User Authentication

Vítor Francisco Fonte, vff@di.uminho.pt, University of Minho, 2024

Overview

- Digital User Authentication Principles
- Password-Based Authentication
- Token-Based Authentication
- Biometric Authentication
- Remote User Authentication
- Security Issues for User Authentication

Authentication

- Why do we need authentication? Because identity:
  - Is a parameter of an access control decision
  - Is recorded when logging security-relevant events to an audit trail
- User authentication is a necessary process that:
  - Helps to ensure only system users have access to system resources (first line of defence from outside)
  - Helps to ensure a system user cannot impersonate another user (gain access to otherwise unavailable resources)
  - Contributes to security auditing and resource accountability

Authentication

- User authentication is a two-step process:
  - Identification: you announce who you are
  - Authentication: you prove you are who you claim to be
- Authentication is often applied to subjects other than human users:
  - Entity authentication is the process of verifying the identity claimed by some system entity

Password-Based Authentication

The most common form of authentication and first line of defence

1. The user inputs a name and a password.
2. The system verifies the user input against entries stored in a password file.
3. If the verification is successful a user authenticated session is established:
    - The system may require further authentication at a later stage (e.g. session is idle for a period of time, or the user requested a particularly security sensitive operation).
    - The system may also simply choose to simply close a session automatically if it is idle for too long.
4. Otherwise:
    - Usually, the login screen will be displayed again and the user will be able to start your next attempt.
    - The system may prevent or delay further attempts when a certain threshold of failed authentications has been reached.

Attacking Password-Based Authentication

- Intercept the password at the time a new user account is created;
- Try to guess the password;
- Get the password from the user through phishing or spoofing, or by keyloggers;
- Get the password from the system by compromising the password file or by social engineering.

Guessing Passwords

Basic attack strategies

- **Exhaustive search (brute force)**
  - Try all possible combinations of valid symbols, up to a certain length.
- **Intelligent search.**
  - Search through a restricted name space.
  - Try passwords that are somehow associated with a user, such as name, names of friends and relatives, car brand, car registration number, phone number, important dates, etc. (OSINT).
  - Try passwords that are known to be popular. E.g. see https://github.com/danielmiessler/SecLists/tree/master/Passwords/Common-Credentials.
  - Try all passwords on a word list (dictionary attack).
  - Combine these approaches with well known password patterns (heuristics are very well developed). E.g. see https://www.openwall.com/john/

Defences Against Password Guessing

- **Change default passwords:** when systems are delivered, they often come with default accounts such as ‘system’ with default password ‘manager’. This helps the field engineer to install the system, but if the password is left unchanged the attacker has an easy job getting into the system. In this particular example, the attacker even gets access to a privileged account.
- **Password length:** to thwart exhaustive search, a minimal password length should be prescribed. Standard Unix systems, however, have a maximal password length set to eight characters only.
- **Password format:** mix upper and lower case symbols and include numerical and other non-alphabetical symbols in your password. The size of the password space is at least $|A|^n$ where n is the minimal password length and $|A|$ is the size of the character set used for constructing passwords.
- **Avoid obvious passwords:** do not be surprised to find out that attackers are equipped with lists of popular passwords and be aware that dictionary attacks have extended the scope of ‘obvious’ quite substantially. Today, you can find an on-line dictionary for almost every language.

Defences Against Password Guessing

Help from the system

- **Password checkers:** as a system manager, you can use tools that check passwords against some dictionary of ‘weak’ passwords and prevent users from choosing such passwords. This imitates – and pre-empts – dictionary attacks against the system.
- **Password generation:** some operating systems include password generators producing random but pronounceable passwords. Users are not allowed to pick their own password but have to adopt a password proposed by the system.
- **Password ageing:** an expiry date for passwords is set, forcing users to change passwords at regular intervals. There may be additional mechanisms to prevent users from choosing previous passwords, e.g. a list of the last ten passwords used. Still, determined users will be able to revert to their favourite password by making a sufficient number of changes until their old password is accepted again.
- **Limit login attempts:** the system monitors unsuccessful login attempts and reacts by locking the user account completely or at least for a certain period of time to prevent or discourage further attempts. The time the account is locked could be increased in proportion to the number of failed attempts.

Passwords and the Human Factor (Again)

- Are longer, complex password better?
  - Users are unlikely to memorised such a password
  - So they tend to write it down somewhere, possibly in a place close to the computer
  - Passwords may end up being more accessible to attackers with physical access to the computer
- What about frequent, mandatory changing of passwords?
  - Users often resort to password patterns based on counters or dates
  - Passwords may end up being more easily guessable

Passwords and the Human Factor (Again)

Lessons from experience

- People are best at memorising passwords they use regularly.
- Hence, passwords work reasonably well in situations where they are entered quite frequently
- But not with systems used only occasionally.
- When changing your password, it is good advice to type it immediately several times.
- It is equally good advice not to change passwords before weekends or holidays **(or academic festivities 😉)**

Spoofing, Phishing and Social Engineering

- Password-based authentication is unilateral:
  - The system verifies the credentials presented by the user.
  - But the user often does not know who has received the credentials.
- In phishing and spoofing attacks the user voluntarily sends the password over a channel, but is misled about the end point of the channel.
- In social engineering attacks the user can be tricked or compelled to share the credentials.

Password-Based Authentication

Spoofing Attacks

- The attacker presents the user with a fake login procedure.
- E.g. A program runs a fake login screen in an idle computer that records the credentials entered by a user, displays a failed authentication message that the user takes as its fault, and exits to the actual system login screen.
- E.g. A fake version of a website the user is (somehow) directed to.
- How to mitigate this type of attacks?
  - Displaying the number of failed logins may indicate to the user that an attack has happened.
  - Trusted path: a guarantee that the user communicates with the actual system and not with a spoofing program (E.g CTRL-ALT-DEL cannot be handled by user programs and invokes MS Windows OS login screen).
  - Mutual authentication: a system may be required to authenticate itself to the user

Password-Based Authentication

Phishing Attacks

- E.g. For example, a message could claim to come from a service you are using, telling you about an upgrade of security procedures, and asking you to enter your username and password at the new security site that will offer you stronger protection.
- Users should take care to enter their passwords only at the ‘right’ site, but in practice it is not always easy to recognise the right site.

Password-Based Authentication

Social Engineering Attacks

- The attacker may impersonate a user and trick a system operator into releasing the password to the attacker.
- Social engineering attacks are more successful when they better understand the psyche of the target (via personal knowledge or OSINT).
  - Is this a person that can be bullied?
  - Is this a person that is very supportive of struggling users?
- Crucial: security training, policies, practices and regular audit of organisations

Password-Based Authentication

Password “caching” attacks

- **Passwords will be held temporarily in intermediate storage** locations such as buffers, caches, or even a web page. The management of these storage locations is normally beyond the control of the user and a password may be kept longer than the user expects.
- E.g. A user of a web browser may be able to recover a previously terminated authentication session by clicking the back button, sometimes even when the session was terminated by that user. Sometimes the user credentials themselves can be recovered using the same technique.
- E.g. User credentials may be exposed in temporary files, in the space that once belonged to a now deleted file, in the volatile memory chips (RAM), due to buffer overruns, inadequate use or improper implementation of cryptographic protocols, etc.

Protecting the Password File

- Attacking a password:
  - Can be intercepted by a keylogger at the machine at the user's end.
  - Can be intercepted in transit so it should be sent through a secure tunnel.
  - Can be exposed by compromising the password.
- An attacker can directly **impersonate** a user when the contents of an **unencrypted password file** are disclosed or when entries in the password file can be modified.
- Even the **disclosure of encrypted password**s may be a concern. **Offline dictionary attacks** can then be conducted and protection measures such as limiting the number of unsuccessful login attempts would not come into play.

Protecting the Password File

- Approaches to protecting the password file:
  - Cryptographic protection.
  - Access control enforced by the operating system.
  - A combination of the two, possibly with further enhancements to slow down dictionary attacks.

Protecting the Password File

Cryptographic protection

- We do not even need an encryption algorithm. A one-way function will do the job.
  - A one-way function is a function that is relatively easy to compute but significantly harder to undo or reverse. That is, given x it is easy to compute f(x), but given f(x) it is hard to compute 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 then compares f(x′) with the expected value f(x). If the values match, the user has been successfully authenticated. If f is a proper one-way function, it is not feasible to reconstruct a password x from f (x).
- The password file could now be left world-readable but for offline dictionary attacks. In a dictionary attack, the attacker hashes all words in a dictionary and compares the results against the hashed entries in the password file. If a match is found, the attacker knows that user's password. **One-way functions can be chosen to slow down dictionary attacks** (as is often the case).

TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2022

hardware: 8 x A100
password hash: MD5

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	2 secs	7 secs	31 secs
8	Instantly	Instantly	2 mins	7 mins	39 mins
9	Instantly	10 secs	1 hour	7 hours	2 days
10	Instantly	4 mins	3 days	3 weeks	5 months
11	Instantly	2 hours	5 months	3 years	34 years
12	2 secs	2 days	24 years	200 years	3k years
13	19 secs	2 months	1k years	12k years	202k years
14	3 mins	4 years	64k years	750k years	16m years
15	32 mins	100 years	3m years	46m years	1bn years
16	5 hours	3k years	173m years	3bn years	92bn years
17	2 days	69k years	9bn years	179bn years	7tn years
18	3 weeks	2m years	467bn years	11tn years	438tn years

HIVE SYSTEMS

› Learn about our methodology at hivesystems.io/password

TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2023

hardware: 8 x RTX 4090
password hash: MD5

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	Instantly	Instantly
7	Instantly	Instantly	1 sec	2 secs	4 secs
8	Instantly	Instantly	28 secs	2 mins	5 mins
9	Instantly	3 secs	24 mins	2 hours	6 hours
10	Instantly	1 min	21 hours	5 days	2 weeks
11	Instantly	32 mins	1 month	10 months	3 years
12	1 sec	14 hours	6 years	53 years	226 years
13	5 secs	2 weeks	332 years	3k years	15k years
14	52 secs	1 year	17k years	202k years	1m years
15	9 mins	27 years	898k years	12m years	77m years
16	1 hour	713 years	46m years	779m years	5bn years
17	14 hours	18k years	2bn years	48bn years	380bn years
18	6 days	481k years	126bn years	2tn years	26tn years

HIVE SYSTEMS

› Learn how we made this table at hivesystems.io/password

TIME IT TAKES A HACKER TO BRUTE FORCE YOUR PASSWORD IN 2024

hardware: 8 x RTX 4090
password hash: bcrypt

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	3 secs	6 secs	9 secs
5	Instantly	4 secs	2 mins	6 mins	10 mins
6	Instantly	2 mins	2 hours	6 hours	12 hours
7	4 secs	50 mins	4 days	2 weeks	1 month
8	37 secs	22 hours	8 months	3 years	7 years
9	6 mins	3 weeks	33 years	161 years	479 years
10	1 hour	2 years	1k years	9k years	33k years
11	10 hours	44 years	89k years	618k years	2m years
12	4 days	1k years	4m years	38m years	164m years
13	1 month	29k years	241m years	2bn years	11bn years
14	1 year	766k years	12bn years	147bn years	805bn years
15	12 years	19m years	652bn years	9tn years	56tn years
16	119 years	517m years	33tn years	566tn years	3qd years
17	1k years	13bn years	1qd years	35qd years	276qd years
18	11k years	350bn years	91qd years	2qn years	19qn years

HIVE SYSTEMS

› Learn how we made this table at hivesystems.io/password

Protecting the Password File

Access control at the operating system level

- Access control mechanisms in the operating system restrict access to files and other resources to users holding the appropriate privileges.
- Only privileged users may have write access to the password file. Otherwise, an attacker could get access to the data of other users simply by changing their password, even if it is protected by cryptographic means.
- If read access is restricted to privileged users, passwords in theory could be stored unencrypted.
- But if the password file contains information that is also required by unprivileged users, then the password file must contain encrypted passwords. However, such a file can still be used in dictionary attacks.
- A typical example is /etc/passwd in Unix. Therefore, many versions of Unix store enciphered passwords in a file that is not publicly accessible. Such files are called shadow password files.

Protecting the Password File

Reducing the threat of a dictionary attack through salting

- When a password is encrypted for storage, additional information, the salt, is appended to the password before encryption. The salt is then stored with the encrypted password.
- If two users have the same password, they will therefore have different entries in the file of encrypted passwords.
- Salting slows down dictionary attacks as it is no longer possible to search for the passwords of several users simultaneously.
- Salting makes impractical the usage of pre-calculated tables of hashed passwords (Rainbow tables).

Single Sign-On

- **Multiple authentication can be a significant usability issue.** Forget about the problem of potentially having to remember several different passwords and picking the right one at each occasion; having to re-enter the same password several times is bad enough.
- A single sign-on service solves this problem. You enter your password once. The system may store this password and whenever you have to authenticate yourself again, the system will take the password and do the job for you.
- A single sign-on service adds to your convenience but it also raises new security concerns.
  - How do you protect the stored password? Some of the techniques mentioned previously will no longer work because the system now needs your password in the clear.
  - System designers have to balance convenience and security. Ease of use is an important factor in making IT systems really useful. Unfortunately, many practices which are convenient also introduce new vulnerabilities.

Approaches to User Authentication

- Something you know
- Something you hold (have)
- Who you are
- What you do
- Where you are

Approaches to User Authentication

Something you know

- The user has to know some ‘secret’ — previously shared with the system — to be authenticated. You have already seen a first example of this mode of authentication. E.g. A password, a PIN, personal information, etc.
- In this mode of authentication, anybody who obtains your secret ‘is you’. On the other hand, you leave no trace if you pass your secret to somebody else.
    - When there is a case of computer misuse in your organisation where somebody has logged in using your username and password, can you prove your innocence? Can you prove that you did not divulge your password?
- A password does not authenticate a person, successful authentication only implies that the user knew a particular secret. There is no way of telling the difference between the legitimate user and an intruder who has obtained that user’s password.

Approaches to User Authentication

Something you hold

- The user has to present a physical token to be authenticated. E.g. a physical key that opens the lock of a door, a card or an identity tag, etc.
- A physical token can be lost or stolen. As before, anybody who is in possession of the token has the same rights as the legitimate owner.
- To increase security, physical tokens are often used in combination with something you know (e.g. a PIN) or contain information identifying the legitimate user (e.g. a photo).
- However, not even the combination of mechanisms can totally prevent a fraudster from obtaining the information necessary to impersonate a legitimate user, nor does it stop a user from passing on that information voluntarily.

Approaches to User Authentication

Who you are

- Biometric schemes use unique physical characteristics (traits, features) of a person such as face, fingerprints, iris patterns, hand geometry, or possibly even DNA (future?).
    - A reference biometric reference template is constructed based on multiple samples taken from the user.
- Biometric schemes can be used for:
    - Identification: a 1:n comparison that tries to identify the user from a database of n persons;
    - Verification: a 1:1 comparison that checks whether there is a match for a given user.
- In contrast to password-based authentication, the stored biometric reference template will hardly ever match precisely the template derived from the current measurements.
    - A matching algorithm measures the similarity between reference template and current template.
    - The user is accepted if the similarity is above a predefined threshold.
- The problem of false positives and negatives:
    - Accepting the wrong user (false positive) is a security problem.
    - Rejecting a legitimate user (false negative) creates embarrassment and potential availability (and usability) problems.

Approaches to User Authentication
Who you are

- Two important error rates:
  - FMR = number of successful false matches / number of attempted false matches
  - FNMR = number of rejected genuine matches / number of attempted genuine matches
- Balancing FMR and FNMR:
  - The lower the FMR the higher the FNMR
  - Balance the two errors depends on the application
- Also, biometric traits may be unique but they are not secrets.
  - E.g. Fingerprints can often be retrieved from objects and then duplicated.
  - It is not too difficult to construct rubber fingers that defeat most commercial fingerprint recognition systems.
  - In some contexts, use of biometric authentication may be perceived as unacceptable or undesired

Approaches to User Authentication

What you do

- People perform some mechanical tasks in a way that is both repeatable and specific to the individual. A.k.a behavioural security.
- As before, the authentication system has to be set up so that false positives and false negatives are reduced to levels acceptable for the intended application.
- E.g. Forgery of hand-written signatures are relatively easy to perpetrate for skilled criminals.
    - For greater security, users could sign on device that measures attributes like writing speed and writing pressure.

Approaches to User Authentication

Where you are

- When you log on, the system may also take into account where you are.
- Some operating systems already do so and grant access only if you log on from a certain terminal.
  - A system administrator may only be allowed to log in using the system's physical console.
  - A regular user may only be allowed to log in using a particular workstation in the office premisses.
- Decisions of this kind are becoming more frequent in mobile and distributed computing.
  - A precise GPS location may by required (using a trusted device) at the user end at login.

Approaches to User Authentication

Combining approaches

- E.g. A time-based one-time password (TOTP) application is relies on an initially shared seed (secret, something you know), stored in a user device (something you have), and on time as the moving factor.
- Side note: hash-based one-time password (HOTP) relies on an initially shared seed and on an event-based counter as the moving factor.
  - Often regarded as weaker than TOTP but stronger than passwords (susceptible to brute-force attacks).