Engineer

Authentication

proof that a request is from who it claims

What it is

Authentication is the check a system makes before it accepts a claimed identity: someone says “I am this account”, and the system asks for evidence that only the holder of that account should be able to produce. Engineers often shorten it to authn, and interfaces call the same moment “sign in” or “log in”.

Identification is the claim itself, such as typing a username. Authentication is the evidence for that claim. Authorization is the separate decision, made afterwards, about what the now-known identity is allowed to do. Even HTTP blurs the line: RFC 9110 names status code 401 “Unauthorized”, yet defines it as a request that lacks valid authentication credentials, and obliges the server to send a WWW-Authenticate header describing how to supply them. The code for “we know who you are, and the answer is still no” is 403 Forbidden.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer realm="api"
```

The evidence comes in kinds, usually called factors. Something you know is a password or a PIN. Something you have is a phone, a hardware security key or a private key file. Something you are is a fingerprint or a face. Multi-factor authentication combines two different kinds, so that a thief who obtains one still lacks the other. Two passwords are still one factor.

Much of it involves no person at all. Services present API tokens to each other, and SSH shows the machine-shaped version clearly: RFC 4252 makes “publickey” the only method every implementation must support, and the client proves it holds a private key by signing with it, so the key never crosses the network. Single sign-on moves the check elsewhere. With OpenID Connect an identity

provider authenticates the user and hands the application a signed ID token, so the application never sees a password. Passkeys, built on the W3C Web Authentication API, keep a private key on the device, scoped to one website, and sign the server's challenge together with the origin the browser is actually talking to, so a lookalike domain receives nothing it can replay.

Use the concept in a decision

Put "What is authentication?" to work in a real decision. We can help relate the concept to the system you are building or buying. Accounts, passwords and permissions arranged so people hold exactly the access their job needs, and leavers lose it that day.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Wrote 1,000 RBAC rules to secure Linux servers — 1,000 access rules written to close a security gap

Wrote 1,000 RBAC rules to secure Linux servers Get in touch →

Why it matters

Every other control assumes this one worked. Permissions, audit logs, rate limits and billing are all keyed on an identity, and if the check that set the identity is weak, each of them faithfully applies to the wrong party. A log line saying an administrator deleted a table records which account did it, and only good authentication turns that into who.

The practical decision is how strong the check must be for each door, and NIST SP 800-63B gives the clearest scale to decide with. Its first assurance level, AAL1, accepts a single factor. AAL2 requires two factors, and a verifier at that level must offer at least one phishing-resistant option. AAL3 requires a phishing-resistant authenticator whose private key cannot be exported. The guideline is blunt about one popular choice: codes typed in by hand, whether sent by text message or shown in an app, are not phishing-resistant, because nothing ties the typed digits to the site that asked for them.

Revision 4 of the same document, which superseded revision 3 on 1 August 2025, also settled several arguments about passwords. A password used as the only factor must be at least 15 characters long, and one used only inside multi-factor login at least 8. Verifiers may impose no other composition rules and may not demand periodic changes, but must force a change when there is evidence of compromise. New passwords are checked against a blocklist of common and breached ones, the user is told why a choice was refused, and the stored form is salted and hashed with a scheme that has a cost factor. Consecutive failed

attempts against one authenticator are capped at 100. For sessions at AAL2 it recommends asking again at least every 24 hours, and after an hour without activity.

Our recommendation follows from that. Where users already hold an account with a competent identity provider, let that provider do the check and accept its signed answer, because a password the application never stores is one it can never leak. Where the application must keep passwords, follow 800-63B literally rather than by memory, and put a phishing-resistant method in front of anything administrative.

What it looks like in practice

When the company moved its source code onto its own git server, authentication was the first item on the list of what running a forge means. The forge chosen is SSH-first and has no administrative web interface, so the way in is a key. A session that arrives anonymously is turned away, as is one that arrives without a key, and every declared repository is marked private besides. The listener that checks those keys, on port 23231, accepts connections from the loopback interface alone, so every login from outside first passes through a jump host, and the firewall's declared surface stays the same size.

The same concern looks different at the scale of a whole organisation. In an earlier role as Infrastructure Engineer, the work was consolidating identities and access across M365, 1Password, Red Hat SSO and OKTA SSO, where permissions had been granted ad hoc, over time, by different people. Least privilege was enforced, and onboarding and offboarding were standardised so that access was revoked promptly instead of lingering after someone moved on. Unauthorised-access risk dropped sharply and access became auditable: the team could finally answer "who can reach this, and why".

Authentication also has a face, and users judge it by its error messages. On one application's sign-in against a major mail provider, the message said the password was incorrect. The password was right; the provider requires an application-specific password for third-party clients and rejects the account password however carefully it is typed. The message was rewritten to name the provider, say that it requires an app-specific password for other clients, and say where to create one. A test now drives that refusal and asserts that the message contains the provider's name and the required credential type, so the old wording cannot return unnoticed.

Where it goes wrong

- **The message blames the user for a cause elsewhere.** A locked account, an expired password, a clock drifting out of step with a code generator and a provider demanding a different credential all arrive as “wrong password”, and the user retypes something that will never work. Name the cause the user can act on, while keeping the reply identical for an account that exists and one that does not, so the form gives nobody a list of valid usernames.
- **A phishing page relays the code in real time.** The user types a password and a six-digit code into a copy of the real page, which forwards both to the genuine site within the 30 seconds that RFC 6238 recommends as the default time step. One-time codes stop stolen and reused passwords; they do little against a live relay. Origin-bound keys such as passkeys sign for one site only, which is why NIST counts them as phishing-resistant and typed codes not.
- **Credentials outlive the people who held them.** Offboarding closes the account in the directory, while an SSH key in an `authorized_keys` file, a personal access token and a shared administrator password carry on working. Each credential needs an owner, a place where it is recorded and a date when someone looks at it again.
- **The session stays valid long after the sign-in.** A cookie or token that lasts for months is a password with a long fuse. Signing out has to end the session on the server, not just delete the cookie in one browser, and a self-contained token that the server never checks back on stays usable until it expires, whatever happens to the account.
- **Account recovery is the weakest entrance.** A strong login beside a reset link sent to an email address makes the mailbox the real credential. Recovery deserves the same scrutiny as the front door, because an attacker will pick whichever one is weaker.

Further reading

- NIST SP 800-63B-4 — the current text: assurance levels, the 15-character password minimum and what counts as phishing-resistant.
- NIST SP 800-63B, revision 3 — the 2017 authentication guideline, kept online and marked as superseded by revision 4 from 1 August 2025.
- RFC 9110, section 11: HTTP Authentication — the challenge and response behind 401, WWW-Authenticate and the Authorization header.
- RFC 4252: The SSH Authentication Protocol — public-key, password and host-based login as SSH defines them.
- RFC 6238: TOTP — how authenticator apps derive their short codes from a shared secret and the clock.
- Web Authentication, Level 3 — the W3C API behind passkeys and its origin-scoped public-key credentials.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is authentication?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/authentication/