Penggunaan Hashing dalam JSON Web Token (JWT) untuk Sistem Autentikasi Pengguna

Ilham Wahabi 13516141
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13516141@std.stei.itb.ac.id

JSON Web Token atau disingkat JWT (baca: jot) merupakan salah satu hal terpenting dalam memastikan keamanan dalam aplikasi kita. Dengan JWT kita bisa memastikan bahwa data yang dikirim maupun diterima merupakan data yang sesuai, dengan cara memverifikasi JWT tersebut. JWT bukan hanya sekedar string acak, tapi lebih dari itu. JWT merupakan Javascript Object Notation (JSON) yang dienkode.

***Kata kunci* — api, enkode, hashing, JSON, situs, token.**

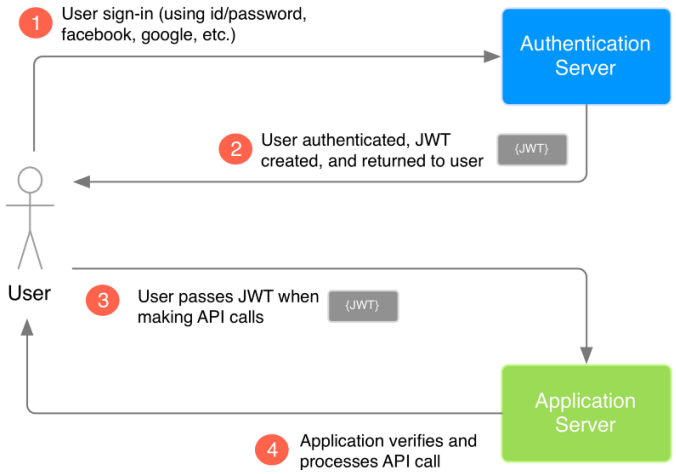
I. PENGENALAN

JSON Object Signing and Encryption group (JOSE) yang dibentuk pada 2011 adalah otak dibalik pengembangan teknologi ini. Tujuan grup tersebut didirikan adalah "standardize the mechanism for integrity protection (signature and MAC) and encryption as well as the format for keys and algorithm identifiers to support interoperability of security services for protocols that use JSON". Orang-orang yang berada dibaliknya antara lain Mike Jones, Nat Sakimura, John Bradley, dan Joe Hildebrand.

JWT sebenarnya merupakan sebuah object JSON yang didefinisikan ulang dalam RFC7519 sebagai cara aman dalam merepresentasikan sebuah set informasi diantara 2 titik. Tokennya sendiri tersusun dari 3 bagin yakni header, payload, dan signature.

JWT juga banyak didukung oleh teknologi server seperti .NET, Python, Java, NodeJS, Javascript, Pearl, Ruby, Elixir, Go, Rust, Lua, Scala, D, Clojure, Objective-C, Swift, C, kdb+/Q, Delphi, PhP, Crystal, dan 1C.

JWT digunakan luas dalam system autentikasi pengguna baik itu situs web maupun aplikasi mobile.



gambaran proses autentikasi

Dalam diagram tersebut digambarkan salah satu contoh penggunaan JWT dalam autentikasi pengguna. Alurnya sebagai berikut :

1. Pengguna masuk ke laman suatu situs
2. Jika informasi yang dimasukkan sesuai dan ada dalam data tersimpan maka pengguna akan terautentikasi dan saat itu juga JWT akan dibentuk dan dikirimkan ke pengguna
3. Saat pengguna melakukan request ke server maka JWT yang pengguna terima tadi haruslah dikirimkan juga saat melakukan request tersebut.
4. Jika tidak ada kesalahan dan JWT yang dikirimkan pengguna juga benar maka akan diberikan respon berhasil dan server akan mengirimkan data yang sesuai.

Apa yang terjadi jika kita tidak menggunakan JWT disini? Itu bisa menjadi celah besar bagi peretas untuk mengirimkan respon yang tak diinginkan, contoh kecilnya saat melakukan request DELETE. Jika peretas bisa mengetahui URL Endpointnya maka tanpa JWT peretas bisa mengirimkan respon secara bebas. Namun karena disini kita menggunakan JWT maka akan ada proses verifikasi dulu apakah yang melakukan request merupakan pengguna yang berhak atau tidak.Jika misalnya JWT tidak sesuai atau tidak dikirimkan makan respon akan ditolak oleh server. Misalnya menentukan apakah dia

merupakan admin atau cuma pengguna biasa atau malah pengguna yang tak terdaftar.

Selain itu kelebihan utama lainnya adalah:

- Kecil

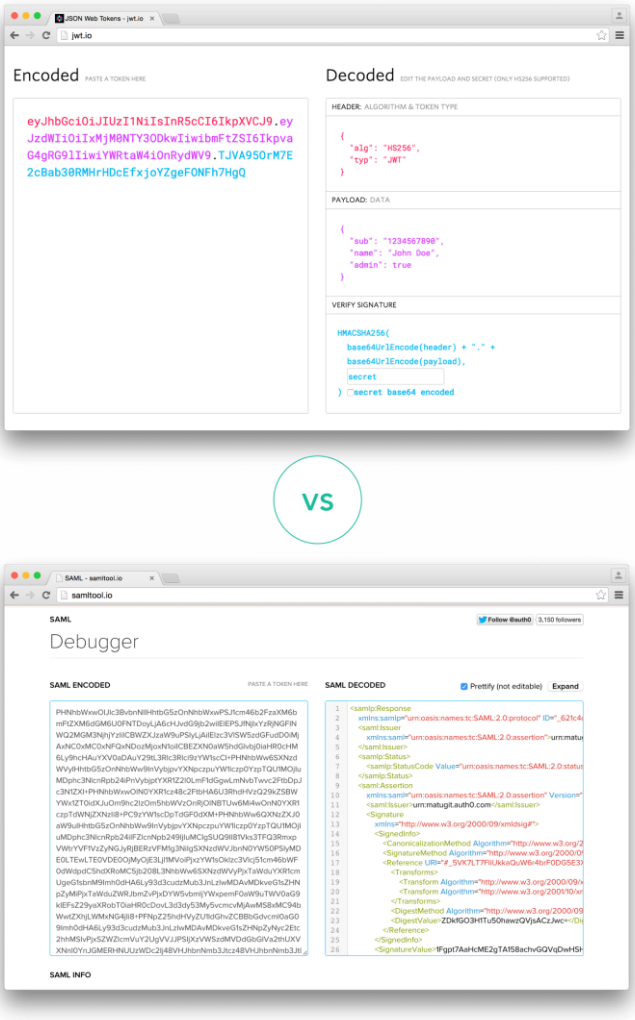
Karena ukurannya yang kecil, lebih kecil dari XML dan SAML, maka transmisi yang dilakukan juga lebih ceapt. JWT bisa dikirimkan melalui URL, parameter POST, atau didalam header HTTP.

- Praktis

Payload sudah mengandung semua informasi yang dibutuhkan sehingga kita bisa menghindari melakukan query ke basis data lebih dari satu kali.

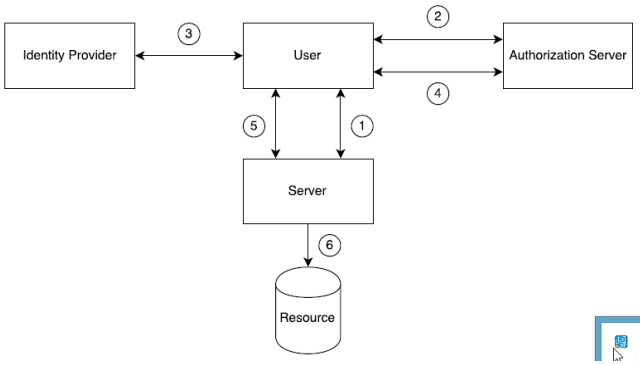
Jika dibandingkan dengan Simple Web Tokens (SWT) dan Security Assertion Markup Languange Tokens (SAML) maka JWT lebih kecil. SWT hanya bisa ditandai secara simetri dengan menggunakan kunci dari algoritma HMAC. Menandai XML dengan menggunakan XML Digital Siganture tanpa mengetahui celah keamanan akan sangat sulit dibandingkan dengan simpelnya menandai JSON.

Apalagi JSON parser merupakan hal lumrah didunia pemograman saking populernya penggunaan JSON. Sayangnya hal tersebut tidak berlaku terhadap XML yang tidak mempunyai pemetaan objek-ke-objek.



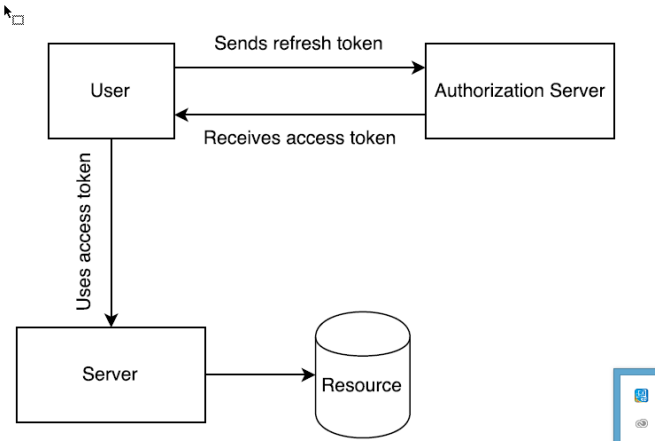
perbandingan JWT dan SAML

JWT juga bisa digunakan di *Federated Identity*



alur federated identitiy

jika bicara mengenai hal ini maka juga tak lepas dengan namanya *refresh token* dan *access token.*



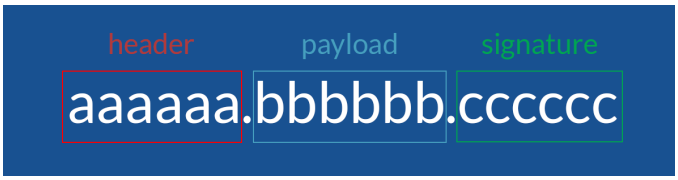
refresh dan access token

Access token adalah token yang diberikan kepada seseorang yang memiliki akses ke sumber terlindungi. Token ini biasanya tidak berumur panjang dan ada tanggal kedaluarsa yang terturlis di token tersebut. Mereka juga bisa saja membawa informasi lain seperti IP address yang menunjukkan darimana request diperbolehkan. Harus diimplementasikan.

Refresh token, dalam arti lain memperbolehkan klien untuk melakukan request access token baru. Sebagai contoh setelah access token kedaluarsa, klien mau melakukan request untuk mendapatkan access token yang baru demi keperluan autentikasi. Maka disini, refresh token diperlukan. Berbeda dengan access token, refresh token biasanya berumur lebih panjang.

II. BAGIAN-BAGIAN JWT

Seperti yang sudah disinggung tadi bahwasanya JWT terdiri dari 3 bagian utama yakni header, payload, dan signature.



A. HEADER

Komponen header dari JWT mengandung informasi tentang bagaimana seharusnya JWT signature dikomputasi. Contoh dari header sebagai berikut :

```
{
    "typ": "JWT",
    "alg": "HS256"
}
```

Dalam hal ini, fungsi dari "typ" adalah mendefinisikan bahwasanya ini merupakan object JWT dan nilai dari "alg" adalah mendefinsikan algoritma yang digunakan untuk memenbtuk komponen signature JWT. Pada contoh diatas kita menggunakan algoritma HMAC-SHA256, algoritma hashing yang menggunakan kunci rahasia untuk mengkomputasikan signature.

Jadi biasanya dilakukan saat pengguna login.

Kita sebenarnya bisa juga menggunakan pasangan kunci/nilai dengan RSA.

B. PAYLOAD

Komponen payload merupakan data yang akan kita simpan dalam JWT kita nantinya. Contohnya sebagai berikut dimana kita menyimpan userId.

```
{
    "userId": "b08f86af-35da-48f2-8fab-cef3904660bd"
}
```

Dalam penggunaannya, tidak hanya userId yang bisa kita simpan. Bisa juga informasi lain seperti umur, lokasi, dsb. Yang perlu diingat semakin banyak data yang disimpan di payload maka sedikit tidaknya akan mempengaruhi terhadap kinerjanya sendiri saat digunakan.

Ada 3 jenis klaim.

Pertama *registered claims* yang sering digunakan yakni

- iss untuk issuer,
- exp untuk expiration time,
- sub untuk subject,
- aud untuk audience,
- nbf mendefinisikan kapan waktu JWT tidak boleh diterima untuk diproses,
- iat untuk issued at, mendifinisikan kapan JWT ini diciptakan,
- jti mendefinisikan pengenal unik untuk JWT, biasanya digunakan untuk mencegah terbentuknya token yang sama kedepannya. Jadi semua token digunakan 1 kali.

Kedua *public claims* adalah yang kita buat sendiri seperti nama pengguna, umur, dan informasi lainnya.

Ketiga *private claims*, adalah klaim yang disetujui produser dan consumer untuk dijadikan private, biasanya subjek yang digunaka untuk kolisi. Jadi gunakan dengan bijak.

C. SIGNATURE

Pseudo code yang digunakan adalah sebagai berikut :

```
//signature algorithm

data = base64urlEncode( header ) + "." + base64urlEncode( payload )

signature = Hash( data, secret );
```

Yang dilakukan dalam langkah ini adalah program yang mengenkode header dan payload tadi kemudian keduanya digabungkan dengan menggunakan pemisah berupa titik. Kemudian lagi string yang telah digabung ini kemudian dimasukkan ke data. Data yang disimpan ini kemudian dihashing dengan menggunakan metode yang telah kita tentukan di header tadi (HMAC-SHA256).

Berikut merupakan header dan payload yang telah dienkode.

```
// header
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9

// payload
eyJ1c2VySWQiOiJiMDhmODZhZi0zNWRhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ
```

Maka didapatkan signature sebagai berikut.

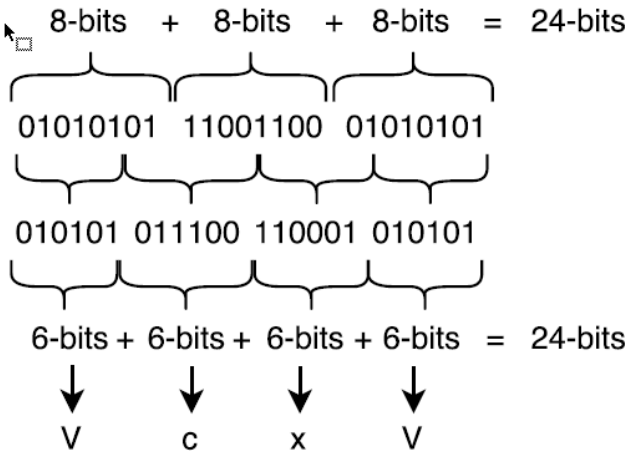
```
// signature
-xN_h82PHVTCMA9vdoHrcZxH-x5mb11y1537t3rGzcM
```

Jika sudah mendapatkan ketiganya kemudian kita harus menggabungkan semuanya kedalam satu string utuh yang dipisakan satu sama lain dengan titik.

```
// JWT Token
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ1c2VySWQiOiJiMDhmODZhZi0zNW
RhLTQ4ZjItOGZhYi1jZWYzOTA0NjYwYmQifQ.-xN_h82PHVTCMA9vdoHrcZxH-
x5mb11y1537t3rGzcM
```

Sekarang JWT tersebut sudah bisa kita gunakan untuk dikirimkan ke pengguna.

Base64 merupakan algoritma enkode binary-to-text. Tujuan utamanya yaitu merubah beberapa sekuens dari octet menjadi sekuens dari karakter printable. Dalam matematika Base64 merubah sebuah sekuens dari angka radix-256 ke sekuens angka radix-64. Kata ‘base’ bisa digunakan sebagai tempat untuk radix, daripada untuk nama algoritma.



enkoding Base64

III

PERTIMBANGAN KEAMANAN

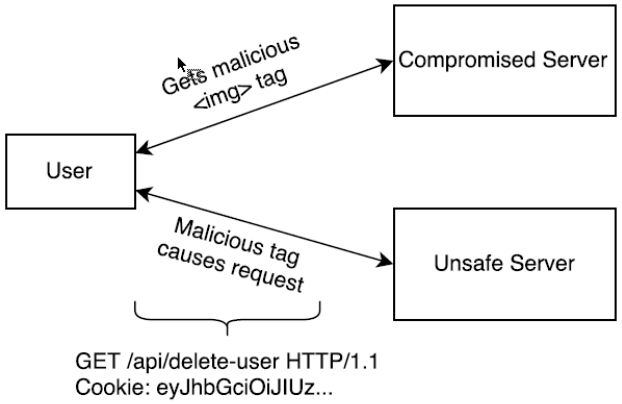
a. Signature Stripping

Intinya dengan cara menghapus signature dan mengklarifikasi bahwasanya JWT belum ditandai. Penggunaan library yang tidak sesuai bisa mengakibatkan kita terserang.

alur serangan signature scripting

b. Cross-Site Request Forgery

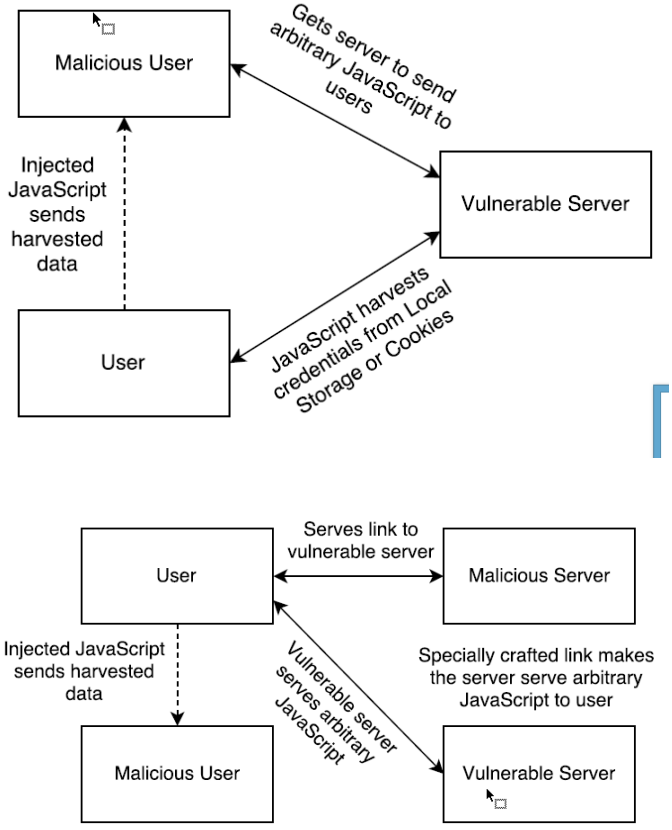
Menyerang situs dimana penggunanya sedang login. Menipu pengguna tersebut agar mengirimkan request dari situs lain.



Alur serangan X-Site Request Forgery

c. Cross Site Scripting

Menginjeksikan javascript ke situs terpercaya.



Alur serangan X-Site scripting

d. Token yang Tidak Aman

Salah satunya kita tidak mendefinisikan header dengan benar. Namun cara ini masih dibenarkan jika ini hanya untuk konsumsi pribadi.

```
{
    "alg": "none"
}
```

Sehingga hanya akan ada 2 string yang dikonkatenasi.

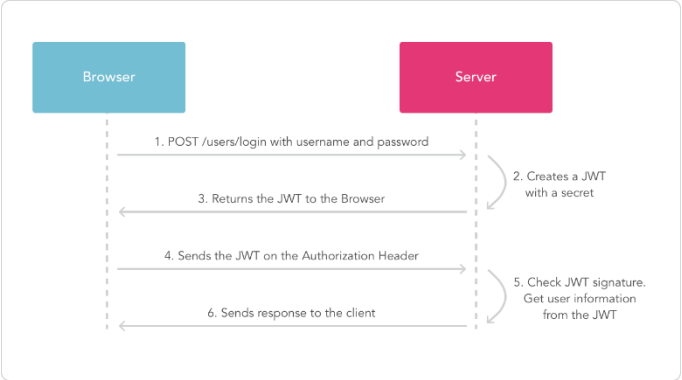
```
eyJhbGciOiJub25lIn0.
eyJzdWIiOiJ1c2VyMTIzIiwic2Vzc2lvbiI6ImNoNzJnc2IzMjAwMDB1ZG9jbDM2M
2VvZnkiLCJuYW1lIjoiUHJldHR5IE5hbWUiLCJsYXN0cGFnZSI6Ii92aWV3cy9zZXR0aW5ncyJ9.
```

IV

KESIMPULAN

Dalam memverifikasi, karena server kita sudah mengetahui algoritma apa yang digunakan dalam mengenkode dan mengetahui kunci rahasinya juga, maka hal tersebut dapat digunakan pula saat memverfikasi respon. Jika tidak tepat menandakan ada kesalahan atau serangan ke server kita. Jadi seolah-olah kita menambahkan pelindung diantara klien dan server.

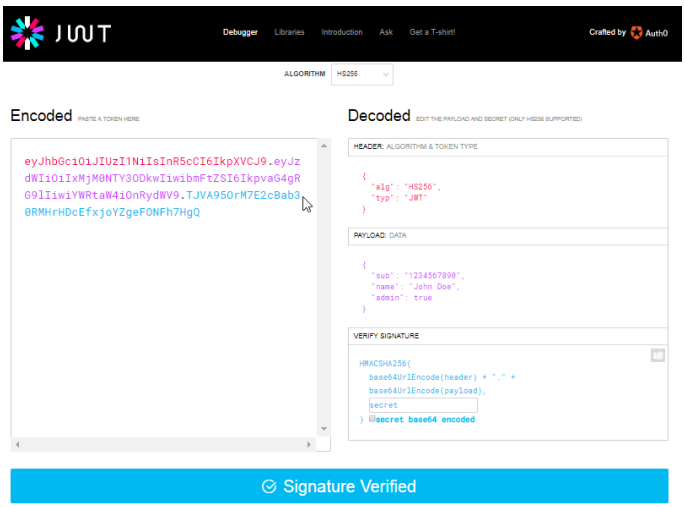
Yang patut diketahui adalah tujuan utama dari JWT bukanlah untuk menyembunyikan data maupun untuk menyamarkannya tapi lebih ke arah memastikan bahwa pengguna yang melakukan rekues merupakan pengguna yang berhak.



JWT digunakan dalam pertukaran informasi

Juga dalam pembentukan JWT kita sama sekali tidak melakukan enkripsi tapi encode dan penandaan. Tujuan dari encode adalah mengubah data jadi dengan mudah dan aman sehingga dapat dikonsumsi oleh berbagai tipe sistem. Penandaan pada data mengakibatkan penerima data nantinya dapat memverifikasi sumber datangnya data tersebut. Jadi karena datanya tidak dienkripsi JWT tidak bisa menjamin keamanan yang tinggi untuk data sensitif.

Untuk sekedar melihat-lihat ataupun sekedar mencoba-coba cara kerja dari JWT ini bisa dilakukan di jwt.io



V

REFERENSI

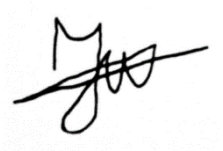
[1] https://en.wikipedia.org/wiki/JSON_Web_Token
[2] https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcecH. Poor, *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.
[3] https://jwt.io/introduction/
[4] https://www.codementor.io/olatundegaruba/5-steps-to-authenticating-node-js-with-jwt-7ahb5dmyr
[5] https://auth0.com/learn/json-web-tokens/
[6] Peyrott, Sebastian. JWT Handbook. Auth0 Inc. 2016.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 3 Desember 2017

Ttd (scan atau foto ttd)



Ilham Wahabi
13516141