# NETWORK SECURITY ESSENTIALS

**BCA – IV**

**Credits – 4**

**Evaluations – 5**

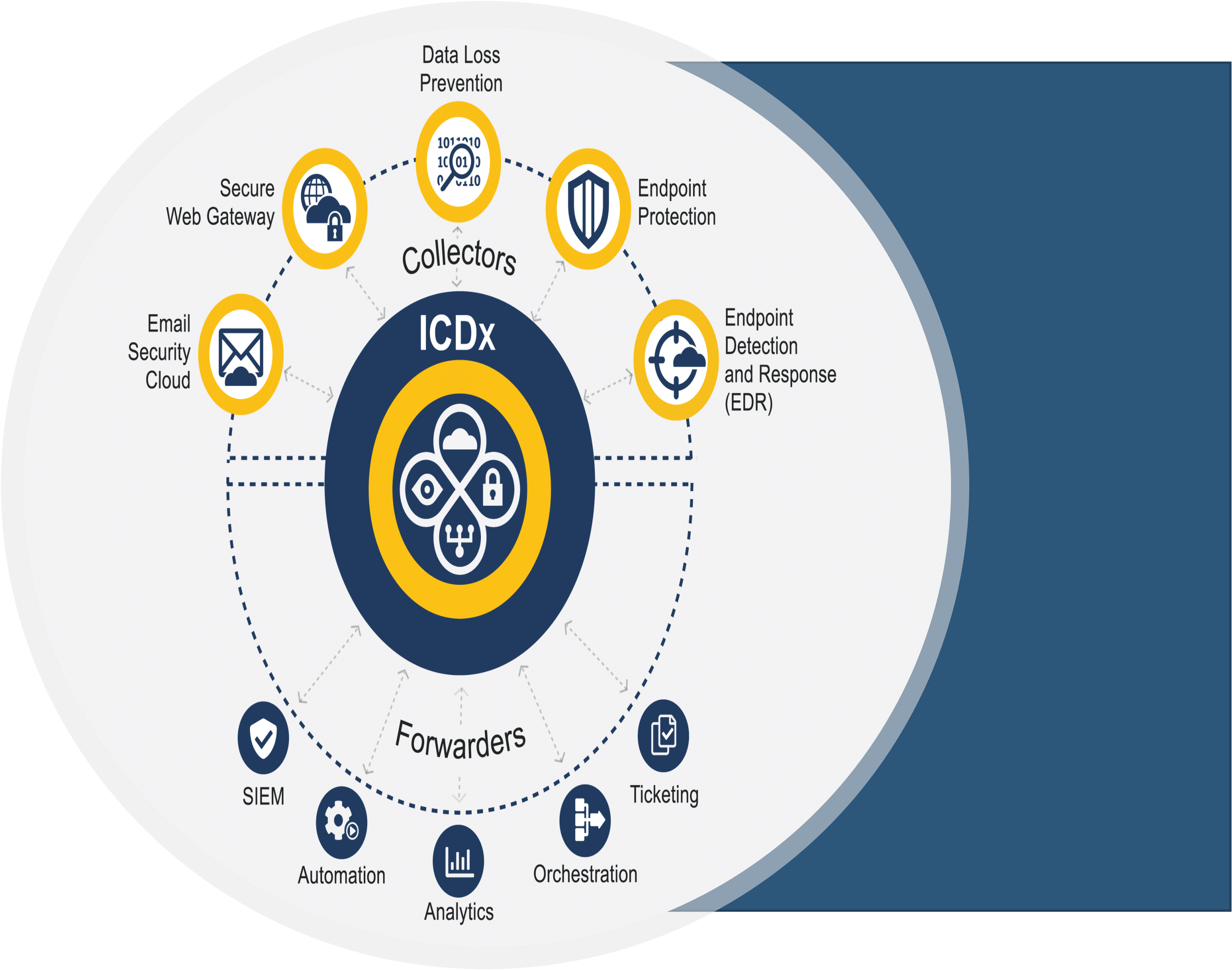
The course is designed to build an understanding of various network security components, protocols and creating the awareness about the issues due to security.

Pre-requisites: An understanding of Basic Computer Networking and security

# UNIT 7

Network Security Applications:

- Authentication applications
- E-mail security



# AUTHENTICATION

Authentication is the act of proving an assertion, such as the identity of a computer system user. In contrast with identification, the act of indicating a person or thing's identity, authentication is the process of verifying that identity. It might involve validating personal identity documents, verifying the authenticity of a website with a digital certificate, determining the age of an artifact by carbon dating, or ensuring that a product or document is not counterfeit.

- **Identification** occurs when a subject claims an identity (such as with a username)
- **Authentication** occurs when a subject proves their identity (such as with a password)
- Once the subject has a proven identity, **Authorization** techniques can grant or block access to objects based on their proven identities.

Authentication is the act of establishing identity via the presentation of information that allows the verifier to know the presenter is who or what it claims. This identity could be any number of things, including:

- People
- Systems
- Applications
- Messages

Need for authentication?

- To control access to a system or application
- To bind some sensitive data to an individual, such as for encryption
- To establish trust between multiple parties to form some interaction with them
- To assure that a piece of information is genuine

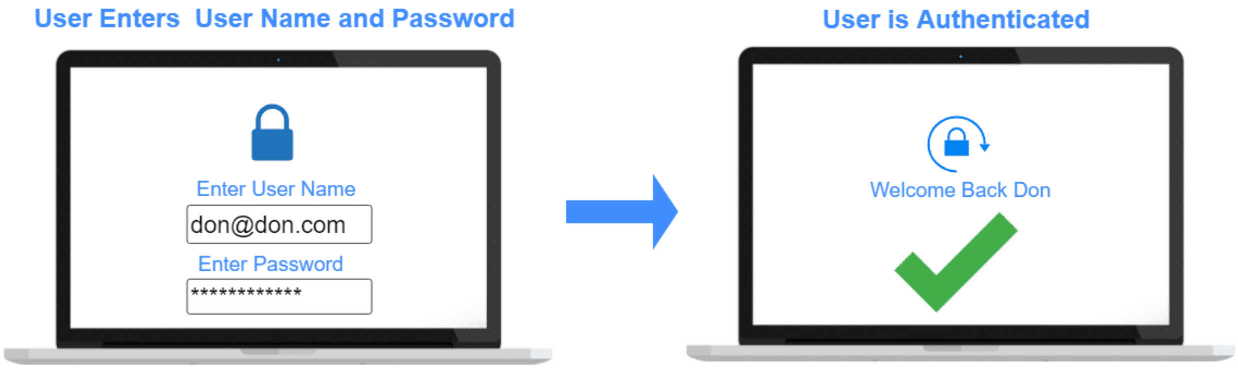
# TYPES OF AUTHENTICATION

There are many different types of authentication that can be used in an application. The selection of the most appropriate type of authentication will depend on the needs of the application; use this guide to determine which makes the most sense for your application.

- Basic, single-factor authentication
- Multi-factor authentication
- Cryptographic authentication

These authentication types apply to all classes of entity that require authentication: systems, users, messages, and applications.

# BASIC / SINGLE FACTOR AUTHENTICATION (SFA)



Basic authentication refers to password-based authentication. A password can be any information that is used to verify the identity of a presenter. Common examples that fall into this category are:

- The common password
- Host or system names
- Application names
- Numerical IDs

Single-factor authentication (SFA) is a process for securing access to a given system, such as a network or website, that identifies the party requesting access through only one category of credentials.

The most common example of SFA is password-based authentication. Password security relies on the diligence of the system administrator or user who sets up the account. Best practices include creating a strong password and ensuring that no one can access it.

# SECURITY ISSUES IN SFA

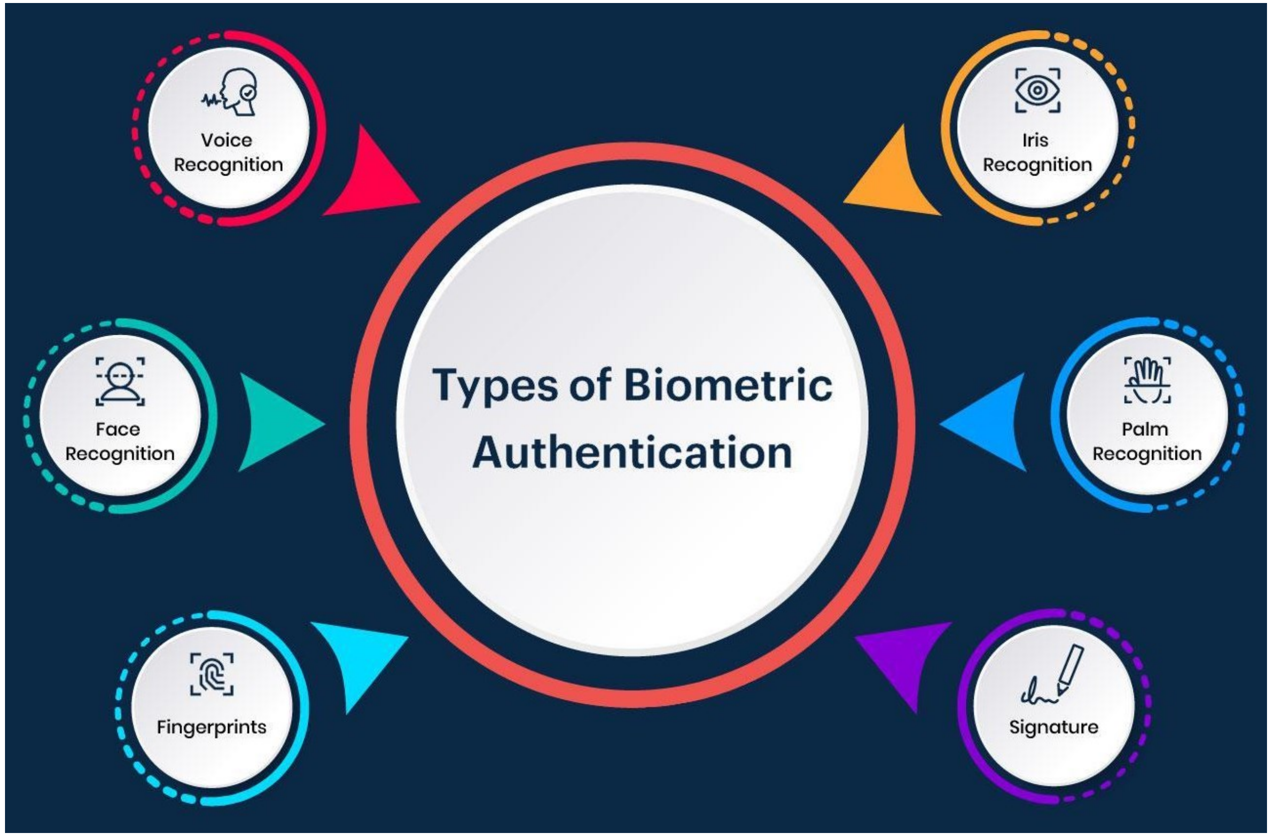
1. One of the main troubles with passwords is that most users either don't understand how to make strong and memorable passwords or underestimate the need for security. A test of password entropy predicts how difficult a given password would be to crack through guessing, brute force cracking, dictionary attacks or other common methods.

2. Social engineering is a major threat to password-based authentication systems. To decrease its social engineering attack surface, an organization must train all users, from management to staff. Password strength means nothing  if an attacker tricks a user into divulging it. Even IT staff, if not properly trained, can be exploited with invalid password-related requests. All employees must be aware of phishing tactics, where false emails and forged websites may be used to acquire sensitive information from an unwitting recipient.

3. Other threats, such as Trojans may also come in email messages.

In short, passwords are one of the most easily stolen/ broken types of authentication.

# STRONG AUTHENTICATION



However, single-factor authentication isn't necessarily weak. Many biometric authentication methods, for example, are strong when properly implemented.

Multiple challenge response questions can make for secure SFA authentication when properly implemented. Biometrics can often make for secure SFA so long as the right kinds and implementations are chosen. Retina scans, finger vein scans and voice recognition are good candidates. One must be doubly sure about the biometric scanner and its implementation when it is a standalone SFA solution rather than one component of MFA.

However, biometric verification systems may require a significant outlay for enterprise deployment. Depending on the degree of security required, it may be preferable to implement multifactor authentication (MFA).

Basic authentication often entails the transmission of a name (username or system name), and the password, which can be easily stolen and compromised if they’re transmitted unprotected across the network. Here are some of the ways to increase the strength of Basic authentication:

1. Use digest authentication—hash or encrypt the password prior to transmission
2. Use pass phrases (longer passwords) and set minimum password lengths
3. Enforce the usage of diverse character sets that include alpha-numeric, special characters, and mixed-case passwords that are not in a dictionary
4. Add security to the connection wherein the password is not transmitted in the clear across the network, such as TLS/SSL
5. Do not store passwords in plaintext in whatever mechanism is used—database, file system, directory

# MULTIFACTOR AUTHENTICATION (MFA)

Multi-factor authentication is the use of a combination of authentication methods to validate identity. The most commonly used description of multi-factor authentication is the use of information that is known only by the person, combined with something in his or her possession. These are typically:

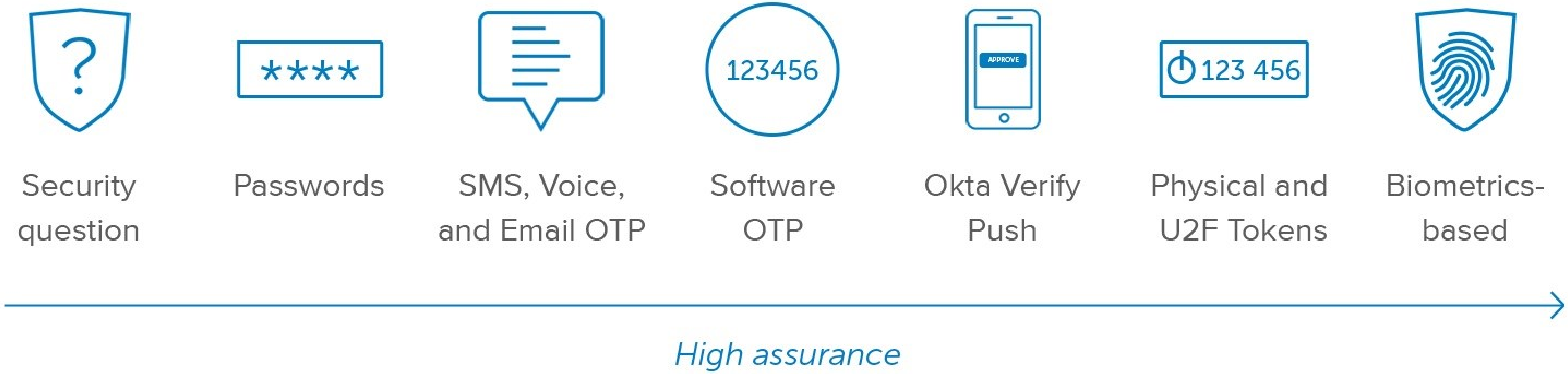
1. The name and password
2. Some form of token

A token is a hardware component that is used during the authentication process; it typically provides another piece of information that cannot be ascertained without physical control of the token. Different types of tokens used in multi-factor authentication are:

- Smart cards
- One-time password/phrases
- Single-use PINs or pseudo-random numbers
- Biometric information

Multi-Factor Authentication (MFA) leverages 2 or more independent factors to grant user access to a system. In typical scenarios, MFA methods leverage at least 2 or 3 of the following categories.

1. Something you know - a password or a pin
2. Something you have - mobile phone or a security token
3. Something you are - fingerprint or FaceID
4. Something you do - typing speed, locational information etc.



Security question

Passwords

SMS, Voice, and Email OTP

Software OTP

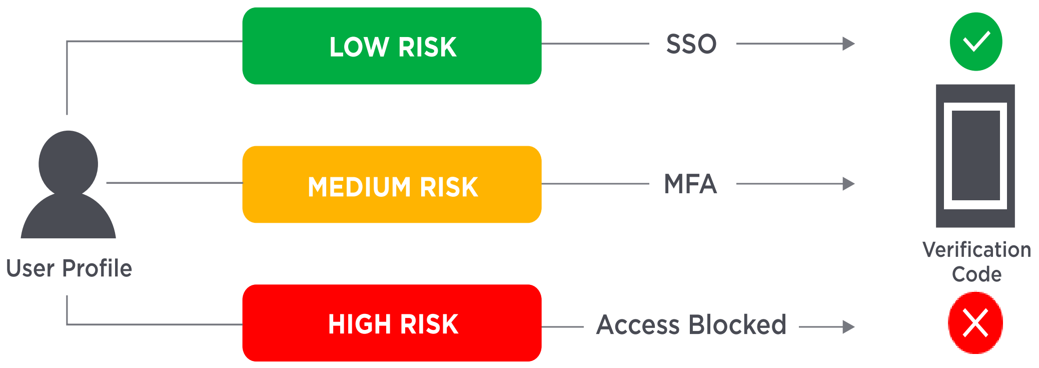
Okta Verify Push

Physical and U2F Tokens

Biometrics-based

*High assurance*

# ADAPTIVE AUTHENTICATION OR RISK-BASED AUTHENTICATION



Another subset of MFA is Adaptive Authentication also referred to as Risk-based Authentication. Adaptive Authentication analyzes additional factors by considering context and behavior when authenticating and often uses these values to assign a level of risk associated with the login attempt. For example:

- From where is the user when trying to access information?
- When users are trying to access company information?
- During user’s normal hours or during "off hours"?
- What kind of device is used?
- Is it the same one used yesterday?
- Is the connection via private network or a public network?

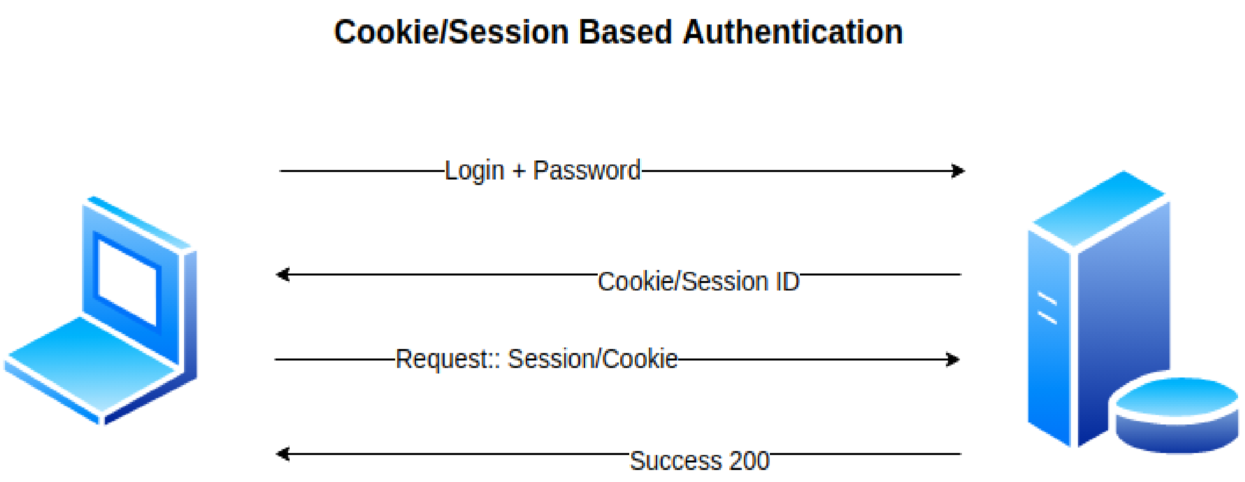
The risk level is calculated based upon how these questions are answered and can be used to determine whether or not a user will be prompted for an additional authentication factor or whether or not they will even be allowed to log in. Thus another term used to describe this type of authentication is risk-based authentication.

# CRYPTOGRAPHIC AUTHENTICATION

This includes the following forms:

- Public Key Authentication
- Digital Signatures
- Message Authentication Code

# SESSION-BASED AUTHENTICATION

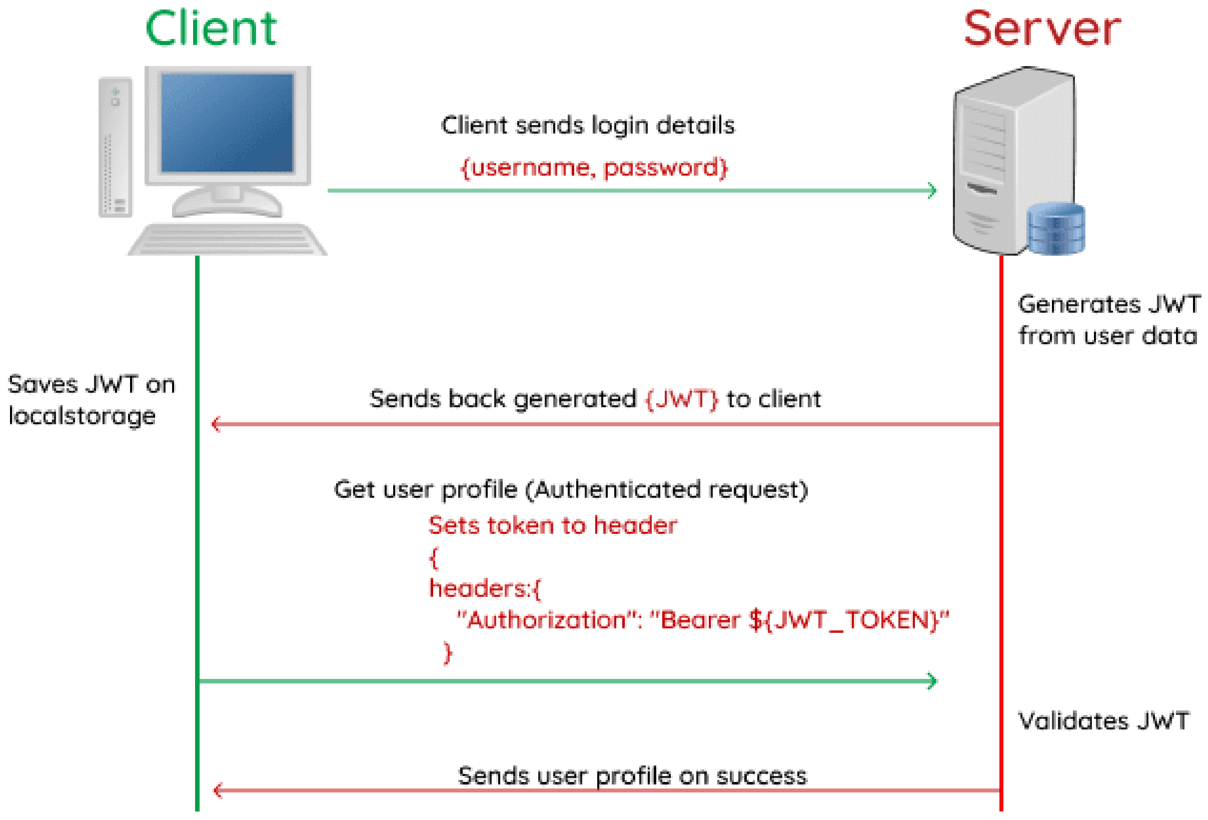


In a session-based authentication, when a user logs in, the server stores the session information (client state) in the server memory and a session id is sent to the client browser - which is then stored using (most likely) cookies on the browser. On subsequent requests to server, the server compares the session id with the session information stored in its memory, and sends the corresponding response. Once a user logs out, the session is destroyed from both client and server side. Sessions are typically stateful on the server side (i.e. they are stored on the server).

**Cookie-based Authentication**

Cookie-based authentication is basically a type of session-based authentication, in which session data are stored using cookies. Considering HTTP is a stateless protocol, cookies are used to store information concerning the user on the browser - incase of subsequent requests to server. Adapting a cookie-based authentication in web applications can prevent your site against XSS (Cross Site Scripting) attacks, as there some security flags available using cookies to protect user data. Additional flags can also be used to prevent a site against CSRF (Cross Site Request Forgery) attacks.

# TOKEN-BASED AUTHENTICATION


Client
Server
Client sends login details
{username, password}
Generates JWT from user data
Saves JWT on localstorage
Sends back generated {JWT} to client
Get user profile (Authenticated request)
Sets token to header
{
headers:{
"Authorization": "Bearer ${JWT_TOKEN}"
}
Validates JWT
Sends user profile on success

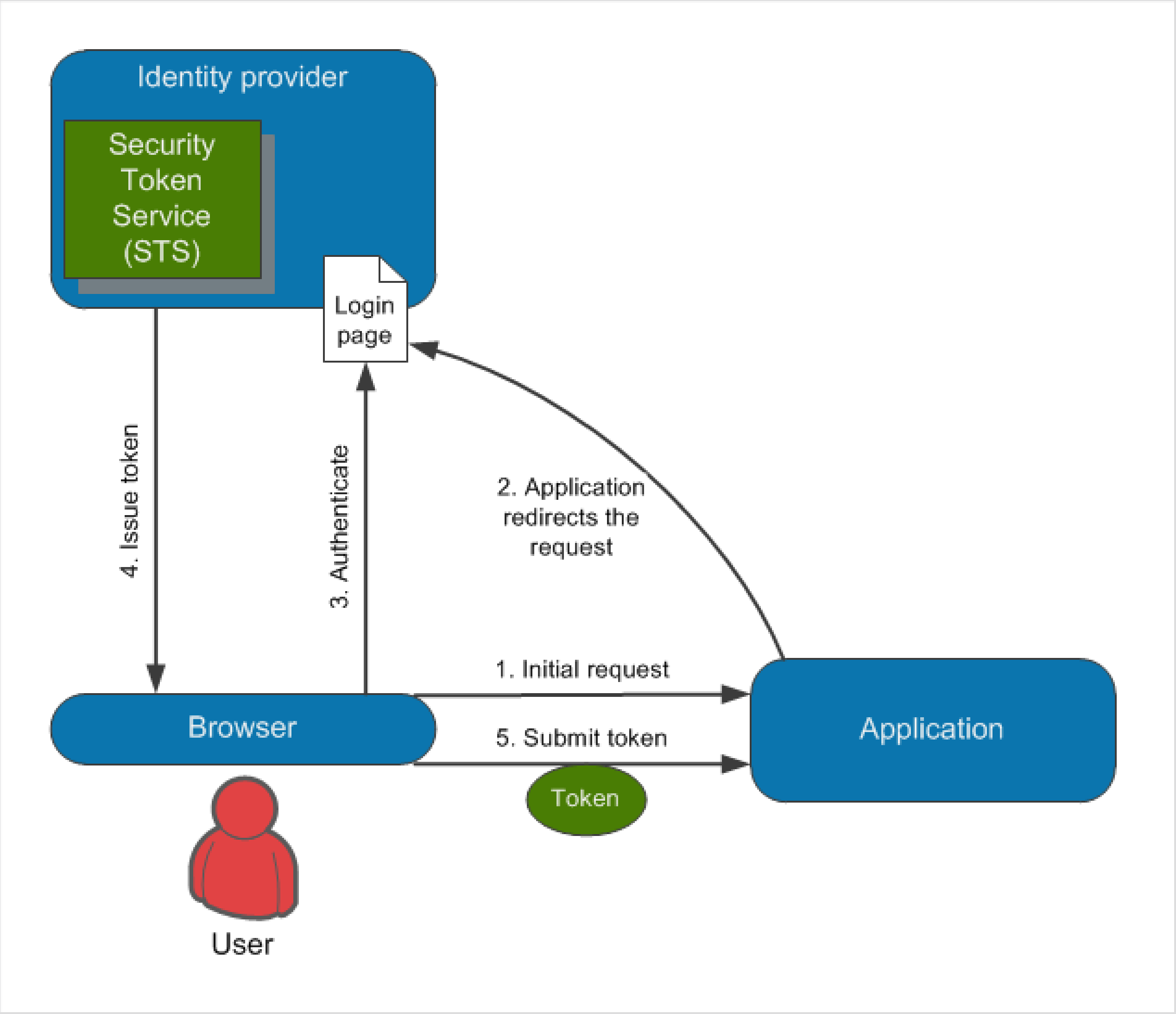
Token based authentication

There's much preference for token based authentication in web applications, due to its scalability and compatibility with mobile applications.

In a token based authentication, the client data is encrypted in a JSON Web Token (JWT) by the server, and sent back to the client. The JWT is usually stored in local storage, but can also be stored in a cookie. On subsequent requests, the web token is first verified by the server, followed by a corresponding response from the server.

One interesting thing about token based authentication is its statelessness (client state NOT stored on server), compared to a stateful (client state stored on server) approach - like in session based authentication. Once the user logs out, token is destroyed from both client and server side.

# CLAIMS BASED AUTHENTICATION

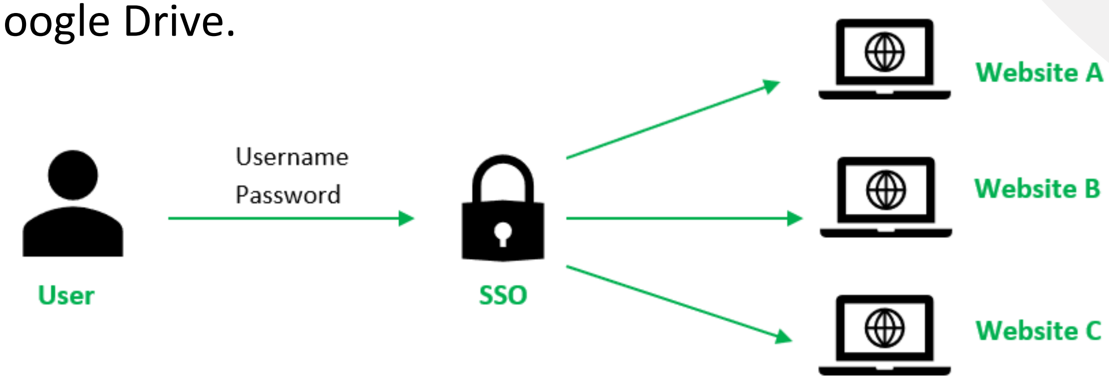


This type of authentication is similar to a token-based authentication. Here users are authenticated on external systems, called identity providers. These identity providers then issue a security token - which contains information about authenticated user. These informations are referred to as claims. Claim is a piece of information that describes a given identity (user) in regards to authorization. When a user requests access to an application, the application redirects him/her to the authentication page of chosen identity provider. After successful authentication, the user is redirected back with some information. The application then requests the user to be validated by external system, and upon successful validation, the user is granted access to the application.

# SINGLE SIGN ON (SSO)

Single Sign On (SSO) is an authentication scheme where users can securely authenticate and gain access to multiple applications and websites by only logging in with a single username and password.

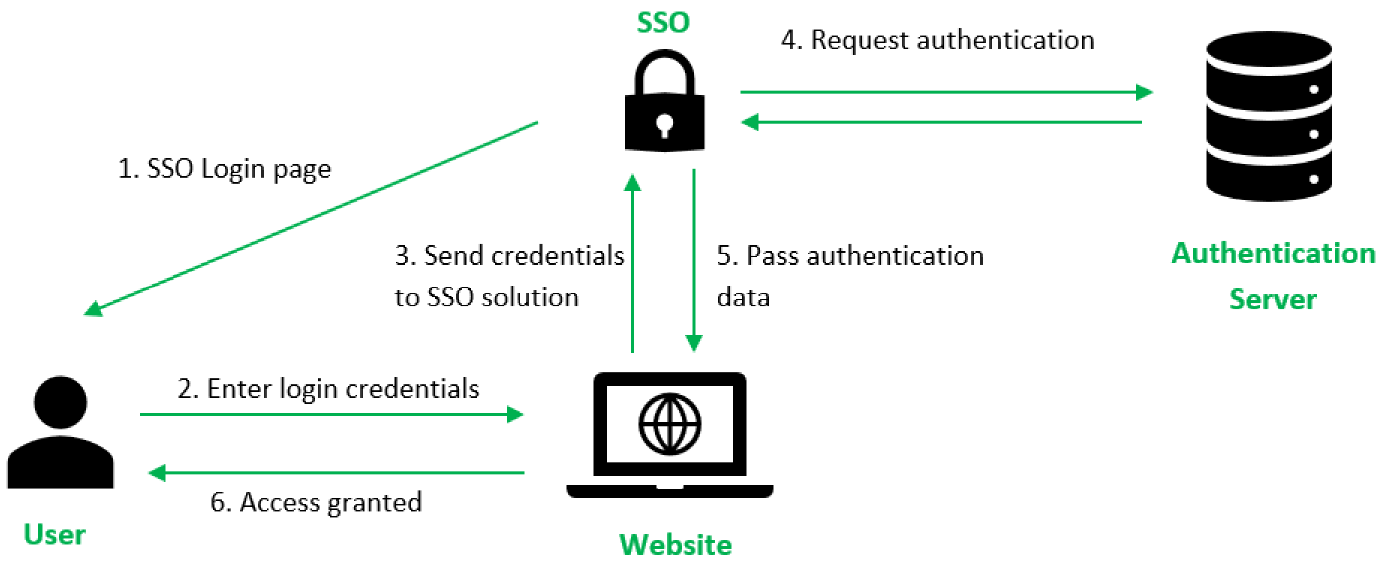
For example, logging in to your Google account once will allow you to access Google applications such as Google Docs, Gmail, and Google Drive.



Without SSO solution, the website maintains a database of login credentials – username and passwords. Each time the user login to the website, it checks the user’s credentials against its database and authenticates the user.

With the SSO solution, the website does not store login credentials in its database. Instead, SSO makes use of a shared cluster of authentication servers where users are only required to enter their login credentials once for authentication. With this feature of one login and multiple access, it is crucial to protect login credentials in SSO systems.
Hence it is highly recommended to integrate SSO with other strong authentication means such as smart tokens or one-time passwords to achieve multi-factor authentication.

# HOW DOES SSO WORK?


SSO
4. Request authentication
1. SSO Login page
3. Send credentials to SSO solution
5. Pass authentication data
Authentication Server
2. Enter login credentials
6. Access granted
User
Website

1. User enters login credentials on the website and the website checks to see if the user has already been authenticated by SSO solution. If so, the SSO solution would give the user access to the website. Otherwise, it presents the user with the SSO solution for login.
2. The user enters username and password on the SSO solution.
3. The user's login credentials are sent to SSO solution.
4. The SSO solution seeks authentication from the identity provider, such as an Active Directory, to verify the user's identity. Once the user's identity is verified, the identity provider sends a verification to the SSO solution.
5. The authentication information is passed from the SSO solution to the website where the user will be granted access to the website.
6. Upon successful login with SSO, the website passes authentication data in the form of tokens as a form of verification that the user is authenticated as the user navigates to a different application or web page.

**Advantages of SSO :**
These are advantages for users, for businesses.

**For Users –**
•Risk of access to 3rd party sites are mitigated as the website database do not store the user’s login credentials.
•Increased convenience for users as they only need to remember and key in login information once.
•Increased security assurance for users as website owners do not store login credentials.

**For Businesses –**
•Increase customer base and satisfaction as SSO provides lower barrier to entry and seamless user experience.
•Reduce IT costs for managing customer’s username and passwords.

**Disadvantages of SSO :**
•Increased security risk if login credentials are not securely protected and are exposed or stolen as adversaries can now access many websites and applications with a single credential.
•Authentication systems must have high availability as loss of availability can lead to denial of service for applications using a shared cluster of authentication systems.