Identification and Authentication

Daniel Bosk[1]

Department of Information and Communication Systems (ICS),
Mid Sweden University, Sundsvall.

auth.tex 2068 2014-11-03 10:52:07Z danbos

Overview

Overview

What Is Authentication?

- Authentication is the process of verifying the identity claimed by some system entity.
- I.e. first you enter your username to *identify* yourself.
- Then you enter your password to *authenticate* that you are truly you.

Types of Identification and Authentication

Identification

- Username or User ID
- The person who opened the account
- Personal Identification Number (Swe. personnummer, Eng. Social Security Number)
- Fingerprint
- Iris scan
- DNA sequence ...
- Cryptographic key or certificate

Types of Identification and Authentication

Authentication

- *Who* you are
- *Where* you are
- What you *do*
- Something you *have*
- Something you *know*

Types of Identification and Authentication

- The most common way of authenticating is by something you know, i.e. passwords.
- There is also the possibility of combining with the others to have two-factor authentication.
- E.g., it is common to have something you know together with something you have, e.g. password and mobile phone.

Bootstrapping Authentication

- All systems have begun sometime and somewhere, and they always will.
- Whenever a user is added to an authentication system we need a beginning for that user.
- How do we know it is the correct user before he or she can authenticate with our system?
- It is kind of a “the hen or the egg” problem.

Bootstrapping Authentication

- Bootstrapping authentication is about getting authentication going.
- To do this we can have several approaches.
- First, if we do not care about who the person is, we can just bootstrap authentication by that person setting up authentication mechanisms at account registration.
- This is very common, usually Web services only want to authenticate the person who registered the account.
- If we care a bit more, then we can require ID checks etc. to set up the authentication mechanisms using a helpdesk.
- If we have address etc. then we can send the things the user needs to sign in via mail (be it snailmail or email).
- Depending on what we base authentication, the set up of the mechanisms can be to register a password, a phone number for texting verification codes, or a fingerprint scan, and so on.

Problems with Bootstrapping

- Attacker *intercepts* a password on account creation.
- Attacker *impersonates* the legitimate user.

Problems with Bootstrapping

- It can be costly to manage.
- Sometimes it is a continuous process, if the same bootstrapping procedure is also used for *recovery from failure*.
- Make sure your system can handle forgotten, lost, or aged authentication means.

Single Sign-On

- We could let someone else who has solved the problem already do the authentication for us, e.g. Google or Facebook.
- This way the user only needs one username and password, and he or she only needs to sign in once.
- However, this makes the SSO provider a very attractive target.
- And they are forced to solve our problem anyway.
- The problem is, now we need to trust them to do it properly ...

Single Sign-On

- Usually these services doesn't provide reliable identities, just that it's the user who registered the account.
- If we need real identities, then we'd need to use BankID or similar solution.

Overview

Time of Check to Time of Use

- Whenever we authenticate a user, we do this for a purpose.
- When does this authentication take place in relation to when we make use of it?
- Usually we authenticate a user in the beginning of a session, e.g. at login.
- Equally often we assume the user is authenticated during the entire session, even when fetching coffee, going by the printer – or even when out to lunch.
- Who knows what happens when the user is away from the computer, one thing is for sure: the computer will not know the difference!

Time of Check to Time of Use

- This problem can be solved with *repeated authentication*.
- We could lock our system, either manually or by timeout.
- We could also authenticate anew when we need to do something requiring more privileges, and if it has been a while since last time – compare with sudo(8).

Phising, Spoofing, Social Engineering

- The issue we have solved so far is to design means for the system to identify and authenticate different users.
- We have another important problem to solve too, how does the user know it is the system he or she is authenticating him- or herself to?
- Thus enters the problem of spoofing, phising, and social engineering ...

Phising, Spoofing, Social Engineering

Spoofing A spoofing attack presents the user with a fake interface, tricking the user into believing he or she is communicating with the correct system.

Phishing A phishing attack asks users for their password, or other information, under false pretences, e.g. that the IT department is changing the security system and needs the user to reenter the password.

Social Engineering The technique of social engineering is to exploit fallacies in human psychology. Here, the attacker might call the user and trick said user into doing what the attacker wants. Or, the attacker might call helpdesk, impersonating a user and require a password reset.

Phising, Spoofing, Social Engineering

- To counter spoofing one could show the user the number of failed login attempts, the time and location for the last successful login, etc.
- We also have the trusted path, e.g. Windows uses the Ctrl+Alt+Del to bring up the authentication dialogue upon login.
- We could also have some other type of authentication of the system to the user.
- We will return to this subject when we discuss secure protocols.

Phising, Spoofing, Social Engineering

- Countering phishing and social engineering is a bit harder, and requires a different approach.
- The most obvious approach is to educate and train users to spot these attempts.
- However, since these techniques exploit weaknesses in human nature, it can be hard.
- But keeping strong policies for recovering from authentication failures, e.g. forgotten passwords, can mitigate social engineering attempts.
- Some technological tools and good practices can support users in avoiding phishing attempts too.

Guessing Passwords

- There are generally two approaches to guessing passwords.
- The first one is exhaustive search (brute force), which is to test all possible combinations of valid characters.
- The second one is educated search, here we use some more knowledge.

Guessing Passwords

- Possible educated search is using dictionaries, adapt to password policy, etc.
- Some more advanced techniques for guessing passwords have also been developed.
- E.g. one could take grammar into account, depending on the password type [Bon12; BS12].

Guessing Passwords

- There are also some measures to be taken to counter password guessing.
- First, change default passwords, those are the obvious first guess.
- Increase the length of the passwords.
- We could increase complexity, we know this is popular – but [Kom+11] shows it is worse for users than further increasing the length.

Guessing Passwords

- We can also generate passwords for users, although this might reduce security by use of post-it notes.
- We can also introduce password ageing, can be annoying with too short intervals – and will reduce security once users introduce systems to remember their last changed password.
- Finally we can remove online guessing by introducing limited login attempts, however at the cost of possible denial of service.

Overview

Mittuniversitetet
MID SWEDEN UNIVERSITY

The Password File

- Once we have data which can be used to authenticate users, we need to store this data somewhere for future authentication.
- Traditionally, in the case of passwords, there has always been a password file (or database) containing all users' passwords.
- Naturally we have to protect this data, otherwise if someone got hold of it he or she could impersonate any user in the system.

The Password File

- There are of course different approaches to protect it.
- One way is to encrypt it.
- Another is to let the operating system's or database's access control protect it.
- A third way is to do both.

The Password File

- To encrypt the file, we do not actually need to encrypt it.
- We could apply an unreversible transformation on all the passwords.
- Using a one-way function, or cryptographic hash function, we can make the passwords unreadable but still verifiable.
- Let h be a hash function and p a password, then we simply compute $h(p)$ to store in our database.
- When the user wants to authenticate with a password p' we simply compute $h(p')$ and compare to the stored $h(p)$.

The Password File

- The function h is selected to be a very slow one-way function, maybe we iterate its application to the password (like 10 000 to 100 000 times, not only 5).
- This way we can slow down guessing attacks should anyone ever come across our password database.
- However, as our password file is currently structured, we can still see if two users have the same password – they would have the same hash value.
- This way we can guess the password for all users at once: we make a guess, check if it matches *any* user's password.
- To counter this we add a *salt*, this is a small value (e.g. 128 bits) selected randomly for each user.
- This way all password hashes will be unique and the attack would have to be on a per-password basis.

The Password File

- `bcrypt` implements all this functionality.
- It should also be available in most languages and libraries.

Alternative Approaches

- We mentioned in the beginning some alternatives to passwords, these were:
  - who you are,
  - where you are,
  - what you do,
  - something you know, and
  - something you have.
- Using two or more of these together gave us two- or multiple-factor authentication, this reduces the risk of false positives.

Alternative Approaches

- "Who you are" is becoming more popular as a basis for authentication.
- This covers biometrics like fingerprints or iris scans.
- "What you do" is also a biometric, however, the focus of this one is on things you do, e.g. how you write your signature – including the resulting signature, the writing speed, the pressure at different points, etc.
- Finally, "where you are" is also becoming easier to apply with positioning systems in smartphones and IP addresses of computers.
- E.g. Google employs where you are if you have enabled two-factor authentication, then you will have too verify with your phone if your IP address changes.

Alternative Approaches

- You can also take this to the new level.
- Some Web sites use visitor fingerprinting techniques to identify users.
- This can be used to detect if the real user is authenticating or not.
- However, these fingerprinting mechanisms are typically used for other purposes, to track users' browsing habits to serve targeted ads to them.
- EFF developed a tool to illustrate the privacy invasion of browser fingerprinting in [Eck10].
  - You can test browser fingerprinting at URL `https://panopticlick.eff.org/`.

Alternative Approaches

- A problem with all of these is the accuracy and failure rate of the technologies needed to sample the data used in authentication.
- E.g. a fingerprint reader is prone to error due to greasy fingers, complex passwords can be typed incorrectly, etc.
- And since it introduces another system in some cases, it also introduces trust issues for the user – is this an authentic fingerprint reader, is it really the Google sign-in in this IFRAME, or is it one that will steal my credentials?

References

[BS12] Joseph Bonneau and Ekaterina Shutova. "Linguistic properties of multi-word passwords". In: *USEC*. 2012. URL: http://www.cl.cam.ac.uk/~jcb82/doc/BS12-USEC-passphrase_linguistics.pdf.

[Bon12] Joseph Bonneau. "Guessing human-chosen secrets". PhD thesis. University of Cambridge, May 2012. URL: http://www.cl.cam.ac.uk/~jcb82/doc/2012-jbonneau-phd_thesis.pdf.

[Eck10] Peter Eckersley. "How Unique Is Your Browser?" In: *Privacy Enhancing Technologies*. Springer. 2010, pp. 1–18. URL: https://panopticlick.eff.org/browser-uniqueness.pdf.

[Kom+11] Saranga Komanduri, Richard Shay, Patrick Gage Kelley, Michelle L. Mazurek, Lujo Bauer,