNFORTUNATELY . . .

The attack is particularly effective… not because of the technology involved, but because it targets the **human factor** via social engineering

- Many MFA **users are not familiar with this type of attack** and would **not understand they are approving a fraudulent notification**
- Others just want to make it disappear and are simply not aware of what they are doing since they approve similar notifications all the time

- … even MFA procedures can be vulnerable to phishing attacks
- Example: **MFA fatigue attacks**
  - **Idea**: flooding a user's authentication app with push notifications (or SMS) in the hope they will accept one and therefore enable an attacker to gain entry to an account
  - Technical difficulty: simple
    - it only requires the attacker to manually, or even automatically, send repeated push notifications while trying to log into the victim's account
    - It does require the attacker to have the victim's credentials, which "could be obtained via, for instance, brute forcing
    - Once attackers obtain passwords, they will perform the push notification spamming repeatedly until the user approves the login attempt and lets the attacker gain access to the account
  - This usually happens because the **user is distracted or overwhelmed by the notifications** and, in some cases, it can be misinterpreted as a bug or confused with other legitimate authentication requests

How can we mitigate this type of attacks?

83

ENTERS FIDO (1)

- In September 2018, Google reported that it has not had any of its 85,000+ employees successfully phished on their work-related accounts since early 2017, when it **began requiring all employees to use physical Security Keys in place of passwords and one-time codes**
- Prior to 2017, Google employees used MFA by relying on one-time passwords generated by a mobile app — Google Authenticator
- A Security Key implements a form of multi-factor authentication known as **Universal 2nd Factor (U2F)**, which allows the user to complete the login process simply by inserting the USB device and pressing a button on the device
  - The key works **without** the need for any **special software drivers**
- Once a device is enrolled for a specific Web site that supports Security Keys, the user no longer needs to enter their password at that site

ENTERS FIDO (2)

- The Web Authentication API (**WebAuthn**) is a standard put forth by the World Wide Web Consortium in collaboration with the FIDO Alliance
- WebAuthn eliminates the need for users to constantly type in their passwords and instead uses **Public Key Cryptography in combination with a challenge response protocol**
- **This avoids the threat from common password-stealing methods like phishing and man-in-the-middle attacks**
- Let's see a bit more in details how FIDO works…

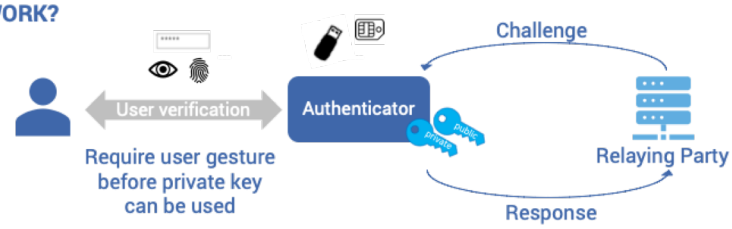
FIDO: PHISHING RESISTANT AUTHENTICATION

FIDO (Fast Identity Online) is the World's Largest Ecosystem for Standards-Based, Interoperable Authentication

https://fidoalliance.org/

Ensuring **strong authentication** and **reducing the use of passwords**

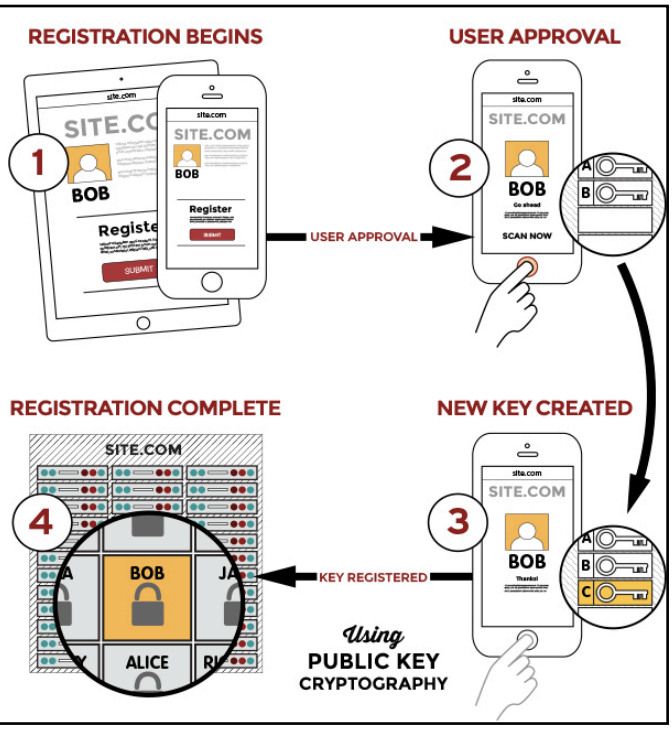
HOW DOES FIDO WORK?



86

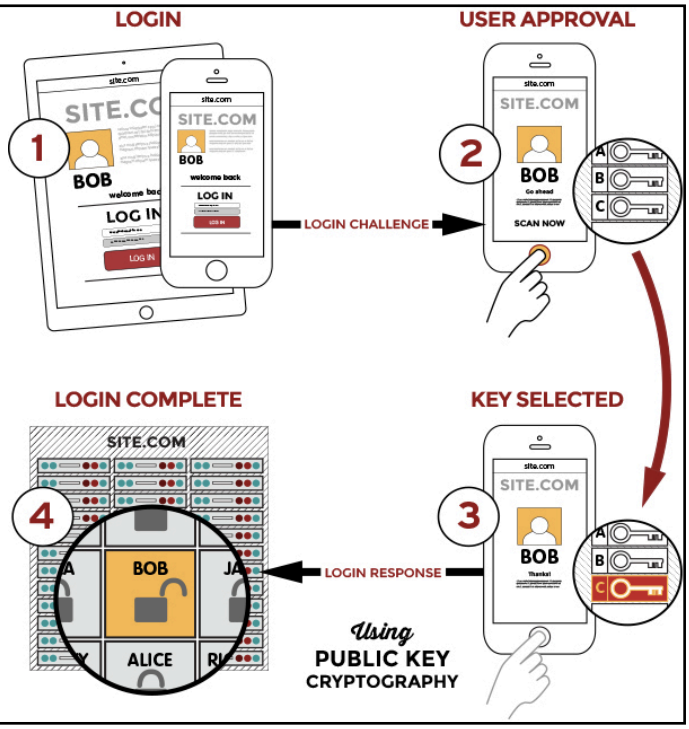
FIDO IN PICTURES (1)

- User is prompted to choose an available FIDO authenticator that matches the online service's acceptance policy
- User unlocks the FIDO authenticator using a fingerprint reader, a button on a second–factor device, securely–entered PIN or other method
- User's device creates a new public/private key pair unique for the local device, online service and user's account.
- Public key is sent to the online service and associated with the user's account
  - The private key and any information about the local authentication method (such as biometric measurements or templates) never leave the local device.



FIDO IN PICTURES (2)

- Online service challenges the user to login with a previously registered device that matches the service's acceptance policy
- User unlocks the FIDO authenticator using the same method as at Registration time
- Device uses the user's account identifier provided by the service to select the correct key and sign the service's challenge
- Client device sends the signed challenge back to the service, which verifies it with the stored public key and logs in the user



HOW FIDO WORKS (1)

- FIDO authentication requires an initial registration step
- In cases where the user device supports multiple forms of authentication (i.e. fingerprint scanner, voiceprint recorder, face ID, etc.), the user is asked to choose a FIDO compliant authenticator from the options available on the device that matches the authenticating app's acceptance policy
- The user then unlocks the FIDO authenticator using whatever mechanism is built into the authenticator – e.g. by providing a fingerprint, pressing a button on a second–factor device, or entering PIN
- Once the authenticator is unlocked, the user's device creates a new and unique public/private cryptographic key pair that will be used for authenticating access
  - The **public key** is then **sent to the online service and associated with the user's account**
  - The **private key** and all other sensitive data related to the chosen authentication method – for example, biometric prints – **remain on the local device and never leave it**

HOW FIDO WORKS (2)

- Authentication requires the client device to prove possession of the private key to the authenticating service by successfully responding to a cryptographic challenge
- The **private key can only be used after successfully authenticating using the registered authenticator**, for example by swiping a finger on the fingerprint sensor, entering a PIN, pressing a button, …
- The device then uses the user account identifier provided by the service to select the correct key and cryptographically sign the service's challenge
- The signed challenge is sent back to the service, which verifies it with the stored public key and logs in the user

Full details available at https://fidoalliance.org/

FIDO AS A STANDARD

- It is a set of open technical specifications that define user authentication mechanisms that reduce the reliance on passwords
- 3 sets of specifications are available
  1. **FIDO Universal Second Factor (FIDO U2F)** provides a standard means for interfacing a second-factor hardware authenticator
     - This interface is mainly used by Web browsers to allow Web applications to interface with a user's hardware authenticator
     - With the release of FIDO2, U2F has been renamed as CTAP1.
  2. **Client to Authenticator Protocols (CTAP)** enables users to authenticate to a Web or native application using an authenticator embedded in the host computer or connected to the host computer
     - Similar to FIDO U2F, CTAP is designed to provide a standardized interface to a hardware authenticator
  3. **FIDO Universal Authentication Framework (FIDO UAF)** defines a framework for users to register their device (i.e. laptop, desktop, mobile) to the online service and select one of the local authentication mechanisms available on the device to authenticate its user
     - The online service can select which locally available authentication mechanism it will accept
     - *Example*: users can register their mobile device and select its embedded fingerprint sensor as the local authentication means used to authenticate them to the online service (another common authentication mechanism is entering a PIN)
     - Once registered and accepted by the online service, users can authenticate to the online service using the local authentication action registered instead of using the more traditional username and password options

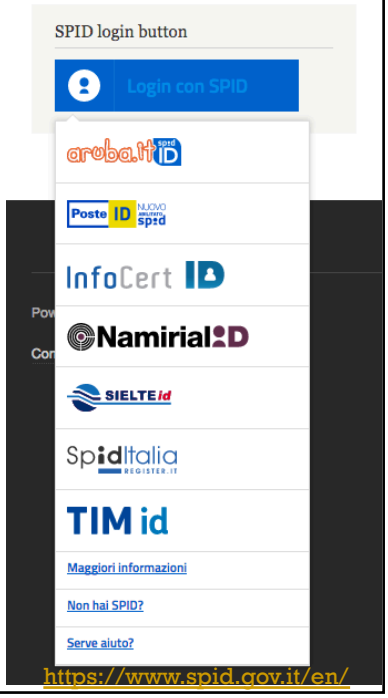
92

OUTSOURCING AUTHENTICATION

Identity providers: the lazy (and scalable) approach to identity management

OUTSOURCING AUTHENTICATION

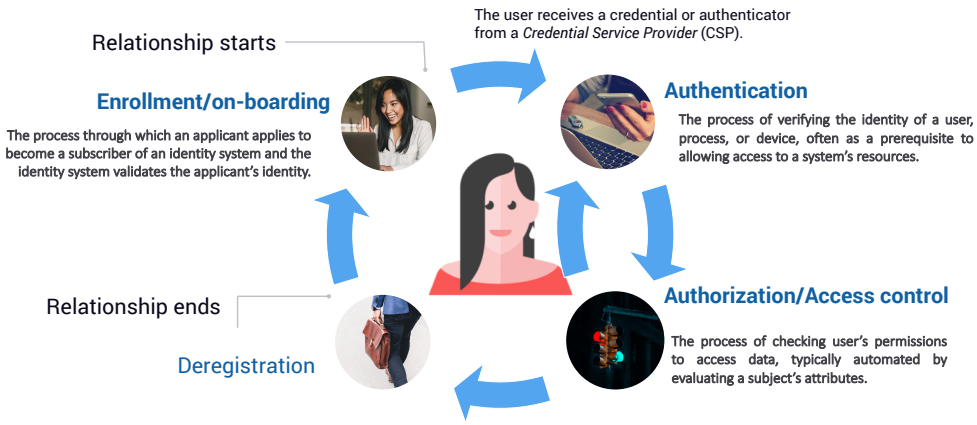
- Several attempts to make a business out of authentication in the consumer space without too much success
- Hardware tokens used as second factor authentication were never widely adopted because of their cost
- Smartphones may change this as they may run applications capable of computing or receiving one time passwords as second factors for authentication without the need of buying dedicated hardware
- National digital identity infrastructures sponsored by many Member States in Europe obtained different levels of success because of the not always clear business models for identity providers
  - An example of this is the *Sistema Pubblico di Identità Digitale (SPID)* in Italy that was more widely adopted only after the pandemics



https://www.spid.gov.it/en/

THE PROBLEM

- Securing all the phases of the identity management lifecycle is far from being trivial
- Organizations (especially, small and medium sized ones) lack resources to devote to security and need to focus on their core business…



94

A SOLUTION

- Delegate authentication to a **trusted** 3rd party identity provider
- Organizations can consume authentication assertions while **assuming** it has deployed a secure identity management lifecycle
- Applications and services becomes dependable on the 3rd party identity provider

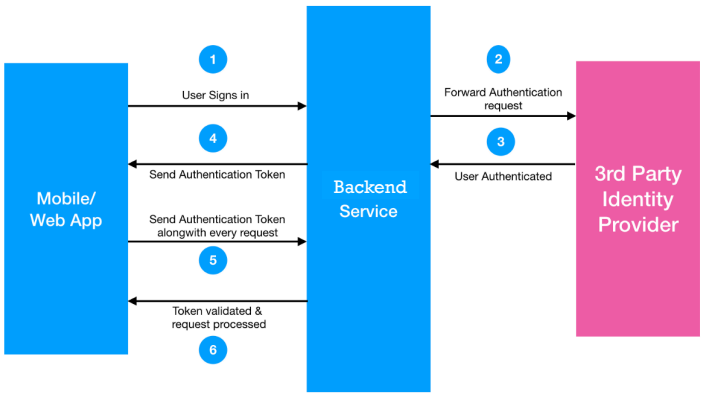


Figure adapted from
https://medium.com/humbledog/azure-app-service-custom-authentication-1b9cab20657

95

SINGLE-SIGN-ON (SSO)

A standard way to outsource authentication to a trusted 3rd party identity provider

- Having to remember many passwords for different services is a nuisance
  - With SSO, you have to enter your password only once
- A simplistic single-sign on service could store your password and do the job for you whenever you have to authenticate yourself
  - Improves on user experience but raises new security concerns
- **System designers have to balance convenience and security**
  - Ease-of-use is an important factor in making systems really useful, but many practices which are convenient also introduce new vulnerabilities

96

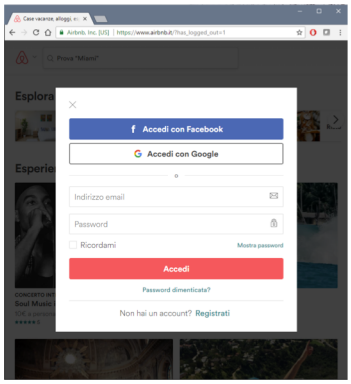
WIDESPREAD ADOPTION OF SSO

Single Sign-On (SSO) allows users to access multiple apps through a single authentication act

SAML 2.0

consolidated, corporate & governmental environments

used for social network (billions of user)



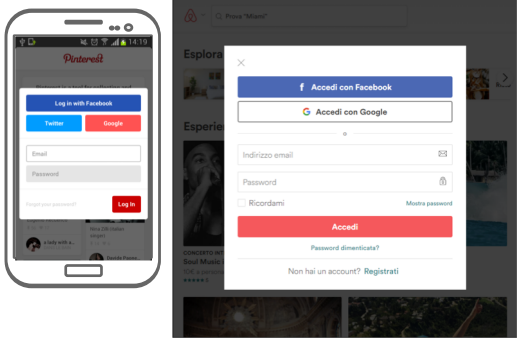
97

PROS & CONS OF SSO

Privacy may also be problematic as the identity provider may link actions of users performed online across different services and applications that trust it

Single Sign-On (SSO) allows users to access multiple apps through a single authentication act

- *Usability*: only a password to remember for several apps
- *Security*: more complex passwords
- *Usability*: shared sessions
- *Security*: Only 1 password to compromise



→ SSO + Multi-Factor Authentication solutions

98

WRAP-UP (1)

Authentication can be seen as the front door of an application, service or ICT system…

- Authentication amounts to verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system
  - Here we focused on user authentication
- Passwords are one of the most widespread and accepted method for user authentication despite their shortcomings
- Adequate security mechanisms should be put in place to protect stored passwords
  - Hashing & salting, access control, …
- Authentication is only one phase of the identity management lifecycle
  - Also other phases (e.g., identity proofing) may have weaknesses
- Given the importance and difficulty of identity management, it is frequently outsourced to trusted third parties with advantages and disadvantages also from the viewpoint of security (and privacy)
  - This paves the way to Single Sign On (SSO) solutions

99

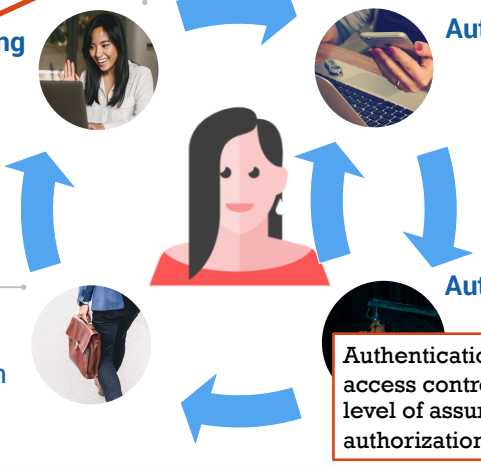
WRAP-UP (2)

Assurance level in enrollment is significant also for authentication; it is pointless, e.g., to have AAL 3 with IAL 1…

Relationship starts

Enrollment/on-boarding

Onboarding is crucial to security of identity management as it associates credentials with user identifiers



Authentication

Passwords are knowledge factors that are widely used but should be either complemented with other factors to provide minimum security guarantees or avoided altogether…

Authorization/Access control

Authentication is crucial for the auditability of access control decisions… without adequate level of assurance about user identity, authorization decisions have little value…

Relationship ends

Deregistration

Crucial to deny access after user should be no more active

y & Trust (FBK)

WRAP-UP (3)

Offline attack

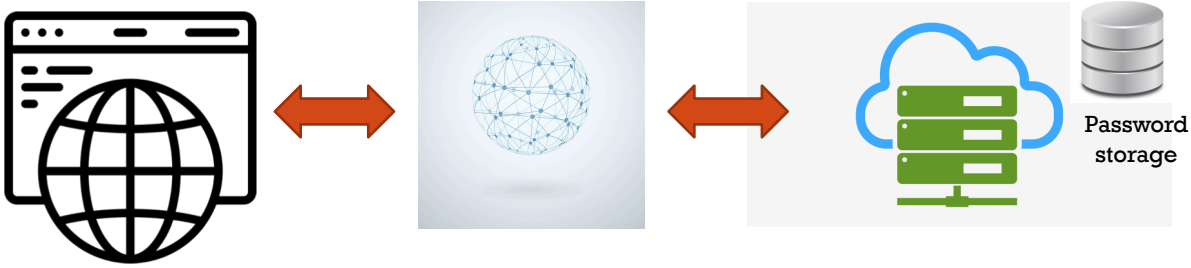
User	Random Salt	Password Hash
Stephen	06917d7ed65c466fa180a6fb62313ab9	b65578786e544b6da70c3a9856cdb750
Lisa	51f2e43105164729bb46e7f20091adf8	2964e639aa7d457c8ec0358756cbffd9
James	fea659115b7541479c1f956a59f7ad2f	dd9e4cd20f134dda87f6ac771c48616f
Harry	30ebf72072134f1bb40faa8949db6e85	204767673a8d4fa9a7542ebc3eceb3a2
Sarah	711f51082ea84d949f6e3efecf29f270	e3afb27d59a34782b6b4baa0c37e2958

Man in the Browser / App

Man in the Middle

LAN based attacker

Exfiltrate



Password storage

Browser / application client

Network

Server / application backend

101