**Attacking Authentication**

* **Authentication Methods**
* **HTTP Authentication**
* **2 factor Authentication Bypass**
* **Insecure captcha**
  + **Authentication Methods**
    - Html form based (most used)
    - Multi factor authentication
    - HTTP Authentication (old method)
      * Such as {Basic , Digest , Bearer}
    - Windows integrated authentication
      * Such as NTLM , Kerberos used mostly in intranet for organization
    - OAUTH Header
  + **HTTP Authentication** 
    - **Basic Authentication**
      * **Request**: GET /BasicAuth HTTP/1.1
      * **Response**: WWW-Authenticate: Basic realm="Any Text"
      * **Request**: Authorization: Basic base64(username:password)
        + Ex: Authorization: Basic YWRtaW46YWRtaW4=
      * **Response**: Failed or Success
    - **Digest Authentication**
      * **Request**: GET /DigestAuth HTTP/1.1
      * **Response**: WWW-Authenticate: Digest realm="Any Text", nonce="w4n4QvmDBQA=fc49b571115859e9a3b7ac4d9c68ec06bdf6415f", algorithm=MD5, qop="auth"
      * **Request**: Authorization: Digest username="user1", realm="Any Text",
        + nonce="w4n4QvmDBQA=fc49b571115859e9a3b7ac4d9c68ec06bdf6415f",
        + uri="/scripts/digest",
        + algorithm=MD5,
        + response="2b2c8d7e6f3c495b3b7d2b963198f596",
        + qop=auth,
        + nc=00000001,
        + cnonce="6b15c646d6dc6d43"

Ex: HA1 = MD5(username:realm:password)

HA2 = MD5(method:digestURI)

response = MD5(HA1:nonce:nc:cnonce:qop:HA2)

* + - * **Response**: Failed or Success
  + **Bypass 2-Factor authentication**
    - **introduction**
      * Something You Know
        + Password, PIN, Connect the Dots, etc.
      * Something You Have
        + USB token, smartcard, RFID transmitter, dongle, etc.
      * Something You Are
        + Biometrics, fingerprints, retina scan, smell I
    - **Types of Authentication Tokens:**
      * Two-Factor Authentication (2FA) is a security feature that prevents your account from being stolen if an attacker knows your password. The website you're logging into requires you to provide a second code, in addition to your normal password. Ideally this code has been generated by using a  One-Time Password (TOTP) Currently, there are three different OATH OTP types that are the most widely used: event-based tokens, time-based tokens, and challenge-based tokens.
      * **Event-Based Token (HOTP):** An OTP system generates event-based tokens on demand using a combination of a static random key value (HMAC; the H in HOTP) and a dynamic value, such as a counter (IETF, 2005). The event-based token is usually valid for a variable amount of time, but could be valid for an unlimited amount of time.
      * **Time-Based Token (TOTP):** An OTP system generates time-based tokens automatically every so often based on a static random key value and a dynamic time value (such as currently time of day). The time-based token is only valid for a certain amount of time, such as 30 or 60 seconds (IETF, TOTP: Time-Based One-Time Password Algorithm, 2011). TOTP is a subset of HOTP.
      * **Challenge-Based Token (OCRA):** An OTP system generates challenge-based tokens on demand (IETF, OCRA: OATH Challenge-Response Algorithm, 2011), using a random challenge key that is provided by the authentication server at each unique user log-in. The challenge-based token is valid for a certain amount of time such as several minutes.
    - **Methods to bypass 2FA**
      * **Bypassing 2fa using conventional session management**
        + This method is about bypassing the two-factor authentication mechanism using password reset functions. In almost all web applications the password reset function automatically logs the user into the application after the reset procedure is completed
        + Go to Change Password > Request Password Reset Token > Use Password Reset token > Login to the web application
      * **Bypassing 2fa Via Oauth mechanism**
        + A Oauth integration is a third party login mechanism that allows a user to login using a third party account. This is normally of a renowned web application such as facebook or google. A typical Oauth request works in the following way
        + [Site.com](http://Site.com) requests facebook for auth token > facebook verifies user account > Facebook send callback code > [Site.com](http://Site.com) logs user in
        + As it is observed that in this process flow there is no intervention of 2fa. An attacker can potentially abuse this mechanism and utilize a Oauth integration to log into the web application rather than using the username and password to do so (Shah, 2014). For this bypass to work the attacker must have access to the Oauth integration account to login on behalf of the user
      * **Bypassing 2fa via brute force**
        + Web developers leave a very disdinctive flaw when they forget to put rate limitation on the input fields, in case of 2fa if the field is not rate limited there is a possibility of brute force attacks using which the attacker can brute force the 2fa code sent to the device. Usually the length of the 2fa code is 4 to 6 characters which often is numbers, and that makes to a possibility 151,800 which in real world scenario is easily brute forceable using a normal computer
      * **Bypassing 2fa using race conditions**
        + An attacker can **utilize previously used or unused values of tokens to verify the device**. However, this technique requires the attacker to have access to the previously generated values, which can be done via reversing the algorithm of the code generation app or intercepting a previously known code.
  + **Examples of vulnerable authentication flaws**
    - Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
    - Permits brute force or other automated attacks.
    - Permits default, weak, or well-known passwords, such as”Password1″ or “admin/admin.″
    - Uses weak or ineffective credential recovery and forgot-password processes, such as “knowledge-based answers,” which cannot be made safe.
    - Uses plain text, encrypted, or weakly hashed passwords.
    - Has missing or ineffective multi-factor authentication.
    - Exposes session IDs in the URL (e.g., URL rewriting).
    - Does not rotate session IDs after successful login.
    - Does not properly invalidate session IDs. User sessions or authentication tokens (particularly single sign-on (SSO) tokens) aren’t properly invalidated during logout or a period of inactivity.
  + **How to test for Authentication Flaws**
    - Sending credentials with GET method instead of POST method
    - Sending credentials over http not https that can allow for eavesdropping
    - Store credentials in plaintext in database passwords must be hashed and salted
    - Allow bruteforcing against passwords many apps Doesn’t implement any prevention mechanism such as lockout the account after nb of failed login trials
    - Allow for bad passwords or small ones that can be easily bruteforced or guessed
    - Insecurely implement change password functionality
    - Insecurely implement Forget Password functionality
    - Insecurely implement Remember me functionality
    - Insecurely implement impersonation functionality
    - Test multi factor authentication
    - insecure captcha