Game Plan

- Background on authentication principles
- Considerations about authentication factors
- Considerations about backend issues
- Questions/discussion/open mike session
Basic Concepts
What is Authentication?

• A means to prove identity ("I am Thor")
• ... or membership of some group ("I am an adult").

• **Not** authorization, which is the process of allowing access to something (usually based on identity or group membership).

• Authentication (AuthN) and Authorization (AuthZ) are often confused.
Basic Authentication Factors

- Knowledge: “Something you know” (or “Something you forget”).
- Possession: “Something you have” (or “Something you lose”).
- Inherence: “Something you are” (or “Something you have been”).
Knowledge

• In the IT world, the classic and most widely-used authentication factor.

• Examples:
  - Passwords,
  - passphrases,
  - PINs,
  - keywords,
  - SSH keys – unless stored on hardware token,
  - X.509 keys – unless stored on hardware token,
  - GPG keys – unless stored on hardware token.
Knowledge - Pros and Cons

• Advantages:
  - Easy to implement.
  - Easy to update/change.
  - Prolific and mostly already there.

• Disadvantages:
  - Easy to compromise.
  - Authentication token not easily controllable: Knowledge can be “copied” trivially.
Possession

• Ubiquitous in many “real-world” domains.
• Gaining popularity in IT.
• Examples:
  − Uniforms,
  − (non-personalized) badges – Sheriff stars and the like,
  − USB tokens,
  − NFC tokens – keyless-go car “keys”,
  − SSH keys \textit{if and only if} stored on hardware tokens,
  − X.509 keys \textit{if and only if} stored on hardware tokens,
  − GPG keys \textit{if and only if} stored on hardware tokens.
Possession - Pros and Cons

• Advantages:
  – Authentication tokens can be controlled and access can be restricted in principle.
  – Compromised or lost tokens are likely detected quickly.

• Disadvantages:
  – Harder to roll out because of physical nature.
  – Cannot be replaced remotely if damaged.
  – More difficult to implement in IT systems.
Inherence

• Used for high-security authentication.
• Rare in the IT world – except for fingerprints and face recognition, mostly on mobile phones.
• Examples:
  - Passports,
  - (personalized) badges – photo-ID cards,
  - fingerprints,
  - iris images,
  - face images.
Inherence - Pros and Cons

• Advantages:
  - Very hard to compromise.
  - Authentication tokens are easy to control: Generally speaking, personal traits are non-trivial to clone.

• Disadvantages:
  - In IT systems, fairly hard to implement reliably and securely.
  - In other domains, sometimes hard to implement correctly as well (for instance, due to the cross-race effect).
  - Recovery from failing authentication is hard – cannot give out a new fingerprint.
Multi-Factor Authentication

- Two-Factor Authentication (2FA) or, more generally, Multi-Factor Authentication (MFA) combines two (or more) authentication factors.
- Purpose: Increase level of security for the authentication.
- Examples:
  - PIN-protected smartcard,
  - photo-ID badge,
  - login with password and authenticator app.
Authentication Factor Considerations
Password Particulars - Password Quality

• Passwords have length limits → dictionary attacks are feasible.
• Usual countermeasure: Complexity rules.
• Observation:
  – Humans are bad at remembering random strings, but
  – computers are good at trying all possible passwords.
• Improvement: Diceware. Pick a random set of words (“CorrectHorseBatteryStaple”) as password. As secure as an 8-letter password.
• NIST guideline: No complexity rules.
Password Particulars - Password Change Frequency

- Common debate whether or not to expire passwords.
- Pro: Password leaks may be undetected, and accounts might be left orphaned. Regular password changes mitigate both.
- Con: Additional burden on users. Typical reaction is to game the system. Bad for security and binds effort.
- Current NIST recommendation is that password changes **not** be enforced.
Password Particulars - Password Reuse

• Reusing a password on many systems is problematic:
  ‐ One compromise translates to many exploitable systems.
  ‐ But popular because it is easier to remember just one password.
  ‐ Also, single sign-on (SSO) is a thing – but SSO at least minimizes the number of potential leaks.

• Ideally, have one password per service and device.
  ‐ Users cannot possibly remember that many passwords.
  ‐ Potential support nightmare.
  ‐ But approaches like OAuth exist and effectively do just that.
Password Particulars - Password Managers

- Support users to keep track of passwords.
- A number of good solutions depending on the use case, but also many really crappy ones.
- Suggested potential candidates to use:
  - Pass (https://passwordstore.org/),
  - KeePass (https://keepass.info/),
  - KeePassXC (https://keepassxc.info/),
  - application password stores (Firefox, Chromium, Thunderbird) – make sure the store is encrypted though.
X.509 Client Certificates - General Observations

- One of the oldest multi-factor systems in IT.
- A number of services support this: Web, VPN, mail.
- Different levels of security:
  - Certificate/private key on password-protected hardware token: True MFA.
  - Certificate/private key in an unprotected file: No MFA at all, but in some respects better than a password (cannot be snooped by shoulder-surfing).
X.509 Client Certificates - Server-Side Config Overview

- Central question: Who do you trust?
- “Trust anchors” need to be defined. Common approaches:
  - Define a set of acceptable Certification Authorities (CAs) – for example, “accept certificates issued by ‘DFN-Verein Global Issuing CA’”,
  - ... possibly in combination with a particular Distinguished Name (DN) requirement – for example, “accept certificates issued by ‘DFN-Verein Global Issuing CA’ that contain the O ‘DFN-CERT Services GmbH’”.
  - In addition, define a set of acceptable DNs – for example, “accept certificates for CN=Tobias Dussa/O=DFN-CERT Services GmbH/ that are issued by ‘DFN-Verein Global Issuing CA’”.
  - Or define a set of acceptable certificates, including the public key of the certificate.
The approaches outlined above blur the line between AuthN and AuthZ. Strictly speaking, the actual AuthN is unchanged – you trust the CA to do the right thing™.

... except for the explicit list of acceptable certificates: Here, no trust is placed in the CA at all, as every particular certificate is hand-picked. If a globally-trusted CA is involved, this maximizes pain and minimizes benefits.

Running your own CA might be a good solution if global trust is not a requirement.
X.509 Client Certificates - Config Snippet Examples

- Verify client certificate is issued by, say, DFN-Verein Global Issuing CA:
  ```
  ssl_verify_depth 3;
  ssl_client_certificate /etc/ssl/certs/dfn-verein-global-issuing-ca.pem;
  ssl_verify_client on;
  ```
- Verify client certificate contains proper O value:
  ```
  if ($ssl_client_i_dn !~ "O=DFN-CERT Services GmbH") {
    return 401;
  }
  ```
- Verify client certificate contains proper OU value:
  ```
  if ($ssl_client_s_dn !~ "OU=CAT") {
    return 401;
  }
  ```
SSH Keys - General Observations

- Very common solution for shell-based access problems.
- Allows passwordless automated access at a reasonable level of security.
- Cryptographically identical to X.509 certificates, but without a trusted central authority.
- Can be forwarded through SSH connections.
SSH Keys - Aspects of Pain

• In many, many (most?) cases, SSH keys tend to proliferate and linger, because they are easy to deploy, but hard to track down.

• SSH keys can be password-protected, but enforcing password protection is very hard.

• Passwordless SSH keys are like written-down passwords, password-protected SSH keys are a little more secure (say, 1.5-Factor AuthN), but only SSH keys on USB tokens are true 2FA.
Backend Considerations
Password Particulars – Storing Securely

● First idea: Store passwords as-is – cleartext. Not good:
  - If someone breaks into the store, all passwords are compromised.
  - Worse: The sysadmin cannot claim ignorance – constant background doubt whether admins misuse that knowledge.

● Better: Store passwords in encrypted form.
  - Preferably using asymmetric ciphers → decryption keys can be split up and stored by trusted third parties.
  - This allows for password recovery should it become necessary – for instance, when adding a new password-storage backend without user interaction.
Password Particulars - Still Storing Securely

- State of the art: Storing passwords in salted and hashed form.
  - "Hashing" is a one-way function: Given a hash value, one cannot easily derive the original password.
  - However, two identical passwords yield the same hash value → "salting" comes into play.
  - "Salting" appends a random value to the password before hashing. This "salt" is not secret and stored with the password hash. To verify a password is correct, look up the salt and hash the password plus salt → if result matches stored hash value, the password was correct.
  - In reality, the "hashing" is way more complicated to make brute-force attacks harder.
Central Authentication - What Has It Ever Done For Us?

• Very old idea. Many techniques have been around since the last millennium. Examples:
  - Yellow Pages (YP),
  - Network Information System (NIS)/NIS+,
  - Lightweight Directory Access Protocol (LDAP),
  - Shibboleth,
  - OpenID.

• Makes credential management easier.
• ... but adds more dependencies.
Central Authentication - Central Benefits

- Only one place to keep data up-to-date → makes life much, much easier if accounts are deactivated or retired or credentials updated.
- No sensitive (credential) data stored on client systems, just on the central AuthN server → fewer headaches worrying about password-storage security.
Central Authentication - Central Worries

• Need to trust authenticating party. This is **not** obvious!

• Single Point of Failure: How to use systems when central AuthN system is not available? Caching helps, but introduces new problems.

• Single Point of Attack: Central authentication servers are very, very, very juicy targets.

• Requires some effort to coordinate (user IDs and so on).

• Finding one password opens lots of systems.
Thank you

Any questions?

www.geant.org