Authentication Authorization and PKI Lab

How Does Authentication Work?

In a traditional authentication process, the user types in their credentials, such as a username and a password. The authentication system queries a user directory, which is either stored in the local operating system or on an authentication server. If the credentials match, the user is allowed to access the system. In the second stage, permissions assigned to users determine what objects or operations they are allowed to access, and other access rights, such as allowed access times and rate limits.

Traditional Authentication with Username and Password

The traditional authentication process created several challenges:

- Traditionally, each IT system or application would handle its own authentication. This created a burden on application developers and meant that authentication systems were non-standard, in many cases not secure.
- Applications are increasingly delivered over the web. Today most applications communicate via HTTP and HTTP/S. These are stateless protocols, meaning that in a traditional authentication model, users would have to login to a web application every time they accessed it.
- Password-based authentication is very easily compromised by attackers, and is also inconvenient to users.

New Authentication Methods

The following innovations help address these challenges:

- **Standardized authentication protocols** such as OIDC and SAML make it possible for application developers to use a standardized, secure authentication mechanism, without having to develop one themselves.
- **Token-based authentication** enables users to verify their identity once via an authentication service. The service then issues a signed authentication token to the

application, allowing them to sign in without providing credentials again, until the token expires.

- **Multi-factor authentication** supplements password-based authentication with additional, stronger authentication methods, such as biometric authentication, physical tokens, and one-time passwords sent to mobile devices.

What Is Authentication Used For?

Authentication is a cornerstone of cybersecurity, serving multiple critical purposes across various domains:

- **Securing access to systems and data:** Authentication ensures that only authorized users can access sensitive systems and data. This is vital for protecting organizational resources from unauthorized access and potential breaches.
- **User accountability:** By verifying user identities, authentication systems maintain logs of user activities. This enables organizations to track actions back to specific users, which is essential for audit trails and compliance with regulatory requirements.
- **Protecting sensitive transactions:** In financial services, healthcare, and other sectors handling sensitive transactions, authentication helps verify the identities of parties involved, reducing the risk of fraud and ensuring data integrity.
- **Enabling secure remote work:** With the rise of remote work, authentication is crucial for securing remote access to corporate networks and applications. It ensures that remote employees can safely connect to organizational resources from various locations.
- **Supporting Identity and Access Management (IAM):** Authentication is a key component of IAM systems, which manage user identities and their access to resources. This supports the principle of least privilege, ensuring users have access only to the resources necessary for their roles.
- **Enhancing user experience:** Modern authentication methods, such as single sign-on (SSO) and biometric authentication, streamline user access to multiple applications and services. This reduces the need for multiple passwords and improves overall user convenience and security.

Authentication vs. Authorization: What's the Difference?

Authentication is the gatekeeper that decides who gains access to an organization's resources, including critical systems like databases, networks, business apps, and web applications. Once users are authenticated, they undergo authorization to determine which resources or specific functions they can access, based on the system's role or permission structure.

The difference between authentication and authorization can be summarized as follows:

- **Authentication** is responsible for verifying the identity of a user, process, or device
- **Authorization** is responsible for identifying if an authenticated entity has permission to access certain resources, or perform certain operations

Access control refers to the complete process of granting access to users, including both authentication and authorization.

What are Authentication Factors?

An authentication factor is a certain proof that verifies a user's identity. Here are the three categories of authentication factors:

Knowledge Factor

A knowledge factor is a category of credentials that users are expected to know. Common knowledge factors include usernames, passwords, personal identification numbers (PINs), and answers to security questions.

Once a user logs in to an application, the security system asks them to provide their credentials – typically a username and a corresponding password. However, since passwords consist of a sequence of numbers, special characters, and letters, threat actors can easily crack, guess, or steal them. You can mitigate this by adding multifactor authentication (MFA).

Possession Factor

This factor requires users to provide evidence of possessing physical items, such as SIM cards, smart cards, mobile phones, FIDO2 security keys, and hardware OTP tokens. By checking whether the user has a certain piece of hardware, organizations can make it much more difficult to breach.

A threat actor can bypass this by conducting a swapping attack and gaining remote access to or stealing a certain piece of hardware. However, bypassing possession factors is much more difficult than launching a brute force attack.

Inherence Factor

Inherence is considered the strongest authentication factor because it asks users to confirm their identity by presenting evidence inherent to unique features. Common inherence factor examples include biometrics like fingerprint scans, retina pattern scans, and facial recognition.

Types of Authentication

The primary types of authentication used to authenticate users and service connections.

- **Token authentication** – a commonly-used authentication protocol that allows users to authenticate themselves once and receive a token verifying their identity. As long as the token is valid, the user can access the website or application without signing in again.
- **Password authentication** – this requires users to memorize their credentials—typically a username and password in the form of letters, numbers, or special characters. The combination of username and password verifies the user's identity. The more complex the password and the more frequently it is renewed, the more secure the account.
- **Biometric authentication** – identifies individuals based on their unique biological characteristics. It stores data about an individual—for example, their fingerprint or the shape of their iris—and then compares a real-time reading with this stored data. Biometric authentication is convenient for users and is inherently secure.
- **Certificate-based authentication** – uses a digital certificate to identify a user before accessing a resource. Digital certificates are impossible to forge without possessing the private key. It can be used to authenticate a user, device, or service account. Most certificate-based authentication solutions come with a cloud-based management system.
- **Multi-factor authentication (MFA)** – a combination of two or more authentication factors. These can include any of the above types. Combining

several factors significantly improves security and makes it much more difficult for attackers to compromise accounts.

- **Passwordless authentication** – allows a user to access an app or IT system without entering a password or answering security questions. Instead, users provide other forms of proof such as fingerprints, proximity badges, or codes generated by hardware tokens. This authentication is often combined with MFA and single sign-on (SSO).

Authentication Protocols

An authentication protocol is a set of rules that allow a system to verify the identity of an endpoint (laptop, desktop, phone, server, etc.) or a user. Here are a few common authentication protocols.

Password Authentication Protocol (PAP)

PAP is the least secure protocol for authenticating users, primarily because it is not encrypted. This is a login process that requires a username/password combination to access the specified system and verifies the provided credentials against a user directory.

Challenge Handshake Authentication Protocol (CHAP)

CHAP is an authentication protocol that uses a three-way exchange to authenticate users, verifying their identity with strong encryption. This works as follows:

1. The local device sends a “challenge” to the remote host

2. The remote host sends a response using a cryptographic hash function

3. The local device checks if the hash value of the response matches the expected response, and if so, establishes an authenticated connection (“handshake”). Otherwise, it closes the connection.

CHAP is more secure than PAP, because PAP only performs authentication when the user is first authenticated, while CHAP verified authentication on an ongoing basis.

OpenID Connect (OIDC)

OIDC leverages the authentication and authorization mechanisms of OAuth 2.0, commonly applied by numerous identity providers. It was created by the OpenID Foundation (OIDF), a non-profit dedicated to OpenID technology.

Here is the key difference between OIDC and OAuth 2.0:

- OAuth 2.0 is an authorization protocol
- OIDC is an identity authentication protocol

OIDC helps a client service verify the identity of end-users. It can also share (on request) user claims such as name and email address.

OIDC works with various clients, including single-page applications (SPA) and mobile applications. Here are key benefits of OIDC:

- You can use OIDC for single sign-on (SSO) across several applications.
- OIDC uses JSON Web Tokens (JWT), and HTTP flows to avoid sharing end-user credentials with client services.
- The protocol comes with built-in consent, requiring explicit consent from users before sharing their data.
- OIDC is simple to implement and is ideal for use in mobile applications.

Lightweight Directory Access Protocol (LDAP)

LDAP is a software protocol that enables users or applications to locate data about organizations, individuals and other resources, such as files and devices in a network. It can be used for resources on the public Internet or a corporate Intranet. The LDAP directory tells the user where in the network something is located. For example, it is possible to search for a specific user or a service available on the network. LDAP returns the hostname, and then the user can use DNS to obtain the IP and connect to it.

JSON Web Token (JWT)

JWT is an encoded version of a “claim”, a secure transfer of information between two parties. A claim can be used to:

- Assert that a specific party issued the token and it is authentic
- Determine how long the token is valid
- Provide information about permissions granted to the user
- Provide general information about the user which can be used by the application

JWTs use a digital certificate to prove who issued the claim. Technically, a JSON Web Token includes three parts: a header, specifying the algorithm used in the certificate, a payload, which contains the information included in the claim, and the digital signature.

Best Practices for Authentication Security

Use Passwordless Authentication

Passwordless authentication replaces traditional passwords with alternatives like biometrics, hardware tokens, or one-time codes, enhancing security and user experience by reducing password-related breaches. It often integrates with identity providers using standards like WebAuthn and can be combined with multi-factor authentication (MFA) for added security, ensuring protection even if a user's biometric data or hardware token is compromised.

Use Secure Authentication Tokens

Authentication tokens must be securely stored and transmitted to prevent unauthorized access. This includes encrypting tokens during transmission and storage using secure algorithms and employing secure storage solutions like hardware security modules (HSMs) or secure key management services. Using short-lived tokens and implementing token revocation mechanisms can reduce the impact of token theft. Educating developers on secure token handling is essential to prevent vulnerabilities like token leakage.

Monitor and Log Authentication Attempts

Monitor and log all authentication attempts to detect suspicious activities. Implement logging mechanisms that capture login times, IP addresses, and user-agent strings. Use centralized logging systems and SIEM tools for real-time analysis, with automated alerts for unusual patterns like multiple failed login attempts or logins from unfamiliar locations.

Implement Account Lockout Mechanisms

To protect against brute-force attacks, implement account lockout mechanisms that disable accounts temporarily after a set number of failed login attempts (e.g., 3-5). Configure lockout durations based on security needs, from a few minutes to several hours.

Use progressive delays to increase lockout duration with each failed attempt, deterring attackers further. Ensure lockout events trigger alerts for administrators to investigate potential attacks and assist legitimate users in regaining access.

Public Key Infrastructure (PKI) Definition

Public key infrastructure (PKI) refers to tools used to create and manage public keys for encryption, which is a common method of securing data transfers on the internet. PKI is built into all web browsers used today, and it helps secure public internet traffic. Organizations can use it to secure the communications they send back and forth internally and also to make sure connected devices can connect securely.

The most important concept associated with PKI is the cryptographic keys that are part of the encryption process and serve to authenticate different people or devices attempting to communicate with the network.

Why is PKI important?

PKI is crucial because the encryption and authentication it manages and makes possible ensures trustworthy, secure communication online. For an enterprise, PKI can make the difference between an intruder gaining access to the network through a connected device and keeping a potentially dangerous threat away from the organization.

How Does PKI Work?

PKI (Public Key Infrastructure) uses certificates and keys for secure communication. It involves two keys: a public key, available to everyone for encrypting messages, and a private key, kept secret and used for decryption. These keys are mathematically connected, making it difficult to derive the private key from the public one. Certificates, issued by a certificate authority (CA), verify the identity of devices or individuals and ensure their authenticity by validating the certificate's legitimacy.

Symmetric encryption

Symmetric encryption uses a single key for both encrypting and decrypting data. During World War II, Germany used symmetric encryption for secure communication. The process involves encrypting a message through a series of permutations, where the same input can yield different outputs each time, making it challenging to deduce the encryption method.

The term "symmetric" reflects the use of the same key for both encryption and decryption. While this method is secure, the key's compromise poses a risk, as it is essential for both operations.

Asymmetric encryption

The risk of symmetric encryption is solved with asymmetric encryption. With asymmetric encryption, two different keys are created to encrypt messages: the public key and the private one. The message still has to pass through complicated mathematical permutations to get encrypted. However, the private key decrypts it, and the public key encrypts it.

The mathematical properties used to make public and private keys today are Elliptic-curve cryptography (ECC), Rivest-Shamir-Adleman (RSA), and Diffie-Hellman. Each uses different algorithms to make encryption keys. However, they each share the same overall principles regarding how the public and private keys are related.

For example, the RSA 2048 algorithm generates two random prime numbers that are each 1024 bits in length. These are then multiplied by each other. The answer to that problem ends up being the public key. The two random prime numbers used are the private key.

If the two prime numbers are smaller, including, for instance, only two digits, it will be relatively easy for a program to figure out what they are. However, because they each have 1024 digits, it is extremely difficult to figure them out—even when you know the product of the equation.

Sensitive data exposure

Sensitive data exposure is a common cyberattack method where critical information like credit card details, medical records, Social Security numbers, and passwords is compromised. Protecting this data is crucial due to stringent regulations like GDPR. To safeguard data:

1. Encrypt Data: Use secure encryption for data at rest and in transit.

2. Use SSL Certificates: Establish encrypted connections between web browsers and servers.

3. Hash Passwords: Apply strong hashing functions for stored passwords.

4. Employ Updated Algorithms: Ensure the use of robust and current algorithms, keys, and protocols.

What Are PKI Certificates?

PKI certificates function like digital passports, enabling entities to exchange PKI-encrypted data securely. They include the public key and are issued by a Certificate Authority (CA), with the CA's role confirmed by a Registration Authority (RA) that handles certificate requests. Certificates are stored in a certificate database on the CA's server and in a certificate store on local devices. The certificate policy outlines the roles and identities of all involved entities and is published within the PKI perimeter, often linked in the certificate itself, particularly for X.509 certificates.

Common Uses of PKI Certificates

1. **Hypertext Transfer Protocol Secure (HTTPS)**: In HTTPS, the certificates identify each website the user tries to reach to make sure the messages sent back and forth are not intercepted or changed. If someone gains unauthorized access, they can engage in fraudulent activity, such as send fake wire transfers or take people's credit card information. HTTPS can also help prevent a range of man-in-the-middle (MITM) attacks because the hacker has to know how to decrypt the information to effectively intercept and then change or steal the data being sent between two entities. When the communicating parties have to encrypt their messages, not only is key data kept secure but so are passwords and other private information that hackers want to get their hands on.
2. **Secure Shell (SSH)**: SSH provides authentication for computers and users, and uses X.509 certificates. Although different SSH protocols can use different certificate formats, they all perform the same basic function: making sure users and computers are who they claim to be.
3. **Signing and encrypting emails**: Certificates also come into play when you have to make sure your email communications are secure. As with SSH, there are different options for the implementation of PKI certificates for sending emails, but they perform the same essential functions: securing emails that get sent and received, while also ensuring the sender and receiver are who they claim to be.

How To Get a PKI Certificate

The process of creating a certificate follows several, logical steps. First, a private key is created, which is used to calculate the public key. Then, the CA requires the private key owner's attributes presented for verification.

After that, the public key and the owner's attributes are encoded into a digital signature known as a certificate signing request (CSR). This then gets signed by the owner of the key. The signature the owner provides serves as proof that they are the rightful possessor of the private key.

The final step involves the CA. The CSR gets validated by the CA, which then also adds its own signature to the certificate using the CA’s private key. At this point, the certificate is considered legitimate, and communication can commence.