Beveiliging van JSON Web Tokens in Web Authenticatie

Auteur: Lin Wenhao
Klas: BR2
Datum: 1 mei 2024

Samenvatting

Onderzoeksdoel
In dit onderzoek wil ik meer inzicht krijgen in hoe JWT (JSON Web Tokens) werkt om uiteindelijk de veiligheid van deze authenticatiemethode in het web authenticatie proces te evalueren op basis van de informatie die ik heb. Vervolgens zal ik de gebruiksscenario's samenvatten waarbij JWT wel en niet van toepassing is.

Methodologie
Om een beter inzicht te krijgen in JWT (JSON Web Tokens), zal ik externe bronnen gebruiken en de veiligheid van deze authenticatiemethode voor web authenticatie evalueren op basis van de verzamelde informatie. De bronnen die in het onderzoek worden gebruikt, zullen worden gelabeld in de Bronnenlijst.

Bevindingen

Conclusies

Inhoudsopgave

1. inleiding

In het vierde blok werd ik nieuwsgierig naar JSON Web Token, een authenticatietechniek die ik leerde kennen tijdens een schoolproject. Ik ga de afkorting JWT gebruiken, waar ik nieuwsgierig naar was omdat we in eerdere projecten bij het authenticeren van een gebruiker de ID van de gebruiker direct doorstuurden naar de API-server, maar met JWT gebruikten we een reeks willekeurige tekens zodat de gebruiker de authenticatie-informatie niet kon lezen. Ik was dus benieuwd hoe de API-server de JWT genereerde en parseerde, en hoe het authenticatieproces beveiligt.

Ik zal dus een hoofd vraag in dit onderzoek beantwoorden:

- Hoe beïnvloeden JWT-tokens de veiligheid van netwerkauthenticatie?

Voordat ik de hoofdvraag beantwoord, leid ik naar het antwoord op de hoofdvraag door de volgende 5 deel vragen te beantwoorden:

- Hoe worden JWT-tokens gegenereerd en opgeslagen?
- Wat is het verschil tussen JWT-token en traditionele authenticatiemethoden? (De validatiemethode die we vorig blok gebruikten)
- In welk geval is het gebruik van een JWT-token de meest geschikte keuze?
- Risico's en kwetsbaarheden in JWT-token.
- Omstandigheden waarin JWT-token niet geschikt is voor gebruik, beperkingen.

In het volgende zal ik deze vragen beantwoorden en mijn antwoorden zullen gebaseerd zijn op externe informatiebronnen. Om ervoor te zorgen dat de informatie betrouwbaar is, controleer ik waar mogelijk de betrouwbaarheid van de bronnen en beoordeel ik deze door gebruik te maken van verschillende, maar wederzijds bevestigende informatiebronnen. Ik zal ook mijn eigen begrip van en mening over deze informatie opnemen in mijn antwoorden op de analyse van specifieke vragen. Ten slotte zal ik alle informatiebronnen in een bronnenlijst opnemen.

De volgorde van mijn onderzoek zal beginnen met het verkennen van de deel vragen en eindigen met de hoofd vraag.

2. Generatie en Opslag van JWT

2.1 Generatieproces van JWT

Volgens jwt.io is de introductie over JSON Web Tokens. Het JWT bestaat uit drie delen: 'Header', 'Payload' en 'Signature'.

De header bestaat uit twee delen: het eerste deel bevat het type token en het andere deel is het gebruikte ondertekening algoritme, het specificeert het algoritme dat moet worden gebruikt om het token te genereren of te valideren.

De payload draagt daadwerkelijke informatie, ook wel 'claims' genoemd. Maar er zijn ook 3 soorten 'claims' : geregistreerd(registered), publieke(public), en privé(private) claims. Geregistreerd claims is een declaraties die bijvoorbeeld de geldigheidsperiode van het token aangeven. Publiek claims bevatten informatie die de gebruiker authentiseert, zoals een e-mailadres of gebruikersnaam. Privé claims kunnen alle informatie bevatten, bijvoorbeeld informatie over de instellingsvoorkeuren van de gebruiker.

De handtekening (signature) is het derde deel van JWT dat verantwoordelijk is voor het verifiëren dat er niet met het bericht is geknoeid tijdens de overdracht. Hij doorloopt de header en payload na Base-64 codering en genereert samen met de sleutel een hash waarde die deel uitmaakt van de handtekening.

Header, Payload en Signature worden base64 gecodeerd en het resultaat zijn drie strings die worden samengevoegd met een punt. Zo zie JWT er uiteindelijk uit: xxxxx.yyyyy.zzzzz.

2.2 opslag van JWT

Volgens Auth0, De JWT wordt gegenereerd op de API-server en vervolgens naar de client gestuurd, waarna de webpagina het token opslaat in de lokale opslag. De documentatie vermeldt de volgende risico's van het rechtstreeks opslaan van JWT in lokale opslag (Local Storage):

- Aanvallers kunnen lokaal opgeslagen gegevens verkrijgen via cross-site scripting (XSS).
- Omdat JWT lokaal in platte tekst wordt opgeslagen, is het token beschikbaar voor iedereen met toegang tot de browser.

De documentatie suggereert dus om het token in het geheugen van de browser op te slaan, zoals in een web worker. In de documentatie van Mozilla.org over de beveiliging van web workers staat dat de Worker-interface een thread op systeemniveau creëert en omdat de Worker-thread strak wordt beheerd vanuit andere threads, kunnen kwaadaardige scripts die lijken op XSS-aanvallen hun weg naar de gegevens in de Worker-thread niet vinden.

3. Verschillen in het verificatieproces tussen de JWT en plaintext authenticatie

3.1 Definitie van plaintext authenticatie

Plaintext authenticatie betekent dat gebruikersinformatie direct in het API-verzoek wordt gezet, wat ook betekent dat de gebruikersnaam of het wachtwoord van de gebruiker zichtbaar is.

3.2 Verificatieproces van plaintext authenticatie

Volgens de documentatie van Apinizer (API-beheertool) is in dit geval stuurt de gebruiker een verzoek naar de API-server aan de client kant, en zet meestal zijn gebruikersnaam (authenticatie informatie) in de request header. De API-server voert een verificatieproces uit wanneer het een verzoek ontvangt, dat stappen kan omvatten zoals controleren of de gebruikersnaam geldig en het wachtwoord correct is. Als de authenticatie succesvol is, verwerkt de server het verzoek en stuurt de juiste gegevens terug of voert de juiste actie uit. Als de verificatie mislukt, stuurt de server een foutbericht terug om de client te informeren dat de toegang werd geweigerd of dat geldige verificatiegegevens vereist zijn.

Deze methode is eenvoudig, maar er zijn risico's. Ten eerste kan iedereen de aanvraaginformatie wijzigen om toegang te krijgen tot andere gebruikers. Als gebruiker kan zijn identiteitsinformatie worden afgeluisterd en onderschept omdat deze in duidelijke tekst wordt verzonden, waardoor informatie uitlekt.

3.3 Verificatieproces van JWT

Na het genereren van JWT zijn er drie onderdelen, namelijk “header”, “payload” en “signature”, en wordt gescheiden door een punt (.) Scheid ze om een volledig JWT te vormen. Deze worden omgezet in een base64 URL-string en naar de client gestuurd en opgeslagen in de browser van de gebruiker. De gebruiker stuurt het verzoek met een JWT in de verzoekheader en de server parseert de JWT wanneer hij deze ontvangt. Maar hoe controleert de API de authenticiteit van de JWT?

Volgens een artikel op FreeCodeCamp bestaat er voor het HS265-algoritme een privésleutel die niet alleen kan worden gebruikt om een JWT te genereren, maar ook om de authenticiteit van de JWT te verifiëren. Het genereert een “handtekening” met behulp van de “header” en “payload” van de binnenkomende JWT en de privésleutel, en vergelijkt vervolgens de gegenereerde handtekening met de binnenkomende JWT. De gegenereerde handtekening wordt vervolgens vergeleken met de “handtekening” van het binnenkomende JWT. Als ze overeenkomen, is de JWT geldig.

3.4 samenvatting van verschillen

In de beveiligingsvergelijking tussen deze twee is JWT gebaseerd op ondertekende en versleutelde tokens. Het voorkomt dat anderen de tokeninformatie wijzigen tijdens de overdracht. Plaintext authenticatie heeft geen handtekeningverificatie en kan daarom de authenticiteit van het bericht niet verifiëren. En door de transmissie van duidelijke tekst bestaat er ook een risico op het lekken van informatie.

4. Toepassingsgebieden van JWT

4.1 Verschil tussen Stateloos en Stateful

Volgens een blog, De JWT is een stateloze authenticatiemethode, betekent dat de server de authenticiteit van het token niet hoeft te controleren met de database via een netwerkverzoek bij het uitvoeren van authenticatie. In plaats daarvan verifieert de server de authenticiteit via de handtekening van het token, en de informatie over de gebruiker wordt opgeslagen in de payload van het JWT, zodat de server deze informatie rechtstreeks uit het JWT kan halen zonder de gegevens via database op te vragen.

Het tegenovergestelde van stateloosheid is stateful, waarbij de API-server het token en de identiteitsinformatie van de gebruiker die aan dat token is gekoppeld, opslaat in een database. Wanneer een gebruiker een verzoek verstuurt, vraagt API-server de database naar de geldigheid van het token en stuurt vervolgens de gebruikersinformatie terug die bij het token hoort.

4.2 Voordelen van JWT in scenario's met meerdere servers

Staatloosheid maakt het mogelijk om JWT te gebruiken in applicatiescenario's met meerdere servers. Elke server hoeft alleen de authenticiteit van de JWT te verifiëren zonder rekening te houden met de status van het token, wat betekent dat de API-servers zich allemaal onafhankelijk kunnen authentiseren zonder verzoeken naar de database te sturen voor query's, wat ook de druk op de database sterk vermindert.

5. Risico's en kwetsbaarheden van JWT

Bronnen

- [Introduction to JSON Web Tokens]. jwt.io. URL: https://jwt.io/introduction
- [Token Storage]. auth0. URL: https://auth0.com/docs/secure/security-guidance/data-security/token-storage
- [About Thread Safety]. Mozilla. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API/Using_web_workers#about_thread_safety
- [Plain Text Authentication]. Apinizer. URL: https://docs.apinizer.com/plain-text-authentication-16810822.html
- [How to Sign and Validate JSON Web Tokens – JWT Tutorial]. Kris Koishigawa. December 9, 2022. URL: https://www.freecodecamp.org/news/how-to-sign-and-validate-json-web-tokens/
- [The Purpose of JWT: Stateless Authentication]. Jan Brennenstuhl. Mei 2, 2023. URL: The Purpose of JWT: Stateless Authentication - jbspeakr.cc