Whitepaper

Payatu

Payatu whitepaper

Unlocking the World of Authentication Methods and Their Attacks

Author

MOHAMED ALTHAF
Associate Security Consultant

Contributor

TANVI TIRTHANI
Senior Content & Media Strategist

Table of Contents

1.0

Introduction

This whitepaper aims to provide valuable insights to developers, security professionals, and organizations through an in-depth review of authentication methods and their corresponding attacks. This allows them to strengthen their safety measures and protect against unauthorized access and data breaches.

The whitepaper thoroughly examines several authentication methods, including Basic and Form-based Authentication, Token-based Authentication, OAuth, and SSO. It indicates any potential security dangers or attacks related to each authentication, such as MITM, brute force, token theft, and XML injection.

The whitepaper also provides realistic mitigation and remediation points which developers, security professionals, and organizations can implement to improve the security of their authentication procedures and protect against vulnerabilities through intense testing.

2.0

Authentication

Authentication is the process of verifying the identity of a user or system before granting access to the resources or services. By verifying the identity, authentication helps prevent unauthorized access, data breaches, fraud and other security threats.

There are several distinct types of authentication methods that can be used, depending on the level of security. Here are some commonly used types of authentication mechanisms:

1. Basic Authentication
2. Form-based Authentication
3. Multi-factor Authentication
4. Token-based Authentication
5. Single Sign On - SAML
6. NTLM
7. LDAP
8. OAuth

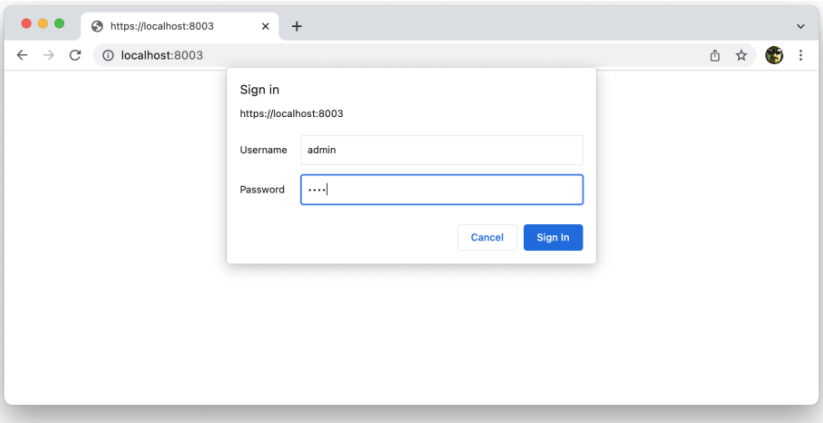
3.0

Basic Authentication and Form-Based Authentication

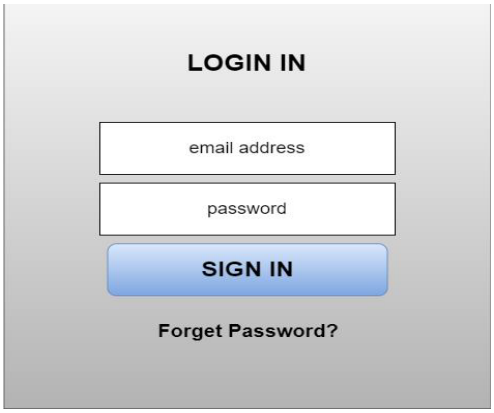
Basic Authentication and Form-based authentication are two alternative approaches for user authentication in web applications.

In basic authentication, Users input their username and password, which are base64-encoded and sent as part of the HTTP request's authorization header, provided in the HTTP protocol.

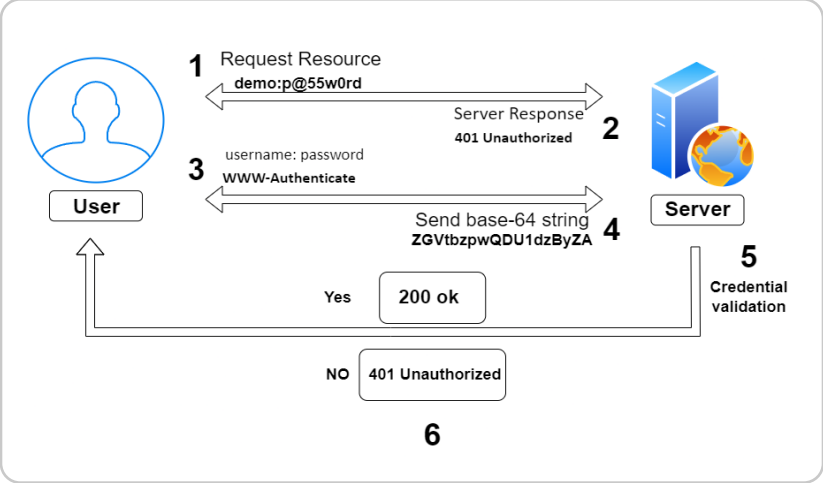
For instance, a client may send the message "Authorization: Basic ZGVtbzpwQDU1dzByZA==" to sign in as "demo" and the password "p@55w0rd." Since basic authentication does not employ cookies, each request must have the credentials in the header to maintain authentication.



Whereas form-based authentication sends the username and password to the server usually through a POST HTTP request using HTML form fields. The server then makes a comparison between the supplied password and the database's password storage. When they match, the server generates a session that is associated with an identifier transmitted between the client and server with each HTTP request. This session identifier is saved in a cookie. The server often redirects to a login page if the cookie is invalid, or the user logs out. With this approach, the authentication may be maintained through sessions and cookies and is given additional flexibility.



3.1 How Basic Authentication Works?



01

Request Resource

When a client (such as a web browser or a command-line tool) wants to access a protected resource on a server, it sends an HTTP request to the server.

02

Server Response

The server responds to the client's request with a 401 unauthorized status code, indicating that authentication is required to access the requested resource. The server also includes a WWW-Authenticate header in the response to specify that basic authentication is the method required.

03

Client Provides Credentials

Upon receiving the 401 responses with the WWW-Authenticate header, the client prompts the user for their username and password or retrieves it from its storage. It then constructs a string in the form `username:password`.

04

Credentials Encoding

The client base64-encodes the `username:password` string. Base64 encoding is not a form of encryption; it's just a way to represent binary data (the username and password) in a text format. This encoded string is typically referred to as the 'authentication token'. For example, if the username is 'user' and the password is 'pass', the authentication token would be 'dXNlcjpwYXNz' after base64 encoding.

05

Credentials Validation

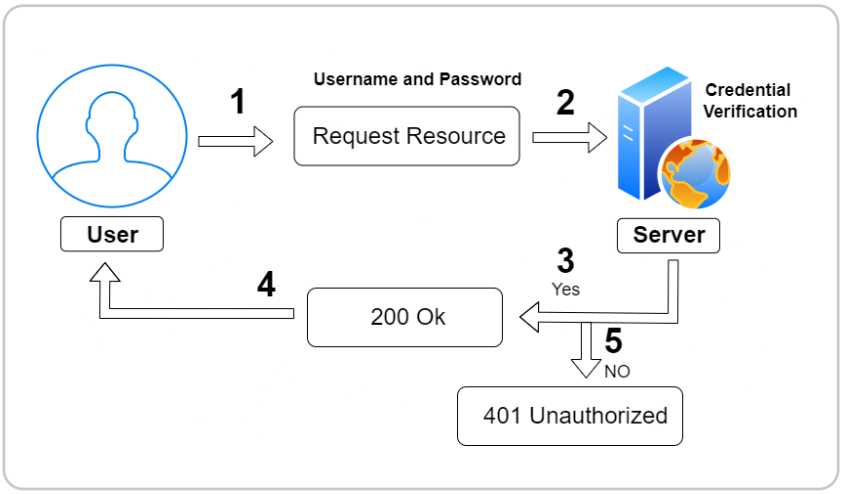
The server receives the HTTP request with the 'Authorization' header. It decodes the base64-encoded token to extract the username and password. It then checks the provided credentials against its authentication database to verify if they are valid.

06

Access Granted or Denied

If the credentials are valid, the server allows access to the requested resource and returns the appropriate response (e.g., status code 200 OK). If the credentials are invalid, the server responds with a 401 Unauthorized status code again, prompting the client to retry with valid credentials.

3.2 How Form-based Authentication Works?



STEP 01

Request Resource

The client (e.g., a web browser or an application) sends an HTTP request to access a protected resource on the server. The server receives the request and identifies that the resource requires authentication.

STEP 02

Server Authentication and Credentials Verification

Upon receiving the request, the server verifies the received credentials against the user database or external authentication service. The server compares the received username and password against its stored records or a user database to authenticate the user's credentials.

STEP 03

Credentials Validation

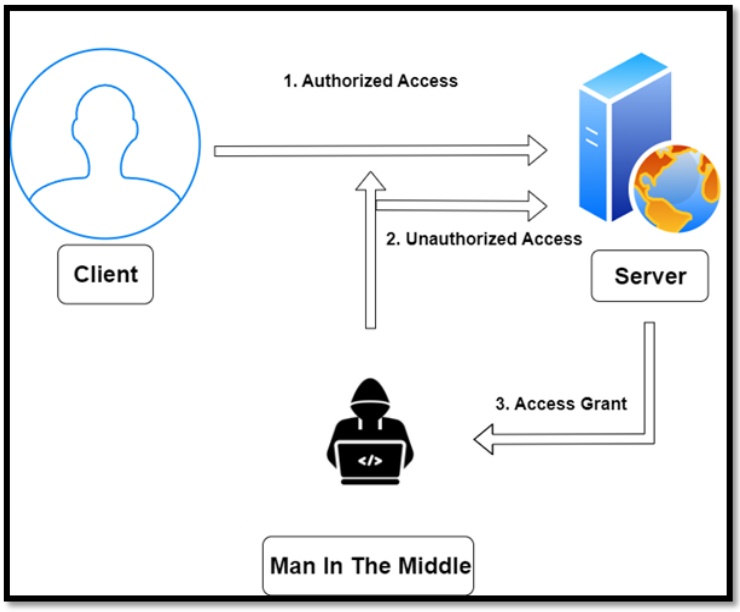
If the provided credentials match, the server grants access (200 OK) to the requested resource. Otherwise, it returns a 401 unauthorized error.

3.3 Attacks in Basic and Form-based Authentication

Below are some common attacks associated with Basic and Form-based Authentication:

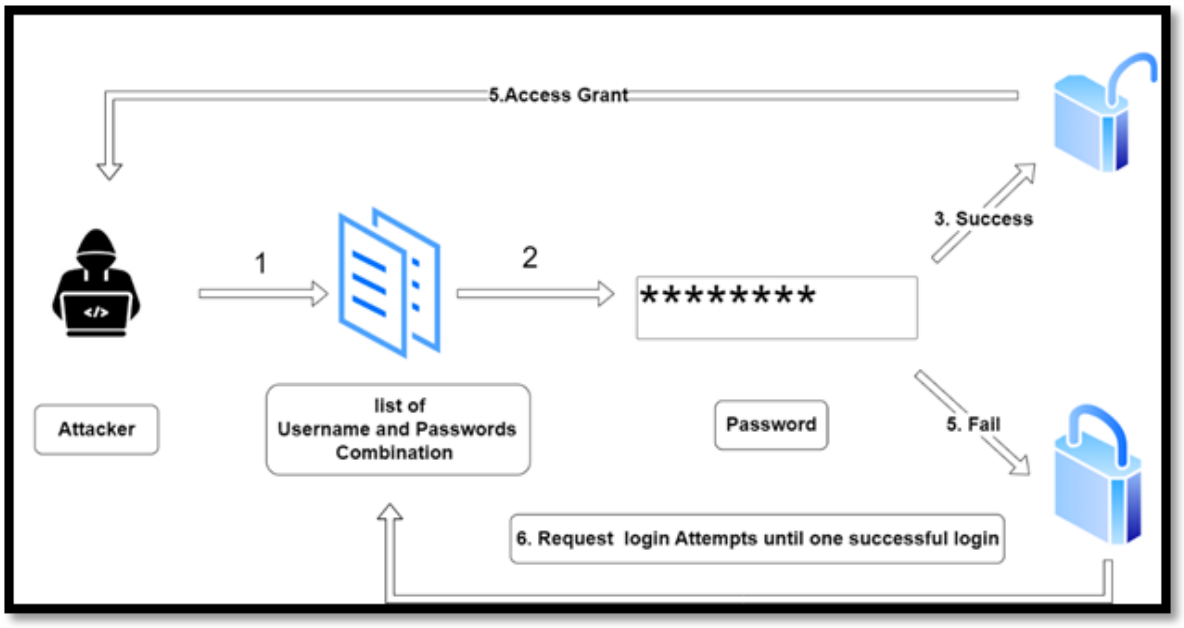
3.3.1 MITM Attack

A man-in-the-middle (MitM) attack is an attack where communications between two parties is intercepted, often to steal login credentials or personal information. The attacker sits in between the victim and a legitimate host by either passively listening in on the connection or by intercepting the connection, terminating it and setting up a new connection to the destination.



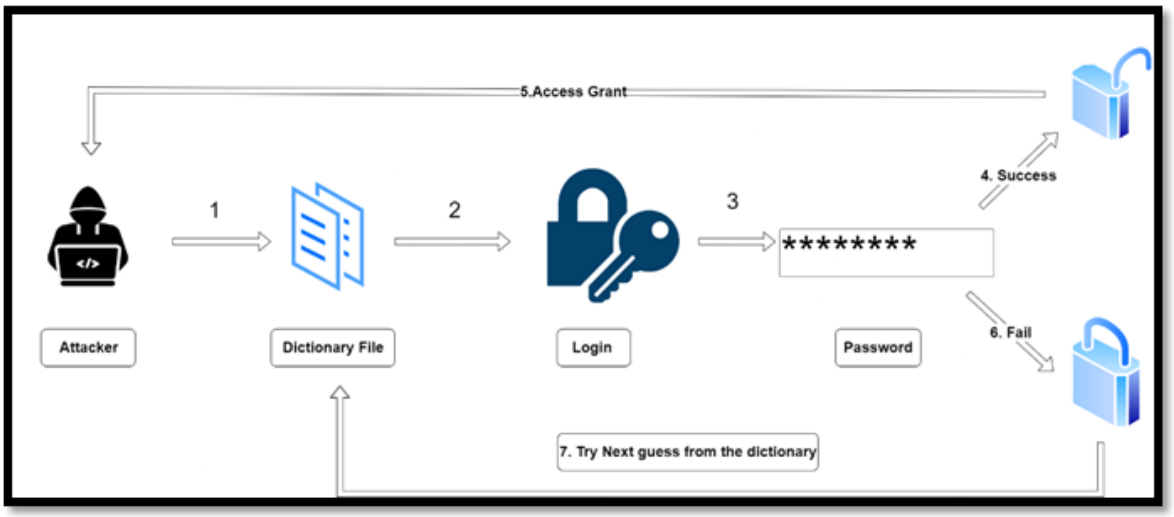
3.3.2 Brute Force Attack

Attackers can attempt brute force by systematically trying different combinations of usernames and passwords to guess valid credentials. Since most of the authentication mechanisms don't have any protection against multiple login attempts, they become vulnerable to such attacks. Brute force attacks are especially dangerous because they can try thousands and thousands of combinations. Once they find a single correct combination, they can easily break into a network and wreak havoc. Brute force attacks remain one of the most common types of cyberattacks today.



3.3.3 Dictionary Attack

Like brute force attack, a Dictionary attack involves using a list of common passwords or previously compromised passwords and trying to gain access to the server. Attackers can automate this process by using readily available password-cracking tools such as Hydra, John the riper, etc.



3.3.4 Weak password/Default passwords

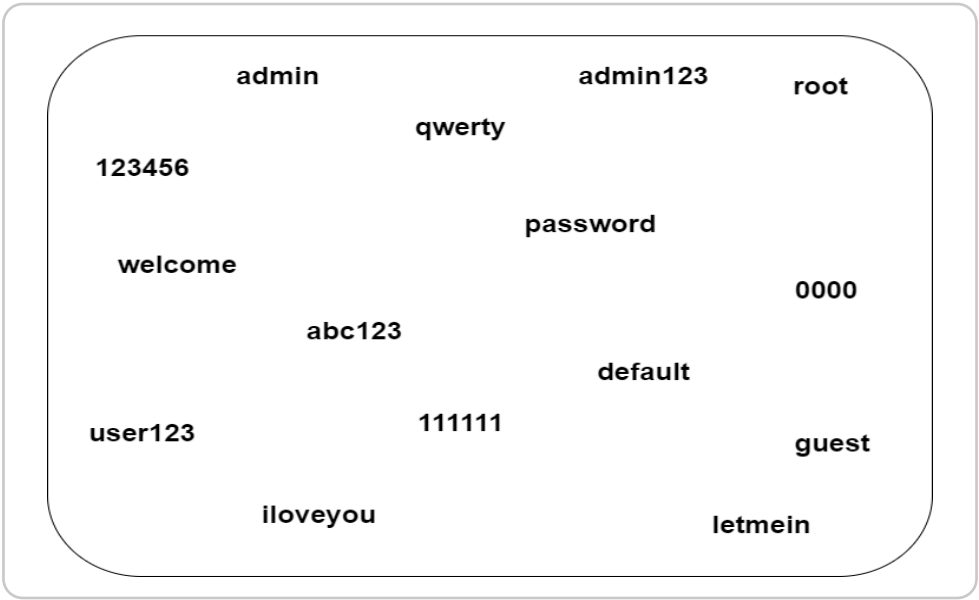
One of the common attacks, like brute force, is due to the use of weak passwords or default passwords. Attackers often exploit the fact that users tend to choose weak or easily guessable passwords or may leave the default passwords unchanged. These passwords are typically short, simple, common, and easy to guess.

By successfully guessing or cracking weak or default passwords, attackers can bypass authentication, and gain access to protected resources and sensitive information.

There are some default passwords for example:

{Username: admin, Password: admin}

{Username: Ravi, Password: ravi@123}



3.4 Mitigation Points

To mitigate attacks against Basic and Form-based Authentication, implement the following remediations:

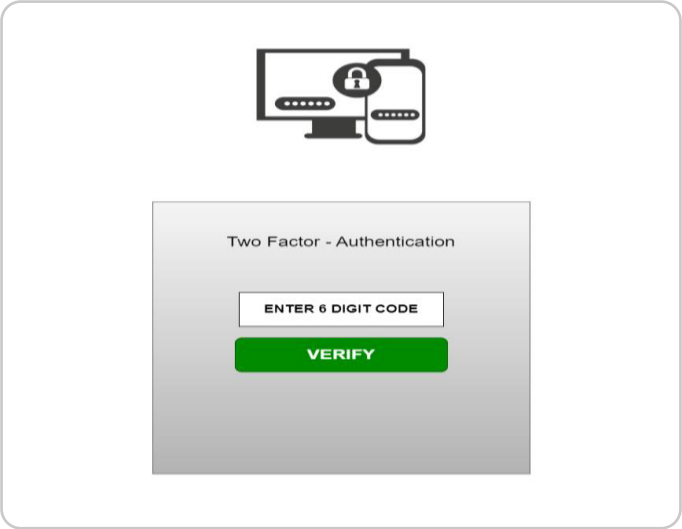
1. **Use Strong Passwords:** Encourage users to create strong, unique passwords that are resistant to dictionary attacks or guessing. Enforce password complexity rules and educate users about password best practices. Some policies suggest or impose requirements on what type of password a user can choose, such as:
    - The use of both upper-case and lower-case letters (case sensitivity)
    - Inclusion of one or more numerical digits
    - Inclusion of special characters, such as @, #, $
    - Prohibition of words found in a password blacklist
    - Prohibition of words found in the user's personal information
    - Prohibition of use of company name or an abbreviation
    - Prohibition of passwords that match the format of calendar dates, license plate numbers, telephone numbers, or other common numbers
2. **Account Lockouts and Rate Limiting:** Implement mechanisms that limit the number of failed authentication attempts within a specific time. This prevents or slows down brute force attacks by temporarily locking or suspending accounts after a certain number of unsuccessful login attempts.
3. **Multi-Factor Authentication (MFA):** Implement multi-factor authentication, which adds an extra layer of security by requiring users to provide additional authentication factors (e.g., a unique code from an authenticator app or SMS verification) in addition to their username and password.
4. **Secure Communication Channels:** Use secure connections like HTTPS to encrypt the communication between the client and the server. This prevents credential interception (MITM) and unauthorized access to sensitive information during transit.

4.0

Multi-Factor Authentication

In multi-factor authentication (MFA), the user is required to provide two diverse types of authentication factors to verify their identity. This typically involves combining something the user knows (e.g., password) with something the user has (e.g., a verification code sent to their mobile device, authenticator app, etc)

The goal of MFA is to add an extra layer of security by requiring both factors for authentication, making it more difficult for attackers to compromise an account or system.

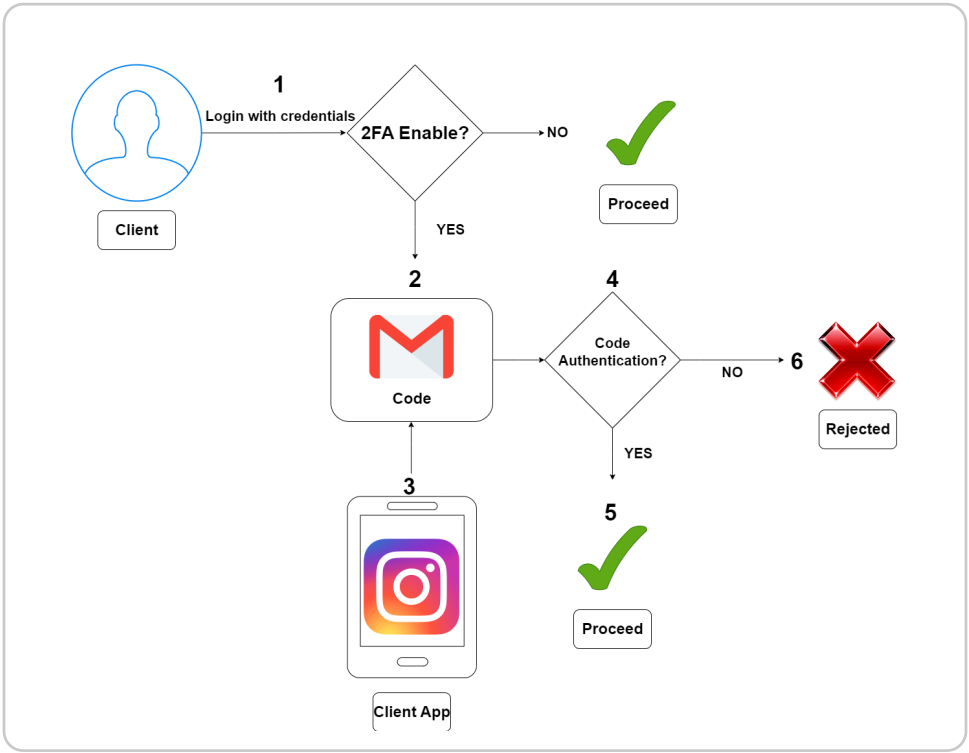


4.1 How Multi Factor Authentication Works?

The Two-factor authentication (2FA) process begins with users entering their login credentials, consisting of a username and password.

The system then checks whether 2FA is enabled for the user's account. If 2FA is not enabled, the user is allowed to proceed into the application. However, if 2FA is enabled, the system requests a second factor of authentication, which can take various forms, such as a one-time code, a biometric scan (like a fingerprint or facial recognition), or another method.

Once the user provides the required second factor, the system verifies its accuracy. If the second factor is correct, the user gains access to the application; otherwise, if it is incorrect, the authentication request is rejected, and access is denied.

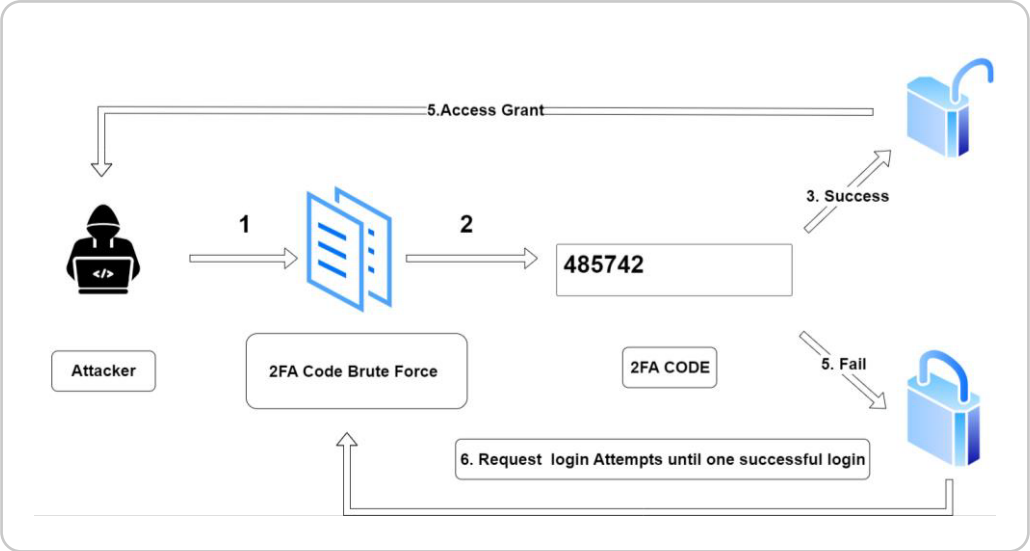


4.2 Attacks & Bypass techniques in MFA

Below are some common attacks associated with Multi-factor Authentication:

4.2.1 MFA/2FA Brute Force Attack

Like brute forcing a password, an MFA code can also be brute forced using a similar approach.



4.2.2 MFA/2FA Bypass

The MFA code can be bypassed via several methods as follows:

1. **Response/Status code manipulation**
    - If the response contains Success:false, change the value to Success:true and it can bypass the 2FA.
    - If the Response Status Code is like 401 Unauthorised, 403 Forbidden, etc, by changing the response Status Code to "200 OK" it bypasses the 2FA.
2. **Brute force the 2FA code**
    - One can attempt to brute force the 2FA code by systematically trying different combinations by trying until the actual valid code is found.
3. **OTP leak in response**
    - Observe the request and response of the OTP page, and check if the OTP is leaking in the response or not.

4. **Use User A's OTP in User B's Account**
    - Use the valid OTP of User A's account into User B's account and check if it bypasses the 2FA or not.
5. **Change the value of Referrer header to/dashboard or/user_profile**
    - Observe the response and change the value of the Referrer or Location header to /dashboard or /user_profile or any other endpoint that comes after the 2FA check.
6. **Try Backup/Recovery codes**
    - Use same techniques used on 2FA such as Response/Status Code Manipulation, brute-force, User A to User B, etc. to bypass the backup code.
7. **Response manipulation**
    - Remove 2FA code:
    - Remove the 2FA code in the request and observe it bypasses 2FA or not
    - Remove parameter & value
    - Remove both 2FA parameter and value, check it bypasses 2FA or not
    - DELETE header
    - Delete the whole 2FA code check request, and observe it bypasses 2FA or not

4.3 Mitigations Points

01

App-Based Authenticators

Use authenticator apps like Google Authenticator, Authy, or Microsoft Authenticator. These apps generate time-based one-time passwords (TOTPs) for 2FA, which are more secure than SMS codes.

02

Backup Codes

Generate and securely store backup codes provided by the service in case you lose access to your authenticator app or device.

03

Biometric 2FA

Utilize biometric authentication (fingerprint, face recognition) as one of the factors in addition to your password. However, biometrics should not be the sole factor due to potential privacy concerns and vulnerabilities.

5.0

Token-based Authentication

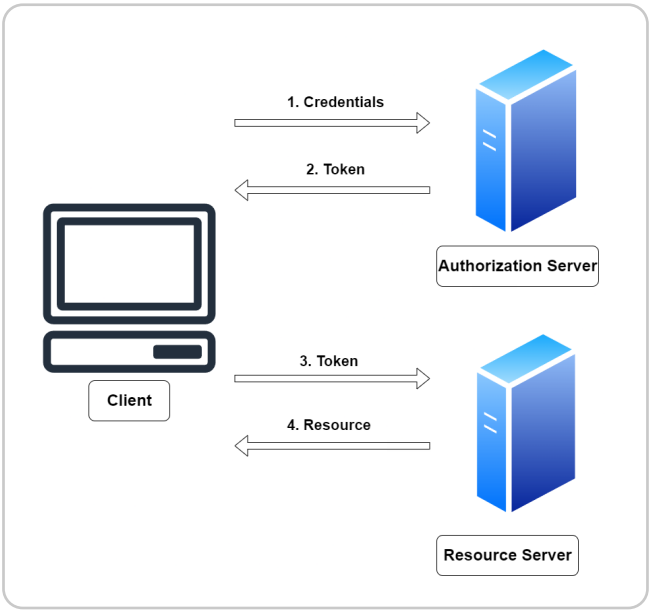
Token-based authentication simplifies the authentication process for known users. It is a type of authentication in which tokens are used to verify a user's identity. A token is a little piece of data that is used to represent the identity of the user.

To begin with, the user sends a request to the application server, containing a username and password. The server then validates them based on values registered in its credentials database.

If the credentials are confirmed, the server responds with an authentication token (which is also stored in the database). When the same user sends requests to access secured resources in the future, the requests can be authorized with the authentication token, rather than the username and password. The server validates the token against the registered token in the database and grants access.

During the life of the token, users can access the website or application via the issued token, rather than having to re-enter credentials each time they go back to the same webpage, application, or any resource protected with that same token. Authentication can be carried out using various types of tokens like JSON Web Tokens (JWT).

5.1 How Token-based Authentication works?



01 Users Authentication

The user log into the application by entering their credentials. This could be a login and password or another type of identification, such as a fingerprint or facial recognition.

02 Token Generation

The application transmits the user credentials to the authentication server. The authentication server oversees validating the user's identity and providing a token. The token is a string of data that represents the identity of the user.

03 Token Distribution

The token is sent back to the application by the authentication server. The token is saved in the user's browser by the application.

04 Token Validation

The user can now access the application's protected resources without having to enter their credentials again. The application sends the token to the server when the user requests a secured resource.

05 Access Granted or Denied

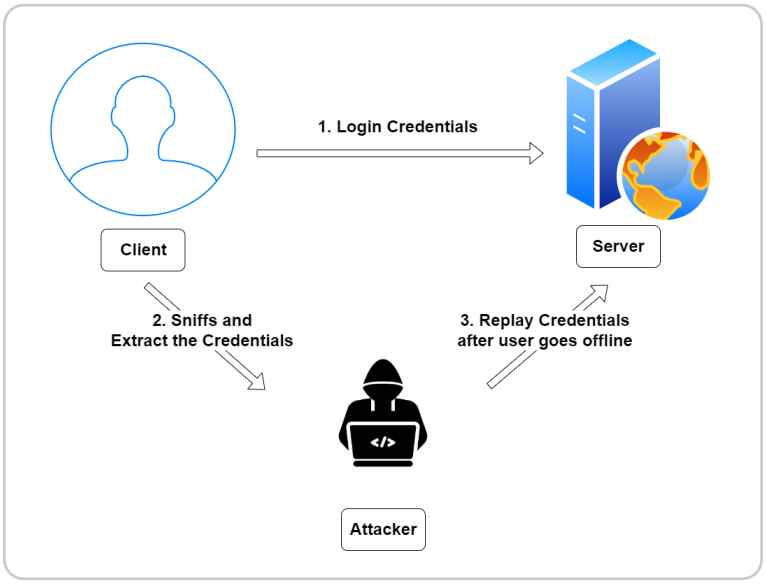
If the token is valid, the server provides access to the resource. The server declines access to the resource if the token is invalid.

5.2 Attacks in Token-based Authentication

Token-based authentication is vulnerable to attacks, just like any other authentication system. Following are some examples of token-based authentication attacks:

5.2.1 Replay Attack

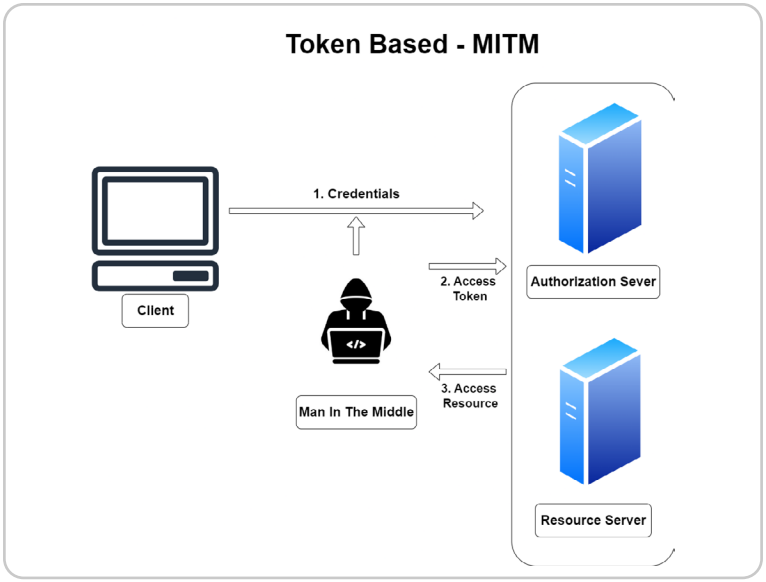
In a replay attack, a valid token is intentionally used by the attacker to gain access to a system or resource without authorization. Token-based authentication systems rely on tokens, which are frequently transient, to confirm a user's or an entity's identity. These tokens, which are often created following successful authentication, allow users to access restricted resources without having to repeatedly enter their login information (such as a username and password).



For example, your device exchanges credentials and session token with a server. A hacker listens in and steals the session token and later sends a request to access the protected resource containing the stolen token. The server believes that it's speaking with the victim (the user's information you stole) as each request contains the session token. The hacker can then do anything that the victim can do on the server.

5.2.2 Token-based MITM Attack

In a token-based MITM (Man-in-the-Middle) attack, the attacker intercepts on communication between a client and a server with a view of taking advantage of the authentication tokens. Tokens, which serve as authentication proof and are often short-lived, can be vulnerable while being transmitted. Attackers have the ability to listen in on conversations, seize tokens, and utilize them in malicious ways. Additionally, they have the ability to modify or replay tokens, which could lead to illegal access to a user's account or system resources. When tokens are not effectively safeguarded while in transit, this kind of assault could put at risk the security of token-based authentication systems.



5.2.3 Token Theft Attack

A token theft attack in token-based authentication happens when an attacker gets unauthorized access to authentication tokens, frequently through intercepting them during transmission or taking advantage of weaknesses in the authentication process.

By using these tokens, which serve as identification documents for users, an attacker may impersonate a valid user and get access to confidential information or carry out malicious deeds on their behalf. Implementing strong security measures, such as token encryption, secure token storage, and token rotation, is essential to preventing attacks on token theft. This will also help to decrease the impact of any potential theft. Incorporating multi-factor authentication (MFA) can also add an additional layer of security by requiring verification processes in addition to the token itself, making it harder for attackers to use stolen tokens for their own gain.

5.3 Mitigation Points

Below are important token-based authentication mitigation points:

Token Encryption

To prevent unauthorized access and interception, ensure that tokens are transmitted and stored in an encrypted format.

Token Expiry

Set short token expiration durations and develop a system to refresh tokens frequently. This lowers the possibility of token theft and unauthorized access.

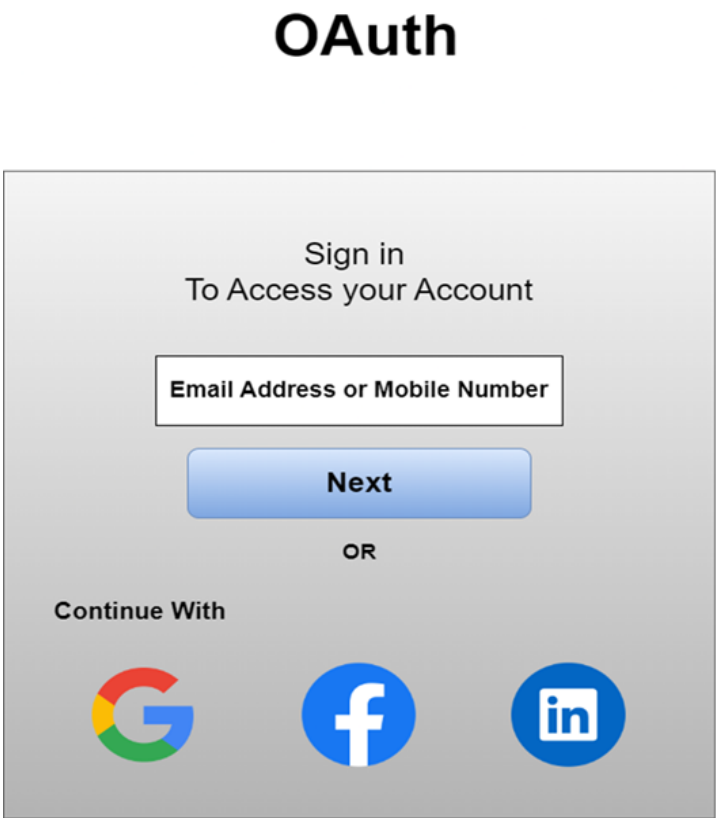
Secure Token Storage

To prevent theft and limit cross-site scripting (XSS) attacks, save tokens securely on the client side using HTTP-only cookies or browser local storage.

6.0

OAuth

OAuth is an Open Authorization mechanism that allows users to grant third-party applications access to their resources without sharing their passwords. Many famous websites and applications, like Google, Facebook, and Twitter, use OAuth.

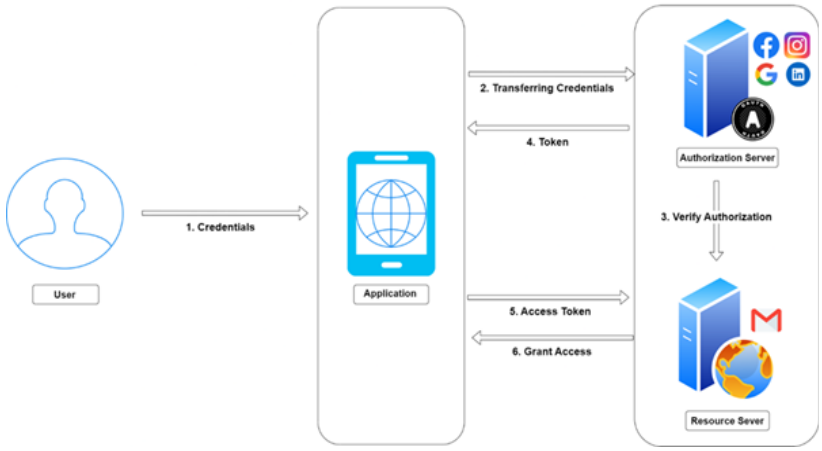


Tokens are used in OAuth. A token is a short piece of data that represents a user's ability to access a resource. When a user wants to provide access to their resources to a third-party application, they must first authenticate with the resource owner (e.g., a website or application). After that, the resource owner generates a token and sends it to the third-party application. The token can then be used by the third-party application to gain access to the user's resources.

6.1 How OAuth works

Tokens are used in OAuth. A token is a short piece of data that represents a user's ability to access a resource. When a user wants to provide access to their resources to a third-party application, they must first authenticate with the resource owner (e.g., a website or application). After that, the resource owner generates a token and sends it to the third-party application. The token can then be used by the third-party application to gain access to the user's resources.

OAuth Work



User Authentication

STEP 01

The user enters their credentials and logs in to the Application. This could be a login and password or another type of identification, such as a fingerprint or facial recognition.

The application transmits the authentication server the user's credentials. The authentication server oversees validating the user's identity and providing a token. The token is a little bit of data that represents the identity of the user. It is usually encrypted to prevent unauthorized access.

STEP 02

Token Generation

Token Distribution

STEP 03

The token is sent back to the application by the authentication server. The token is saved in the user's browser by the program.

Token Validation

STEP 04

The user can now access the application's protected resources without having to enter their credentials again. The application sends the token to the server when the user requests a secured resource.

Token Authorization

STEP 05

If the token is valid, the server validates it and provides the user access to the resource. The server declines access to the resource if the token is fake.

6.2 Attacks in OAuth

OAuth is vulnerable to weakness in security, just like any other system. Several attacks against OAuth solutions can be tried, and developers and security experts must be aware of these possible vulnerabilities.

6.2.1 Access token leakage

Access token leakage is the unintentional sharing of access tokens, which are sensitive pieces of information used in OAuth to authenticate and authorize access to protected services. When access tokens are stolen, unauthorized persons may get access to a user's resources without their knowledge or approval.

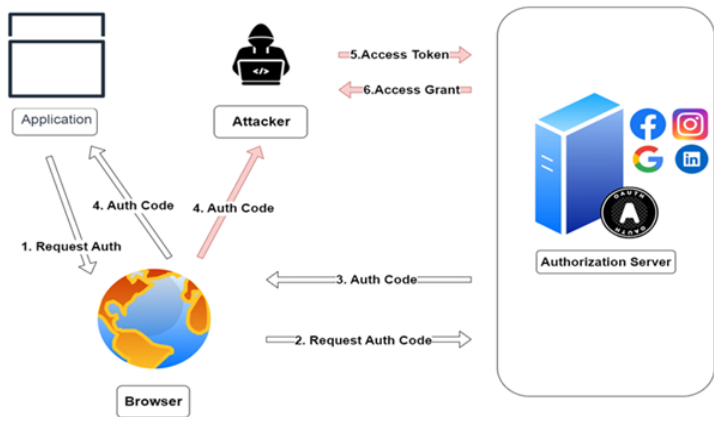
The authorization server frequently issues access tokens to the client once the user provides permission for the client to access their resources. When the access token is obtained, the client utilizes it to perform authorized requests to the resource server at the request of the user.

6.2.2 Authorization code Interception Attack

Authorization code interception is an attack in the OAuth flow in which an attacker attempts to intercept the authorization code while it is being transferred between the client and the authorization server. If the attacker successfully intercepts the authorization code, they may be able to take advantage of it to get an access token from the authorization server, allowing them to gain unauthorized access to the user's resources.

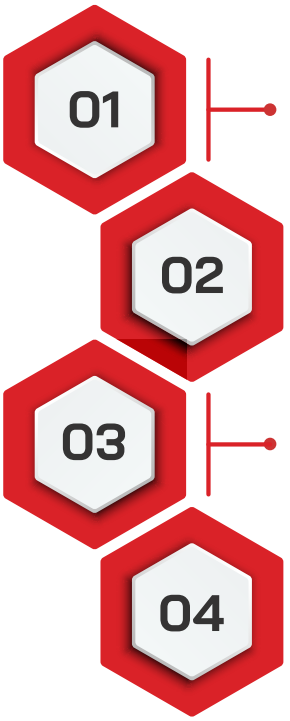
Request Authorization Auth code Apply code on Application Attacker get Auth code Access Code.

Auth Code Interception



6.3 Mitigations Points

To improve the security of OAuth implementations and reduce vulnerabilities, developers need to follow the latest methods and use suitable fixing systems. Here are some critical OAuth mitigation and remediation points:



Use the Authorization Code Grant Type

Use the authorization code grant type instead of the implicit grant type. Because the implicit grant type is less secure, it is better to use the authorization code grant type.

Implement CSRF Protection

Implement CSRF protection correctly. CSRF protection helps to prevent attackers from getting authorization codes or access tokens without the user's acceptance.

Secure Storage of Authorization Codes and Access Tokens

Keep authorization codes and access tokens in a secure location. Authorization codes and access tokens should be stored securely, such as with a hardware security module (HSM).

Validate the Scope Parameter

Validate the scope parameter appropriately. The scope option indicates the permissions that the client application is asking for. It is critical to validate the scope parameter accurately to prevent attackers from gaining more permissions than they are allowed to.

7.0

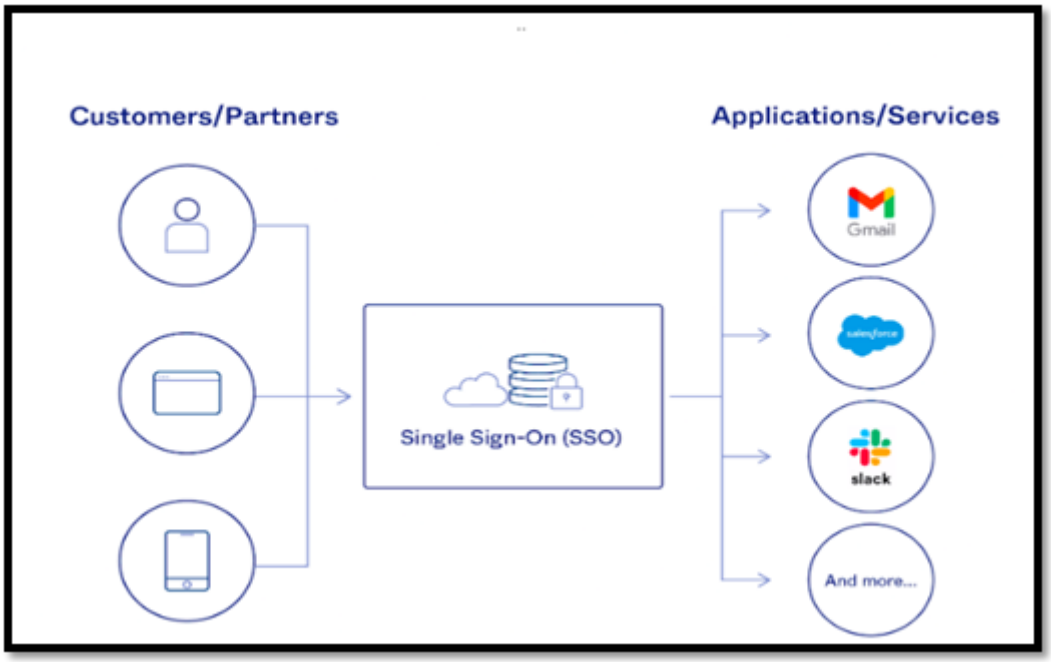
Single Sign On (SSO)

It is an authentication method that lets users access various apps or systems with a single pair of login credentials (username and password). Instead of maintaining different login credentials for each application, users only need to log in once to access numerous services.

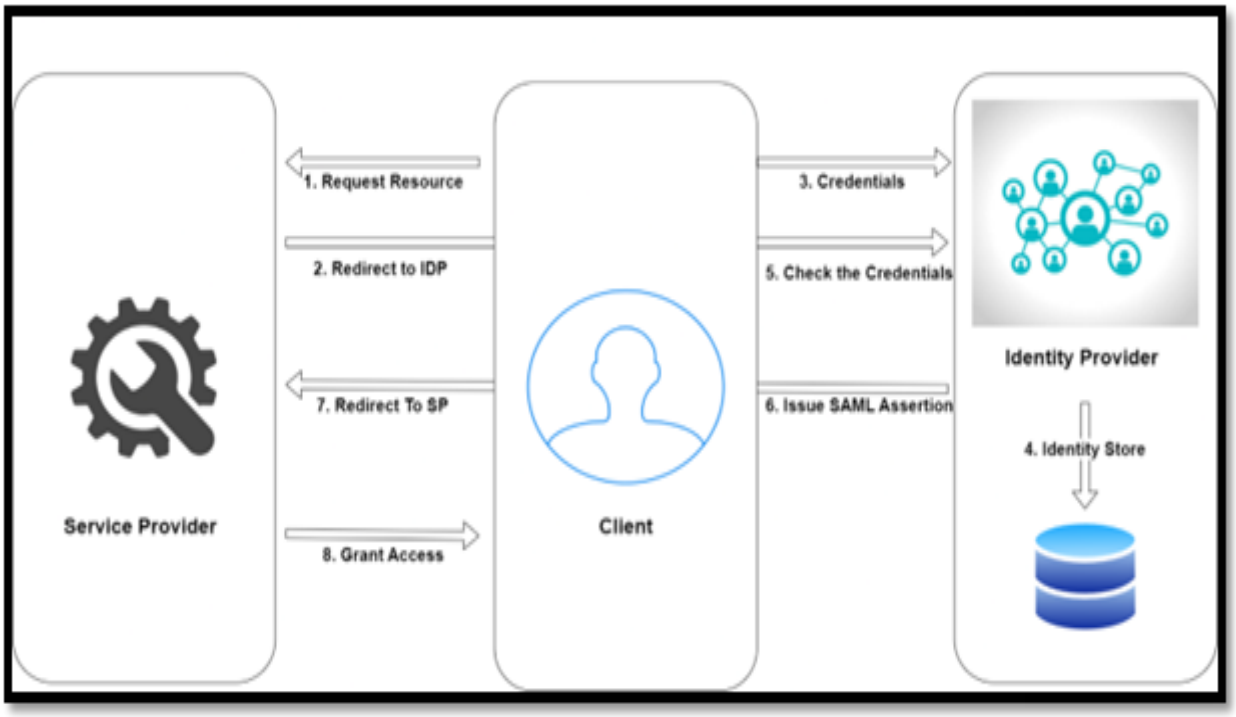
SSO is commonly used in a variety of scenarios, including within organizations for employee access to internal systems, cloud services, and web applications that connect with third-party services.

The benefits of using SSO are improved user experience, increased security (since users do not have to remember numerous passwords), and easier control of access by users to multiple applications.

SSO protocols and standards like SAML (Security Assertion Markup Language), and OpenID Connect make it easier to deploy single sign-on capability. SAML uses signed XML documents for exchanging identity information between services whereas OpenID Connect uses signed JSON Web Tokens.



7.1 How SSO Works?



User Request

STEP 01

The SSO process begins when a user wants to access a protected resource or service offered by the Service Provider (SP). Example of a SP is Gmail. It could be a web application, a cloud service, or any other system that requires authentication.

STEP 02

SP Redirects to IdP

Because the user has not yet been authorized, the SP redirects the user to the Identity Provider (IdP) with an authentication request. Example of an IDP is Auth, Okta, etc. Several techniques, including an HTTP redirect or an embedded login page, can be used to perform this redirection.

User Authentication

STEP 03

The user is provided with an authentication request at the IdP, which may include providing a username and password, or any other specified authentication technique.

IdP Authenticates User

STEP 04

The IdP verifies the user's identity by matching the submitted credentials to its user database.

IdP Issues Assertion

STEP 05

During a situation where the user authentication is successful, the IdP generates an authentication token that includes details about the user's identity and the authentication event.

Assertion Sent to SP

STEP 06

As a response to the authentication request, the IdP sends the created token back to the Service Provider (SP).

Assertion Validation

STEP 07

After receiving the token, the SP checks its authenticity and integrity by confirming the IDPs digital signature. This step confirms that the token was issued by an authorized IDP.

User Granted Access

STEP 08

If the Assertion is valid and the user is authenticated, the Service Provider allows the user to access the requested resource or service. The user can now access various applications without having to enter additional SP login credentials.

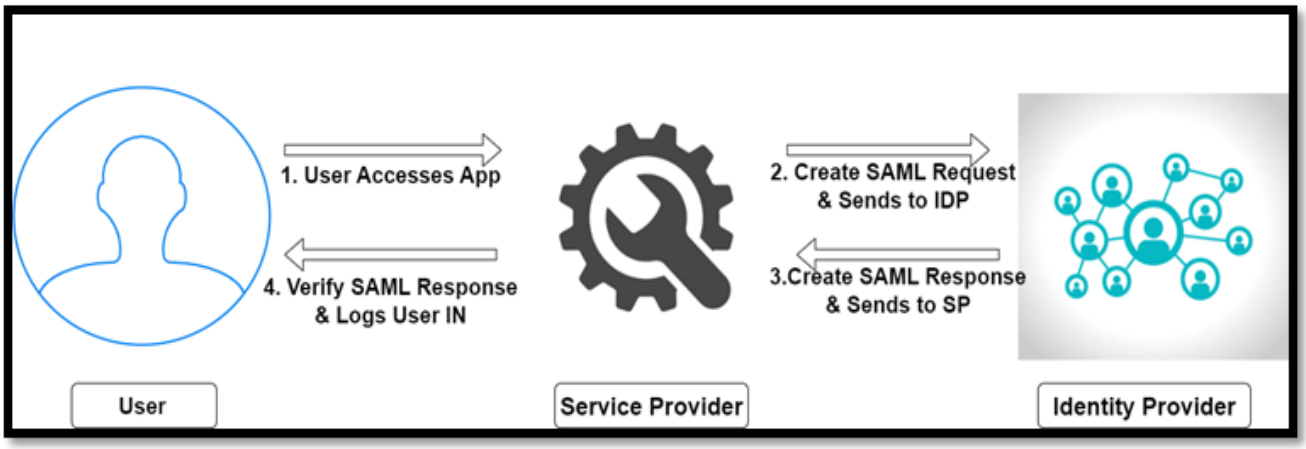
7.2 Security Assertion Markup Language (SAML)

SAML is an XML-based open standard for exchanging identity information/data between services, it consists of an Identity Provider (IdP) who authenticates users, a Service Provider (SP) who must validate user identity, and a SAML assertion containing user information that is transmitted between them.

This allows for smooth and secure access to resources without the need for multiple logins.

SAML is commonly used in enterprise environments and web-based applications to enable smooth and secure single sign-on across multiple systems without requiring users to submit their credentials several times. It is considered a reliable and safe method of identity federation, and it is especially important when trust must be built across the systems of different businesses.

7.2.1 How SAML works?



STEP 01

User Access Request

The Service Provider (SP) application observes when a user wants to access a protected resource that they are not authorized to do so. It sends a SAML Authentication Request (SAR) via a browser redirect to the Identity Provider.

STEP 02

SAML Assertion Generation

The IDP presents a login page to the user and after successfully authenticating the user, the IdP generates a SAML assertion. SAML Assertion is a cryptographically signed XML document (by the IdP) containing information about the user such as the user's identity, attributes, and what the user can access with the SP (authentication context).

STEP 03

SAML Response

The IdP sends the SAML assertion back to the user's browser as a response to the authentication request and the user's browser is redirected back to the SP, with the SAML assertion passing through as a query parameter or a form post.

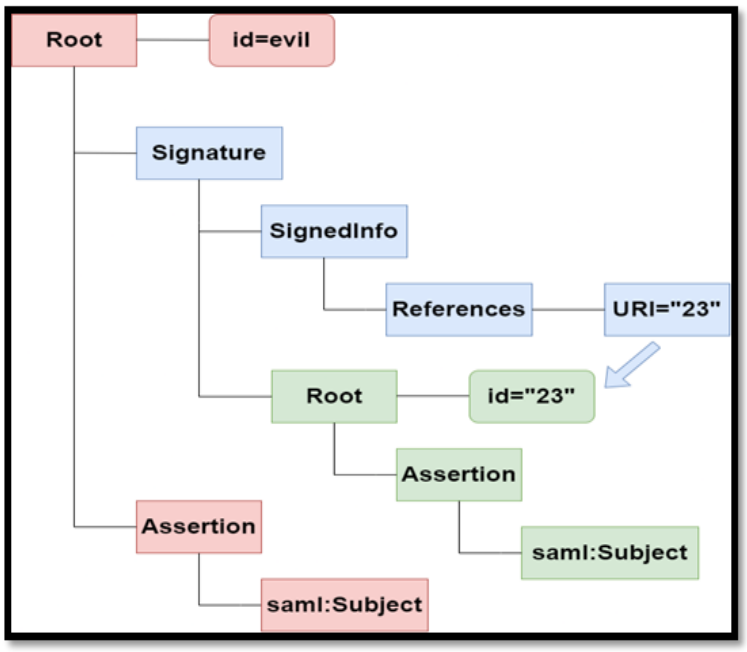
STEP 04

Verification of SAML Assertion

The SP gets the SAML assertion and validates its signature to ensure it came from the trusted IdP. This validation is usually done with public key cryptography. It also examines the user's properties and authentication context to confirm that the user meets the requirements, and the SP provides the user access to the requested resource or application without requiring a separate login.

7.2.2 Attacks in SAML

SAML (Security Assertion Markup Language), like any other technology, can be vulnerable to a variety of attacks if not properly secured. Among the most popular SAML attacks are:



7.2.2.1 XML Signature Wrapping Attack

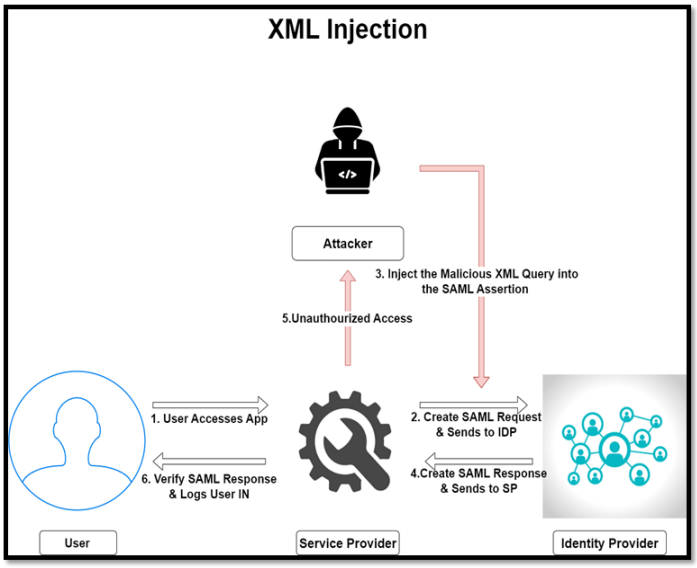
XML documents with XML Signatures are often processed in two separate steps: function activation (business logic) and signature validation. XML Signature Wrapping attacks (XSW) arises if both these modules have conflicting interpretations of the data.

By inserting forged parts into the message structure, the attacker in these assaults alters it but does not compromise the XML Signature. With this modification, it will be possible for the signature verification module to use a different portion of the message than the application logic. As a result, the receiver properly validates the XML Signature, but the fraudulent element is nevertheless processed by the application logic. The attacker can now inject any material by getting through the XML Signature's integrity protection and origin authentication.

7.2.2.2 XML Injection

XML Injection is a vulnerability that occurs when an attacker injects malicious XML code into an XML-based application or system. It is also known as XML Entity Injection or XXE (XML External Entity) Injection. This attack targets flaws in XML processing, allowing the attacker to take advantage of XML entities and potentially obtain unauthorized access to sensitive data or execute arbitrary code on the server.

The attacker would upload malicious XML data to a SAML service provider in a SAML XML injection attack. If the service provider fails to properly validate the input, the



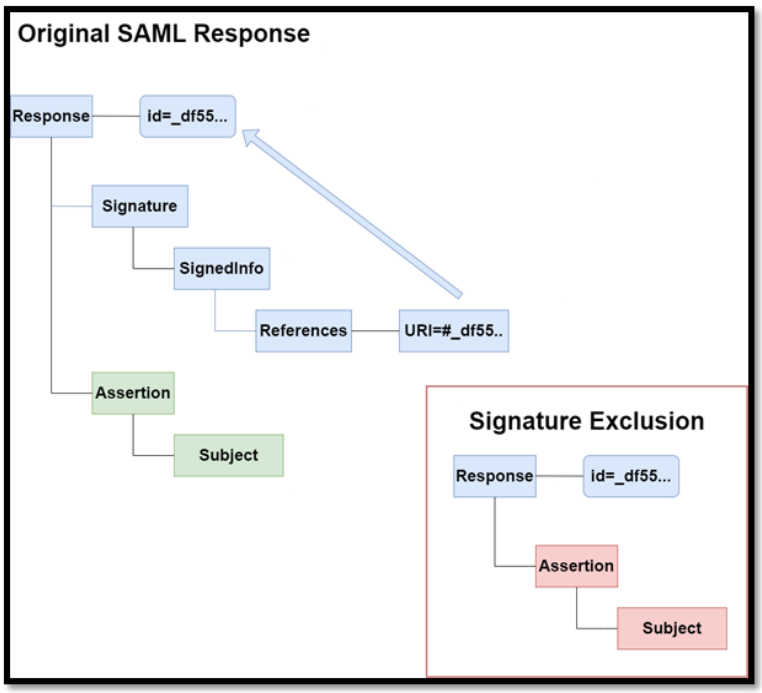
attacker may inject code that allows them to circumvent the authentication procedure and get access to the protected resources of the service provider.

To avoid SAML XML injection attacks, service providers should thoroughly check any XML data received from untrusted sources. This can be accomplished using several approaches such as XML schema validation, XSD filtering, and regular expression matching.

7.2.2.3 XML Signature Exclusion

XML Signature Exclusion is a security vulnerability that occurs when XML documents contain signatures but fail to properly include or validate these signatures. Digital signatures are frequently used in XML-based communication and data exchange to guarantee data integrity and authentication. Incorrectly incorporating or validating the signatures, can result in vulnerabilities.

For instance, if an XML document's signature is not properly validated, an attacker may be able to alter the document's content covertly, endangering the integrity of the data. In contrast, if the signature is ignored or incorrectly validated, it can permit harmful stuff to pass as legitimate, which could result in security breaches or data manipulation.



7.2.2.4 Certificate Faking

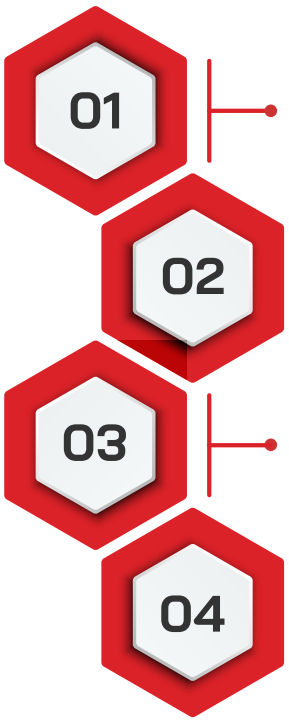
In SAML, certificate faking refers to a security risk in which an attacker attempts to pose as a trustworthy identity provider (IdP) or service provider (SP) by producing a fake digital certificate. For secure communication and entity authentication, SAML uses digital certificates.

If an attacker is successful in faking a certificate, they may assume the identity of an authorized IdP or SP and forge, change, or intercept SAML assertions to obtain access to an organized SAML system or carry out malicious acts.

To avoid this kind of attack, mitigations include restricted certificate validation during SAML transactions, monitoring of certificate transparency, and strong certificate management policies. The security of SAML-based authentication systems must also be regularly maintained through certificate renewal and revocation.

7.2.3 Mitigations points

Try out the following remediation steps to mitigate and address security vulnerabilities related to SAML (Security Assertion Markup Language):



Secure Communication

When transferring SAML messages between the Identity Provider (IdP) and Service Provider (SP), use robust encryption and secure communication protocols such as TLS/HTTPS. This ensures that the information sent remains private and that adversaries cannot intercept it.

Digital Signatures

Use digital signatures to assure the integrity and authenticity of SAML assertions. The SP can use signatures to ensure that the claims received from the IdP were not tampered with during transmission.

Certificate Validation

Verify the validity of SSL/TLS certificates used by the IdP and SP to establish a secure and trusted connection. Certificate validation helps in the prevention of man-in-the-middle attacks and assures that both parties can trust one another.

XML Parsing Security

Use safe XML parsing to avoid XML-based attacks such as XML External Entity (XXE) and XML Injection. These vulnerabilities can be avoided by doing proper input validation and sterilization.

8.0

NTLM

NTLM stands for “New Technology LAN Manager” and is a set of Microsoft security protocols used for authentication, especially in Windows-based networks. It was launched in the early 1990s as part of the Windows NT Operating System and has been used in later versions of Windows.

The NTLM authentication protocol is a method for securely authenticating users and computers in a network context. It uses a challenge-response mechanism that proves to a server or domain controller that a user knows the password associated with an account.

Despite known vulnerabilities, NTLM remains widely deployed on new systems to maintain compatibility with legacy clients and servers.

While NTLM is still supported by Microsoft, it has been replaced by Kerberos as the default authentication protocol in Windows 2000 and subsequent Active Directory (AD) domains.

The NTLM authentication protocols include LAN Manager version 1 and 2.

NTLMv1 (Windows LAN Manager version 1):

01 The first version of the NTLM authentication protocol is known as NTLMv1.

02 It uses weak security methods, making it vulnerable to a variety of attacks, like Pass-the-Hash attack.

03 It utilizes a one-way hash function for the user's password but does not use a server challenge to increase the security of the authentication process.

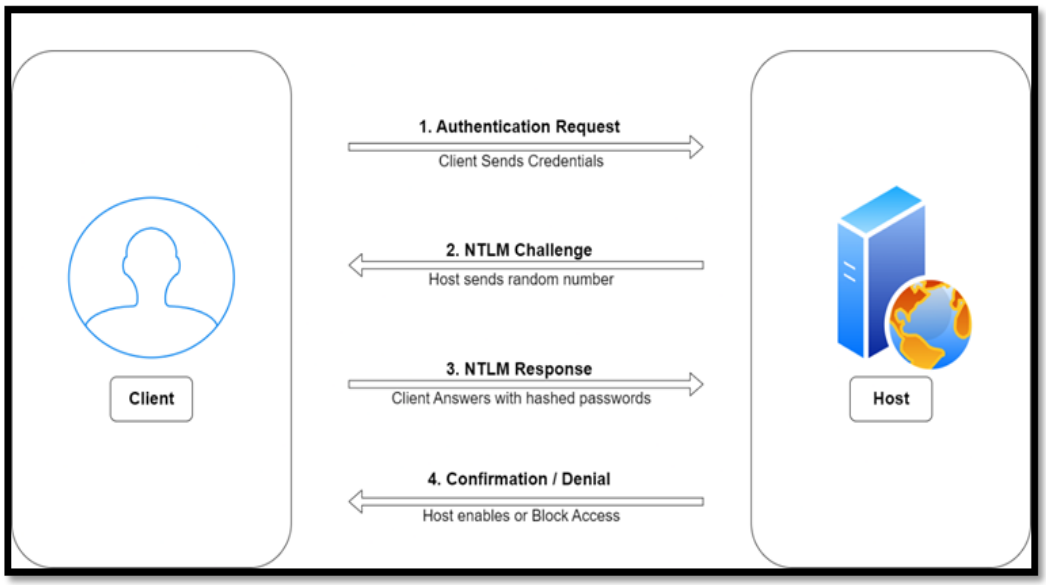
NTLMv2 (NT LAN Manager version 2):

01 NTLMv2 is an enhanced version of the NTLM protocol that provides greater security than NTLMv1.

02 It has stronger encryption and uses the HMAC-MD5 hash method for password hashing, making it more resistant to attacks.

03 The concept of a server challenge is also introduced in NTLMv2, adding an additional degree of protection to the authentication process.

8.1 How NTLM Works?



STEP 01

Client Request

When a client (often a Windows machine) wants to connect to a network resource, such as a file share or a web server, it sends a negotiation message to the server, indicating its willingness to use NTLM authentication. This message contains information such as the client's domain name, workstation name, and supported NTLM protocols.

STEP 02

Server Response

In response to the client's request, the server sends a challenge message, containing a random number known as the "challenge" or "nonce." The nonce is encrypted with a secret key known only to the server and the client's domain controller.

Client Authentication

STEP 03

The client uses its user credentials to compute a response to the challenge. This response includes a hash of the user's password, which is encrypted with the nonce received from the server.

The client hashes the user's password and the challenge using one of the NTLM hashing algorithms (LM Hash or NT Hash for NTLMv1, while HMAC-MD5 is used for NTLMv2).

STEP 04

NTLM Response

The hashed response (also known as the "NTLM response") is returned to the server by the client.

Server Verification

STEP 05

The server receives the client's NTLM answer. It uses its own copy of the client's password hash to compute a response.

STEP 06

Authentication Success or Failure

The server compares its hashed result to the NTLM response from the client. If they match, authentication is successful and access is granted; otherwise, authentication fails, and access is denied.

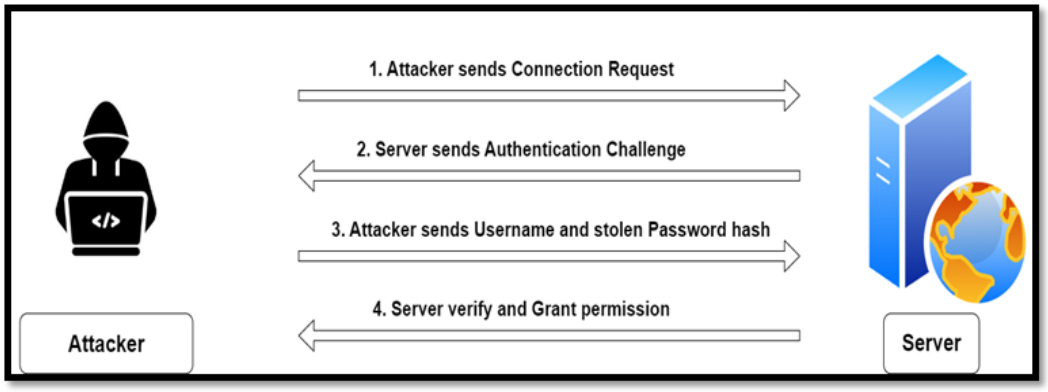
8.2 Attacks in NTLM

NTLM is vulnerable to several recorded flaws and weaknesses, rendering it vulnerable to various attacks. The most notable NTLM-related attacks are as follows:

8.2.1 Pass the Hash Attack

A pass-the-hash attack occurs when an attacker steals a "hashed" user credential and uses it to create a new user session on the same network. Hashing is a crypto function that transforms a password into an irreversible value. This means that even if the attacker obtains the hash, they will not be able to easily crack it to retrieve the plaintext password.

Unlike other credential theft attacks, a pass the hash attack does not require the attacker to know or crack the password to gain access to the system.



However, pass-the-hash attacks can still be successful since NTLM, Windows' authentication protocol, does not require the usage of the plaintext password for authentication. Instead, it just requires the password hash. This means that an attacker can just send the hash to the NTLM authentication service, which will authenticate the user as if they had submitted their plaintext password. The attacker gains full system access, enabling lateral movement across the network.

8.2.2 NTLM Relay Attack

An NTLM relay attack is a type of cyberattack in which an attacker intercepts and relays NTLM (Windows NT LAN Manager) authentication messages between a client and a target server. On networks where NTLM is used for authentication, this attack frequently happens. The attacker intercepts NTLM authentication requests coming from the client and sends them on to another server, usually one that is envious. If successful, this attack may result in illegal access or a rise in server access.

Organizations can put in place security measures to prevent NTLM relay attacks, such as utilizing Kerberos or another more secure authentication protocol, turning off NTLM when it can, and setting up appropriate network segmentation and access rules to limit NTLM communication to trusted systems. Multi-factor authentication (MFA) implementation can also add an additional layer of security, making it more challenging for attackers to be successful in NTLM relay attacks even if they manage to intercept authentication messages.

8.2.3 NTLM Downgrade Attack

A NTLM (NT LAN Manager) downgrade attack is a kind of security attack that targets Windows-based authentication mechanisms. NTLM is a network authentication technique that was used in early versions of Windows operating systems and is considered less secure than current authentication methods such as Kerberos.

The NTLM downgrade attack takes advantage of the fact that many systems are still set up to accept NTLM for backward compatibility, even though more secure authentication protocols are available. An attacker can force a target system to utilize NTLM for authentication instead of a more secure approach by using this vulnerability.

An NTLM downgrade attack happens when an attacker intercepts a client's and a server's communication and forces them to use a weaker version of the NTLM protocol, making it easier for the attacker to abuse the authentication process. Downgrading the protocol allows the attacker to take advantage of known flaws and crack the hashed password used in authentication.

8.3 Mitigation Points

Try out the following remediation points to improve the security of NTLM (NT LAN Manager) authentication in your network environment and mitigate potential vulnerabilities:

Disable NTLM when Possible

Whenever possible, disable NTLM authentication in Favor of more secure authentication mechanisms such as Kerberos

Enable NTLMv2

If NTLM is required for security, make sure NTLMv2 is enabled on all devices. NTLMv2 is more secure than NTLMv1 and protects against additional types of attacks.

Disable LM Hash

In NTLMv1, LM Hash is a bad password hashing technique. Disable LM Hash in the Windows Group Policy settings to prevent the system from storing weak password hashes.

Enforce Strong Password Policies

Encourage users to create strong passwords that are resistant to brute force attacks and password guessing. Password complexity standards, password expiration policies, and account lockout policies should all be implemented.

Passwords for Local Administrator Accounts Should Be Complex and Unique

It is best to avoid using the same local administrator password on numerous platforms. To prevent pass-the-hash attacks, use complex and unique passwords for local administrator accounts.

Implement Network Aegmentation

Network segmentation can help prevent relaying attacks by limiting the ability of attackers to intercept and forward authentication requests.

Use Encryption

NTLM authentication should be used over encrypted channels to protect against attacks that intercept traffic and steal authentication information.

9.0

Lightweight Directory Access Protocol (LDAP)

LDAP is an application protocol used to access and maintain directory information. It is a widely used directory service standard that is primarily used for centralized user authentication, authorization, and information retrieval. LDAP directories are hierarchical data storage with a tree-like structure that commonly looks like an organizational chart.

LDAP is widely utilized in many different applications and services, such as enterprise authentication, user administration, email systems, and centralized access control. Its adaptability and scalability make it an invaluable tool for efficiently managing large-scale directory information. However, as with any technology, proper configuration and security procedures are required to prevent any LDAP-related vulnerabilities, such as injection attacks or unauthorized access to directory data.

9.1 How LDAP Works?

Try out the following remediation points to improve the security of NTLM (NT LAN Manager) authentication in your network environment and mitigate potential vulnerabilities:

STEP 01

Make a connection

1.1 **Client Initialization:** The LDAP client, which could be an application or a system, creates a connection to the LDAP server. This entails supplying the hostname or IP address of the server as well as the port to connect to (usually port 389 for LDAP or port 636 for LDAPS).

1.2 **Server Listening:** The LDAP server is operating and listening on the given port for inbound connections.

STEP 02

Authentication

2.1 **Client Authentication:** If authentication is necessary, the LDAP client supplies credentials, often in the form of a username and password.

2.2 **Server Verification:** The LDAP server validates the credentials provided against its directory database. If the credentials are valid, the server provides the client access. Otherwise, access is denied.

STEP 03

Search and Retrieve data

3.1 **Client Query:** After successful authentication, the LDAP client can submit search queries to the server. These queries include search criteria such as a starting point in the directory tree (Distinguished Name, or DN) and search filters to reduce results.

3.2 **Server Verification:** The LDAP server validates the credentials provided against its directory database. If the credentials are valid, the server provides the client access. Otherwise, access is denied.

3.2 **Result Delivery:** The server sends the search results to the client. Results may contain directory items (objects) and their properties that fit the search criteria.

STEP 04

Modify and Update the Data

4.1 **Client Modification:** LDAP clients can send Modify requests to the server to update directory data. This includes adding, removing, or changing properties and values inside an entry.

4.2 **Server Authorization:** The LDAP server determines if the client has the appropriate permissions to make the requested changes. If the server is approved, the modifications are applied to the directory.

STEP 05

Encryption and Security

5.1 **TLS/SSL Negotiation:** If secure communication is necessary, the client and server use TLS/SSL to negotiate a secure connection. This encrypts the data they exchange.

5.2 **Access Control:** LDAP servers apply access control rules to guarantee that only authorized users can view and modify directory data. These rules are frequently defined using access control lists (ACLs).

STEP 06

Handling Responses and Errors

6.1 **Server Responses:** The server responds to each client request (search, change, etc.). Responses can include signals of success, data required, or errors.

6.2 **Error Codes:** LDAP defines specific error codes and error messages to aid in the diagnosis and resolution of problems. Typical issues include failed authentication, inadequate permissions, and

STEP 07

Disconnect the connection

7.1 **Graceful Closure:** After finishing its actions, the client can gracefully stop the LDAP server connection. Proper closure ensures that resources are released and that network resources are utilized efficiently.

9.2 Attacks in LDAP

LDAP (Lightweight Directory Access Protocol) is vulnerable to a variety of attacks, particularly if not properly secured. Here are some examples of popular LDAP attacks:

9.2.1 LDAP Injection

LDAP injection is a cyberattack that involves inserting malicious data into the LDAP queries to alter or compromise a directory service. An LDAP query is a request sent to an LDAP directory server to retrieve or modify information.

An attacker can employ LDAP injection to extract sensitive data, modify directory entries, or expose the structure of a directory.

Here are some examples of LDAP queries:

- To search for all users in the directory with a given name:
  (&(objectClass=user) (cn=John))
  John represents the name to be searched in the above LDAP query.

- John represents the name to be searched in the above LDAP query.To search for all groups in the directory with a given name:
  (&(objectClass=group) (cn=Marketing))
  Marketing represents the name to be searched in the above LDAP query.

- To search for all users in each group -
  (&(objectClass=user)
  (memberOf=cn=Marketing,ou=Groups,dc=example,dc=com))

It occurs in the same way the SQL injection occurs, when user-supplied data is not properly verified and sanitized, allowing attackers to change the query's behaviour. An LDAP query that is sent to the directory server is altered by the LDAP injection. The attacker can then access the server or extract private data from the directory using the tricked query. This can be accomplished by inserting a malicious query into the user input through any LDAP-compatible application or system.

To protect against LDAP injection, validate and sanitize user input, utilize parameterized queries, follow the principle of least privilege, escape special characters, and keep secure LDAP server configurations. In addition to preventing LDAP injection attacks and safeguarding directory services, regular monitoring and upgrades are critical.

9.2.2 LDAP MITM

In an LDAP man-in-the-middle (MitM) attack, a hostile actor intercepts and potentially modifies communications between an LDAP client and server. The attacker may be able to steal sensitive data or alter LDAP communication because of this interception

Encryption, such as TLS/SSL (LDAPS), should be used to safeguard data transmission in LDAP to defend against MitM attacks. Certificate validation validates the LDAP server's legitimacy.

Network segmentation reduces exposure, while intrusion detection technologies aid in the discovery of suspicious behaviour. To mitigate MitM dangers and preserve LDAP communication, regular upgrades, strong authentication, and security awareness training are also required..

9.3 Mitigation Points

Below are few important LDAP mitigation points:

Implement Strong Access Controls

Enforcing the concept of least privilege guarantees that people and systems only have minimal essential access, decreasing the possibility of unwanted access. Access management is simplified when access control lists (ACLs) are reviewed regularly, and role-based access control (RBAC) is used.

Use Strong Authentication

MFA increases security by requiring users to provide multiple authentication factors. To strengthen authentication, promote strong passwords, and enforce regular updates while removing insecure authentication methods like Insecure Bind.

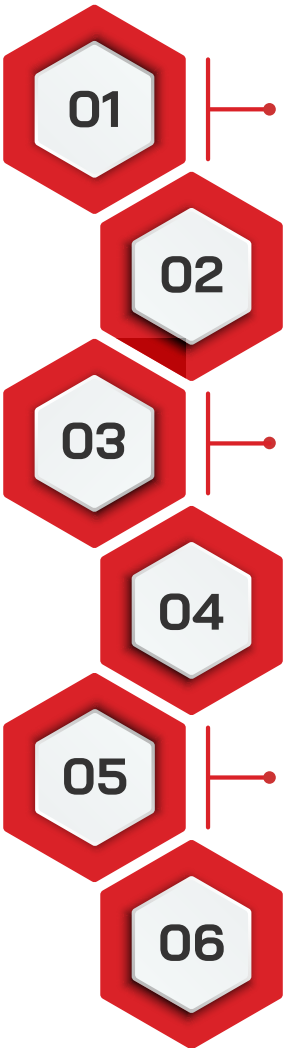
Encrypt Sensitive Data

Using protocols such as TLS or SSL to encrypt data in transit avoids eavesdropping. Encrypting data at rest protects it from unwanted access, even in the event of a data breach. Encryption is critical for maintaining data privacy and security.

10.0

Conclusion

In conclusion, authentication is an important part of security in modern systems and applications. Understanding both the benefits and drawbacks of various authentication methods matters in developing effective security measures. This paper examined many authentication systems, including Basic Auth and Form-based Auth, MFA/2FA, Token-based Auth, SSO, SAML, NTLM, and LDAP as well as the Form-based mitigation steps connected with each.



Basic authentication, while easy, is vulnerable to attacks such as MITM, Brute Force, Dictionary Attacks, and vulnerabilities in weak or default passwords.

Two-factor authentication (2FA) can improve security and protect against brute force attacks.

Token-based authentication provides flexibility and security, but it is vulnerable to MITM, token theft, and token brute force. Mitigation requires secure token transfer, the use of short-lived tokens, and proper validation.

SSO simplifies user access, however it is vulnerable to MITM, Brute Force, and CSRF attacks. Key mitigations include CSRF protection and secure token transmission, as well as strong SSO passwords and 2FA.

SAML offers SSO and federated identity, but it also allows for XML Injection, Attribute Manipulation, and Identity Provider Impersonation. SAML protection requires XML security, encryption, and secure token communication.

NTLM authentication risks pass-the-hash attacks, and LDAP can expose credentials during transmission; implementing stronger protocols like Kerberos and using LDAPS can enhance security.

Overall, organizations should use a layered authentication strategy that combines strong password rules, encryption, multi-factor authentication, secure transmission, and frequent checks for safety. It is also critical to inform users on best practices. Organizations can build a more secure authentication environment, securing sensitive information and preventing unwanted access, by adopting proactive efforts and remaining watchful against new attacks.

About Payatu

Payatu is a Research-powered cybersecurity services and training company specialized in IoT, Embedded Web, Mobile, Cloud, & Infrastructure security assessments with a proven track record of securing software, hardware and infrastructure for customers across 20+ countries.

Our deep technical security training and state-of-the-art research methodologies and tools ensure the security of our client's assets.
At Payatu, we believe in following one's passion, and with that thought, we have created a world-class team of researchers and executors who are willing to go above and beyond to provide best-in-class security services. We are a passionate bunch of folks working on the latest and cutting-edge security technology.

One of the Payatu's brand, EXPLIoT is all set to launch its revolutionary IoT Security Assessment Platform – IoT Auditor.

Don't forget to sign up for its limited access community edition.

If you're only getting started with IoT Security, get your hands on the IoT Security Learning Kit.

Visit the EXPLIoT store here.

Want to know more about the distinct premium services security services offered by Payatu?

Tell us about your specific requirements here, and we will get back to you with a customized sample report.

Payatu Security Consulting Pvt. Ltd.

www.payatu.com
info@payatu.com
+91 20 41207726