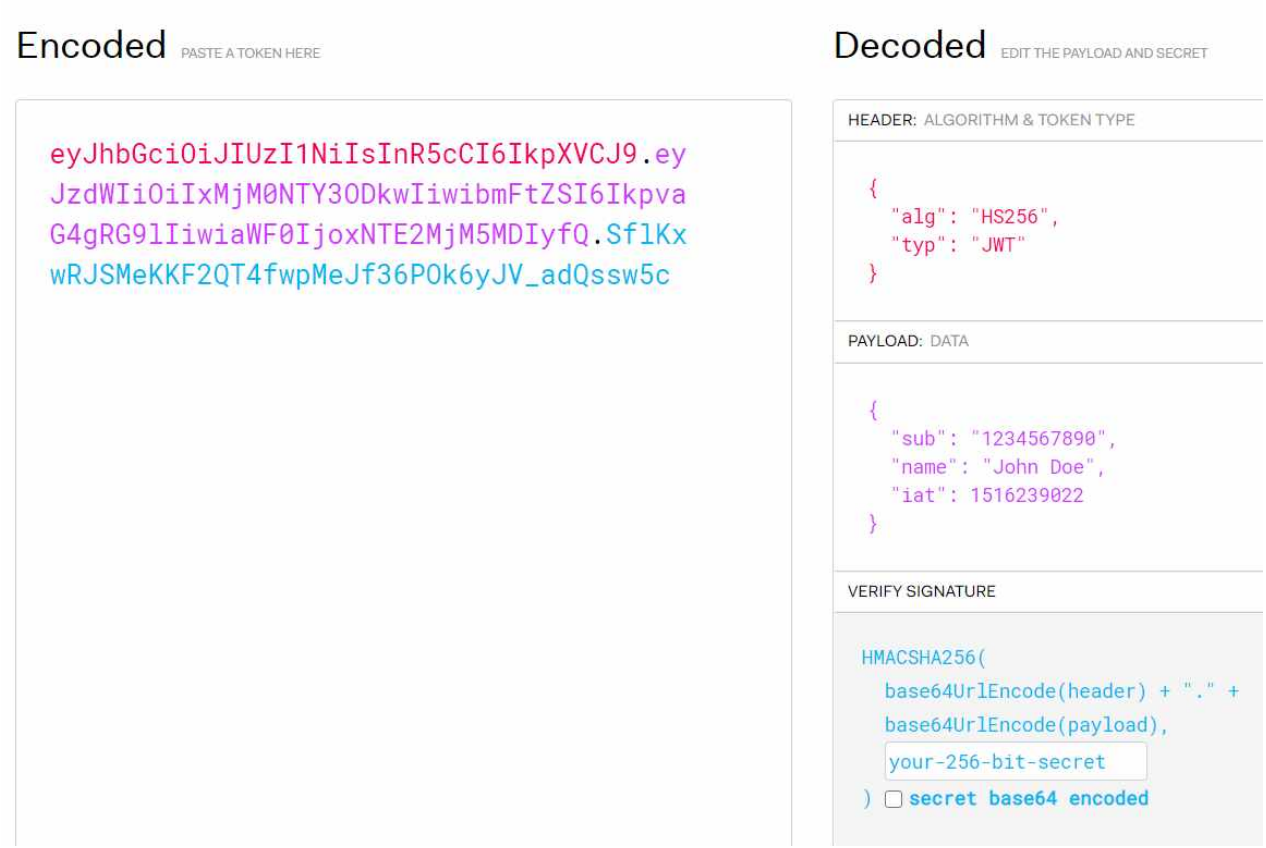
WEB AUTHENTICATION SECURITY

(Elbayi Asgarov, Mark Veiner, Saeed Ahmad)

For secure authentication JSON Web Token (JWT) authentication was used.

JSON web token (JWT), pronounced "jot", is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

Because of its relatively small size, a JWT can be sent through a URL, through a POST parameter, or inside an HTTP header, and it is transmitted quickly. A JWT contains all the required information about an entity to avoid querying a database more than once. The recipient of a JWT also does not need to call a server to validate the token.



In its compact form, JSON Web Tokens consist of three parts separated by dots (.), which are:

Header

Payload

Signature

Therefore, a JWT typically looks like the following.

xxxxx.yyyyy.zzzzz

Header

The header typically consists of two parts: the type of the token, which is JWT, and the signing algorithm being used, such as HMAC SHA256 or RSA.

For example:

{"alg": "HS256",

"typ": "JWT"}

Then, this JSON is Base64Url encoded to form the first part of the JWT.

Payload

The second part of the token is the payload, which contains the claims. Claims are statements about an entity (typically, the user) and additional data. There are three types of claims: registered, public, and private claims.

Registered claims: These are a set of predefined claims which are not mandatory but recommended, to provide a set of useful, interoperable claims. Some of them are: iss (issuer), exp (expiration time), sub (subject), aud (audience), and others.

Public claims: These can be defined at will by those using JWTs. But to avoid collisions they should be defined in the IANA JSON Web Token Registry or be defined as a URI that contains a collision resistant namespace.

Private claims: These are the custom claims created to share information between parties that agree on using them and are neither registered nor public claims.

An example payload could be:

{"sub": "1234567890",

"name": "John Doe",

"admin": true}

The payload is then Base64Url encoded to form the second part of the JSON Web Token.

Signature

To create the signature part you have to take the encoded header, the encoded payload, a secret, the algorithm specified in the header, and sign that.

For example, if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:

HMACSHA256(

base64UrlEncode(header) + "." +

base64UrlEncode(payload),

secret)

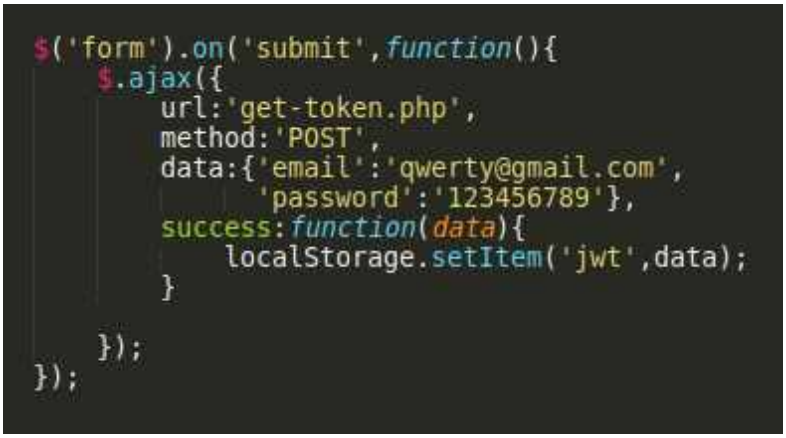
In general, JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it.

Before a received JWT is used, it should be properly validated using its signature. Note that a successfully validated token only means that the information contained within the token has not been modified by anyone else. This does not mean that others weren't able to see the content, which is stored in plain text.

Where to store?

It is not secure to store JWT token on the client-side:

Local storage and cookies can be accessed by any script in the browser leaving JWT vulnerable to XSS attack.



```
$('form').on('submit',function(){
    $.ajax({
        url:'get-token.php',
        method:'POST',
        data:{'email':'qwerty@gmail.com',
              'password':'123456789'},
        success:function(data){
            Cookies.set('jwt',data);
        }

    });
});
```

We can set cookie read only from server side with flags *httpOnly:*

```
        $jwt = JWT::encode($token, $secret_key,"HS512");
        echo json_encode(
                    array(
                "success" => "true",
                "jwt" => $jwt,
            ));

        header("Set-Cookie: jwt=".$jwt."; path=/;httpOnly");
```

Or we can save it in global variable for Single page web apps:

```
    $.ajax({
        url:'http://192.168.43.51/php-jwt/api/login.php',
        method:'POST',
        data:{   "email":$('#login').val(),
                 "password":$('#password').val()
             },
        success:function(data){
            jwt=data;       }
```

Disadvantages of JWT authentication:

JWT tokens cannot be invalidated:

logout

compromised accounts

password changes

permission changes

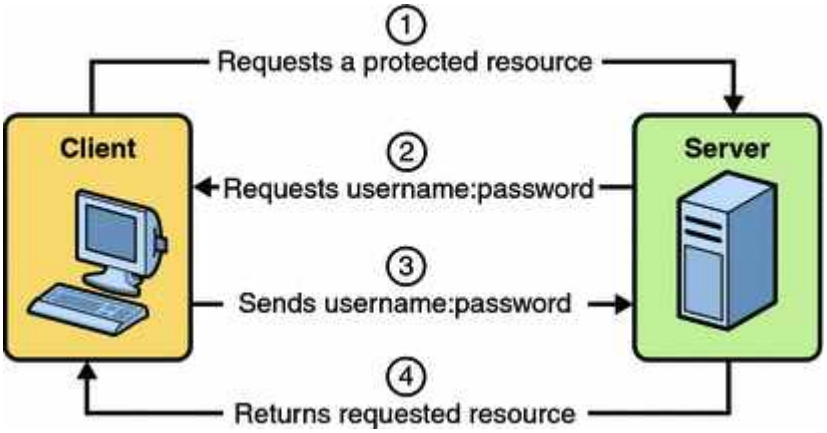
As JWT every time will be sent, when user will request something from server, it is not recommended to include sensitive information in JWT unless https is used. We have option to encrypt JWT itself with an encryption

algorithm (JWE) to prevent man in the middle to parse it but even if it will be intercepted it can be used to forge requests.

1) How HTTP Basic Authentication work

HTTP Basic Authentication requires that the server request a user name and password from the web client and verify that the user name and password are valid by comparing them against a database of authorized users. When basic authentication is declared, the following actions occur:

1. A client requests access to a protected resource.
2. The web server returns a dialog box that requests the user name and password.
3. The client submits the user name and password to the server.
4. The server authenticates the user in the specified realm and, if successful, returns the requested resource.



HTTP basic authentication is not a secure authentication mechanism. Basic authentication sends user names and passwords over the Internet as text that is Base64 encoded, and the target server is not authenticated. This form of authentication can expose user names and passwords. If someone can intercept the transmission, the user name and password information can easily be decoded. However, when a secure transport mechanism, such as SSL, or security at the network level, such as the IPSEC protocol or VPN strategies, is used in conjunction with basic authentication, some of these concerns can be alleviated.

2) HTTP Authentication Mechanisms (RFC2617)?

Digest access authentication is one of the agreed-upon methods a **Web Server** can use to negotiate credentials, such as username or password, with a user's **Web Browser**. This can be used to confirm the identity of a user before sending sensitive information, such as online banking transaction history. It applies a **Hash Function** to the **Username** and **Password** before sending them over the network. In contrast, **Basic Access Authentication** uses the easily reversible **BASE 64** encoding instead of hashing, making it non-secure unless used in conjunction with Transport Layer Security.

Example

The following example is expanded here to show the full text expected for each request and response.

This typical transaction consists of the following steps:

1. The client asks for a page that requires authentication but does not provide a username and password. Typically, this is because the user simply entered the address or followed a link to the page.
2. The server responds with the 401 "Unauthorized" response code, providing the authentication realm and a randomly generated, single-use value called a **Nonce** (Is an arbitrary number that can be used once in communication)
3. At this point, the browser will present the authentication realm (typically a description of the computer or system being accessed) to the user and prompt for a username and password. The user may decide to cancel at this point.
4. Once a username and password have been supplied, the client resends the same request but adds an authentication header that includes the response code.
5. In this example, the server accepts the authentication and the page is returned. If the username is invalid and/or the password is incorrect, the server might return the "401" response code and the client would prompt the user again.

3) HTTP Authentication Security Problem

Personal Information Leakage

HTTP clients are often privy to large amount of personal information such as the user's name, location, mail address, passwords, encryption keys, etc. So, you should be very careful to prevent unintentional leakage of this information via the HTTP protocol to other sources.

- All the confidential information should be stored at the server in encrypted form.
- Revealing the specific software version of the server might allow the server machine to become more vulnerable to attacks against software that is known to contain security holes.
- Proxies that serve as a portal through a network firewall should take special precautions regarding the transfer of header information that identifies the hosts behind the firewall.

- The information sent in the 'From' field might conflict with the user's privacy interests or their site's security policy, and hence, it should not be transmitted without the user being able to disable, enable, and modify the contents of the field.
- Clients should not include a Referrer header field in a (non-secure) HTTP request, if the referring page was transferred with a secure protocol.
- Authors of services that use the HTTP protocol should not use GET based forms for the submission of sensitive data, because it will cause the data to be encoded in the Request-URI.

Authentication Credentials

Existing HTTP clients and user agents typically retain authentication information indefinitely. HTTP/1.1 does not provide a method for a server to direct clients to discard these cached credentials which is a big security risk.

Proxies and Caching

HTTP proxies are men-in-the-middle, and represent an opportunity for man-in-the-middle attacks. Proxies have access to security-related information, personal information about individual users and organizations, and proprietary information belonging to users and content providers.

Proxy operators should protect the systems on which proxies run, as they would protect any system that contains or transports sensitive information

4) How does SSL/TLS work

What is SSL

SSL stands for Secure Sockets Layer, and it refers to a protocol for encrypting and securing communications that take place on the Internet. Although SSL was replaced by an updated protocol called TLS Transport Layer Security some time ago, "SSL" is still a commonly used term for this technology.

The main use case for SSL/TLS is securing communications between a client and a server, but it can also secure email, VoIP, and other communications over unsecured networks.

How Does SSL/TLS work?

There are the essential principles to grasp for understanding how SSL/TLS works.

1. Secure Communication begins with TLS Handshake in which the two communication parties open secure connection and change a public key.
2. During the TLS handshake, the two parties generate session keys, and the session keys encrypt and decrypt all communications after the TLS handshake.
3. Different session keys are used to encrypt communications in each new session.

4. TLS ensures that the party on the server side or the website the user is interacting with is actually who they are claim to be.
5. TLS also ensures that data has not been altered since a Message Authentication Code MAC is included with transmissions.

5) What is the purpose of user side password Hashing?

But simply hashing the password on the client side is only just better than submitting it as plain text to the server. Someone, who can listen for your plain text passwords is certainly also able to listen for hashed passwords and use these captured hashes him/herself to authenticate against your server.

For this matter, more secure authentication protocols usually jump through several hoops in order to make sure, that such a replay attack cannot work, usually, by allowing the client to select a bunch of random bits, which are hashed along with the password, and also submitted in the clear to the server.

On the server:

- generate a few bits of random
- send these bits (in clear text) to the client

On the client:

- generate a few random bits
- concatenate password, the server's random bits and the client random bits
- generate hash of the above
- submit random bits (in clear text) and hash to the server

As the server knows its own random information as well as the client's random bits (it got them as clear text), it can perform essentially the same transformation. This protocol makes sure, that nobody listening in this conversation can use the information later to authenticate falsely using the information recorded

6) Difference Between Hashed and Cryptography Signed Cookies

Encryption is a two-way function; what is **encrypted** can be decrypted with the proper key. **Hashing**, however, is a one-way function that scrambles plain text to produce a unique message digest with a properly designed algorithm, there is no way to reverse the **hashing** process to reveal the original password.

7) Permanent/Time limit/Session cookies in terms of secure authentication

A signed token is a good method for anything where you want to issue a token and then, when it is returned, be able to verify that you issued the token, without having to store any data on the server side. This is good for features like:

- time-limited-account-login;

password-resetting;

- time-limited-form-submission (anti-spam)

Set up expiry time in cookies is good.

For login and password resetting purposes you may want to add an extra factor to the token, a password generation number. You can re-use the salt of the hashed password in the database for this if you like. The idea is that when the user changes passwords it should invalidate any issued tokens (except for the cookie on the browser doing the password change, which gets replaced with a re-issued one). Otherwise, a user discovering their account has been compromised cannot lock other parties out

8) 2FA

2FA, and multi-factor authentication as a whole, is a reliable and effective system for blocking unauthorized access. It still, however, has some downsides. These include

- **Increased login time** – Users must go through an extra step to login into an application, adding time to the login process.

- **Integration** – 2FA usually depends on services or hardware provided by third parties, e.g., a mobile service provider issuing verification codes via text message. This creates a dependency issue, as the enterprise has no means of controlling these external services should a malfunction occur.

- **Maintenance** – Ongoing maintenance of a 2FA system might prove to be a Hard work in the absence of an efficient way of managing a database of users and various authentication methods.

2SV

How to use 2SV

- To use two-step verification, first, you need to know your id and password.
- Your first verification is done by entering your id and password. And we come on second verification.
- The second step of verification can be done by different types of verification like entering the date of birth, verification from the phone, verification to getting OTP on mobile number, verification from your email ID, also verification of from your other device, etc.
- Two-step verification is also used in different apps like Facebook, Snapchat, Instagram, WhatsApp, email, Gmail, etc.

Disadvantages

- The factor can get lost! whenever you need and if you do a mistake, then you guys are logged out from there.

- False Security! When you're trying to log in to your account and if you do a mistake then you will be logged out from there because two-step verification identifies you as a hacker. And if you logged in to your account without any factor then a hacker too.

- The two-step verification is only effective when you take great care of your account ted account.

U2F Universal 2nd Factor

The following are cons of FIDO method Of Authentication
Cost: In the case of FIDO Tokens, they can be quite expensive, costing somewhere in the range of 10-20$ for each token. Individually that amount would not be considered too pricey, but the total price for every employee in a large firm could be quite huge. This cost is mitigated due to the fact that a single token can store multiple keys for various websites and apps.

Extra Steps: While FIDO surely provides additional increased security, this can come at the cost of efficiency and of the most effective use of a user's time. Adding more authentication stages correspondingly increases the overall time and effort necessary for a user to finally become authorized to a use a service. This could become a troublesome issue if a user or multiple users have to authenticate themselves many times in the same day.

MFA Authentication Disadvantages

Multi-factor authentication (**MFA**; encompassing **Two-factor authentication** or **2FA**, along with similar terms) is an electronic authentication method in which a Computer user is granted access to a website or application only after successfully presenting two or more pieces of evidence (or factors) to an authentication mechanism: knowledge (something only the user knows), possession (something only the user has), and inherence (something only the user is). It protects the user from an unknown person trying to access their data such as personal ID details or financial assets.

Disadvantages

- Users may still be susceptible to phishing attacks. An attacker can send a text message that links to a spoofed website that looks identical to the actual website. The attacker can then get the authentication code, user name and password.

- A mobile phone is not always available—they can be lost, stolen, have a dead battery, or otherwise not work.
- Mobile phone reception is not always available—large areas, particularly outside of towns, lack coverage
- Mobile carriers may charge the user for messaging fees

9) BYOA Pros and Cons

Bring your own application provides employee Wireless access to a company's Network for mobile and laptops working from any other location away from the office.

Bring your own applications [(BYOA)] is the trend towards employees' use of third-party apps and cloud services in the workplace.

Advantages

The benefits mentioned consumerization include greater employee participation and satisfaction, as well as improved productivity.

Disadvantages

If a device with confidential information gets stolen, administrators can erase the hard disk remotely. Still, there is no such solution to protect the company in the event of a breach of data security in the environment provided by the Cloud. Another problem associated with consumer technologies in the workplace is the demand for support resources to help a wide variety of non-standardized devices and applications.

Companies can use their organizational guidelines to determine how individual devices can be used to access company data or internal networks**.**

10) Basic Rules for Secure Password Authentication

1. Mark cookies as Secure Makes cookie theft harder
2. Store Password with Salts. It is a random string (series of bits, to be precise, but for the purpose of password storage
3. Use HTTPS
4. Disallow framing (X-Fame-Options: DENY). Otherwise, your website may be included in another website in a hidden iframe and "abused" through JavaScript.
5. Have a same Origin Policy
6. Logout – let your users logout by deleting all cookies and invalidating the session. This makes usage of shared computers safer (yes, users should ideally use private browsing sessions, but not all of them are that savvy)
7. Session expiry – do not have forever-lasting sessions. If the user closes your website, their session should expire after a while. "A while" may still be a big number depending on the service provided.

8. Remember me – implementing "remember me" (on this machine) functionality is actually hard due to the risks of a stolen persistent cookie.
9. Forgotten password flow – the forgotten password flow should rely on sending a one-time (or expiring) link to the user and asking for a new password when it's opened.
10. Limit login attempts – brute-force through a web UI should not be possible; therefore, you should block login attempts if they become too many. One approach is to just block them based on IP.
11. Do not leak information through error messages – you shouldn't allow attackers to figure out if an email is registered or not.
12. For high-risk or sensitive applications use 2 Factor Authentication.

11) SIEM Importance

A **SIEM** solution detects incidents that otherwise can go unnoticed. This technology analyzes the log entries to detect indicators of malicious activity. Moreover, since it gathers events from all sources across the network, the system can reconstruct the attack timeline to help determine its nature and impact.

12) Essential SIEM Tools

1. Solar Winds Security Event Managers
2. Datadog Security Monitoring
3. ManageEngine Event log Analyzer
4. Splunk Enterprise Security
5. OSSEC
6. **LogRhythm NextGen SIEM Platform**
7. AT&T Cybersecurity AlienVault Unified Security Management
8. RSA Net Witness
9. IBM QRadar
10. McAfee Enterprise Security Manager

13) Zero Trust policy

Zero Trust is a security concept that requires all users, even those inside the organization's enterprise network, to be authenticated, authorized, and continuously validating security configuration and posture, before being granted or keeping access to applications and data.

14) SAML, IDP, SP

Security Assertion Markup Language is an open standard for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider. SAML is an XML-based markup language for security assertions.

Security Assertion Markup Language (SAML) is a standard for logging users into applications based on their sessions in another context. This single sign-on (SSO) login standard has significant advantages over logging in using a username/password:

- No need to type in credentials
- No need to remember and renew passwords
- No weak passwords

IDP

An identity provider (IdP or IDP) stores and manages users' digital identities. Think of an IDP as being like a guest list, but for digital and cloud-hosted applications instead of an event. An IDP may check user identities via username-password combinations and other factors, or it may simply provide a list of user identities that another service provider (like an SSO) checks.

What is User Identity

Digital user identity is associated with quantifiable factors that can be verified by a computer system. These factors are called "authentication factors." The three authentication factors are:

- Knowledge: something you know, such as a username and password
- Possession: something you have, such as a smartphone
- Intrinsic qualities: something you are, such as your fingerprint or a retina scan

SP

Service Provider (**SP**) **initiated** SSO(Single Sign On) involves the **SP** creating a SAML request, forwarding the user and the request to the Identity Provider (**IDP**), and then, once the user has **authenticated**, receiving a SAML response & assertion from the IDP. This flow would typically be **initiated** by a **login** button within the **SP**

15) ZERO SIGN OUT

You can log a user out of the Auth0 session and (optionally) from the identity provider (IDP) session. When you're implementing the logout functionality, there are typically three-session layers you need to consider:

Application Session Layer: The first layer is the session inside your application. Though your application uses Auth0 to authenticate users, you'll still need to track that the user has logged in to your application. In a regular web application, you achieve this by storing information inside a cookie.

Auth0 Session Layer: Auth0 also maintains a session for the user and stores their information inside a cookie. The next time a user is redirected to the Auth0 Lock screen, the user's information will be remembered.

Identity Provider Session Layer: The last session layer is the identity provider layer (for example, Facebook or Google). When users attempt to sign in with any of these providers and they are already signed into the provider, they will not be prompted again to sign in. The users may be asked to give permission to share their information with Auth0 and, in turn, your application. It is not necessary to log the users out of this session layer, but you can force the logout.

Redirect users after logout

After users log out, you can redirect users to a specific URL. You need to register the redirect URL in your tenant or application settings. Auth0 only redirects to URLs from the allow list after logout. If you need different redirects for each application, you can add the URLs to your allow list in your application settings.