

الفصل الثاني و العشرين

ثغرة ال JWT

المؤلف

د.م/ أحمد هاشم الفقي

استشاري أمن المعلومات و التحول الرقمي

Ahmed.Hashem.ElFiky@outlook.com

ثغرة ال JWT

- عند تسجيل الدخول في المواقع العادية تعودنا على استعمال جلسة session تربط السرفر بالمتصفح وهذه الطريقة تسمح بابقاء المستخدم متصل بحسابه account مع السرفر
- ولكن هذه الطريقة غير ممكنة عندما نفكر بانشاء تطبيق اندرويد او تطبيق سطح مكتب مبرمج بالجافا او السي شارب مثلا لان هذه التطبيقات اساسا لاتعتمد على جلسات sessions الذي يفهمها المتصفح
- ففي هذا النوع من التطبيقات سنحتاج الى شيء اخر غير الجلسات المربوطة في عملية تسجيل الدخول أو استخدام صلاحيات المستخدم (ادمن .يوزر . مشرف..) سنحتاج لاستخدام مفهوم جديد اسمه ال token او مايسمى بالرمز السري هذا الرمز يقوم السرفر بتوليده عند كل عملية تسجيل دخول المستخدم ثم ارساله الى التطبيق سواء تطبيق اندرويد او تطبيق سطح مكتب او اي شيء..

ثغرة ال JWT (تكملة...)

- وهذا التطبيق يقوم بحفظه بذاكرته الداخلية وسيحتاج لإعادة ارساله الى السرفر عند كل عملية سرية تحتاج لصلاحيات (مثل الاطلاع على بيانات المستخدم او تعديل بيانات المستخدم...) فهذه العمليات تتطلب هذا ال token والذي يقوم السرفر من فك تشفيره وينظر هل هو صحيح؟ هل مازالت مدته صالحة؟ فادا وجد السرفر ان هذا الكود المرسل من التطبيق صحيح سيسمح للتطبيق بالوصول الى البيانات السرية.. اما اذا وجد ان الكود خطأ او انه لم يرسل اليه سيقوم مباشرة بالرد الى التطبيق انه لايسمح له بالدخول للصفحة الفلانية

- يجب ان نفهم الفرق بين session و token وبين session فال session تستخدم بين المتصفح (المستخدم) وبين الخادم (server) اما token فهو كود مشفر يحتوي على بيانات المستخدم يتبادلها المستخدم (التطبيق ايا كان نوعه) وبين السرفر للتعرف على بعضهما البعض ..

ثغرة ال JWT (تكملة...)

- فمهمة الجلسات والرموز السرية هي نفسها ولكن تختلف مواضع استخدامها حسب نوع التطبيق
- سوف نشرح في هذا الفصل اداة jwt وهي اداة شهيرة ذات مقاييس عالمية او يمكنك ان تقول مكتبة تسمح لك بتشفير ال token وفك تشفيره بالسرفر واعطاء مدة صلاحية زمنية محددة سنحددها كما نشاء
- من الناحية التقنية يعتبر JSON Web Token أو JWT معيار مفتوح يحمل رمز RFC 7519 يحدد طريقة مدمجة و مكتفية ذاتيًا لنقل المعلومات بأمان بين الأطراف (client/server) ككائن من نوع JSON

ثغرة ال JWT (تكمّله...)

- دعنا نشرح بعض المفاهيم:

- مدمج :

- نظرًا لصغر حجمها ، يمكن إرسال JWT عبر عنوان URL أو معلمات POST أو رؤوس HTTP بالإضافة إلى ذلك ، كلما صغر الحجم ، زادت سرعة الإرسال.

- مكتفية ذاتيا:

- تحتوي الحمولة (Payload) على جميع المعلومات الضرورية حول المستخدم ، وتجنب الاستعلامات المتعددة لقاعدة البيانات.

- سيناريوهات JWT المعمول بها

- المصادقة :

- هذا هو الاستخدام الأكثر شيوعًا لـ JWT بمجرد تسجيل دخول المستخدم ، سيتضمن كل طلب لاحق JWT، مما يسمح للمستخدم بالوصول إلى المسارات والخدمات والموارد التي يسمح بها الرمز المميز. الدخول الموحد هو إحدى ميزات JWT المستخدمة على نطاق واسع اليوم بسبب انخفاض النفقات العامة وسهولة الاستخدام عبر المجالات المختلفة.

ثغرة ال JWT (تكملة...)

- تبادل المعلومات :

- تعد JSON Web Tokens طريقة جيدة لنقل المعلومات بشكل آمن بين الأطراف. لأن JWT يمكنه التوقيع Signature : على سبيل المثال ، باستخدام أزواج المفاتيح العامة / الخاصة ، يمكن تحديد أن المرسل هو الشخص الذي يدعي أنه هو. بالإضافة إلى ذلك ، نظرًا لاستخدام الرأس Header والحمولة الصافية Payload لحساب التوقيع ، يمكنك أيضًا التحقق من أن المحتوى لم يتم العبث به.

- هيكل JWT

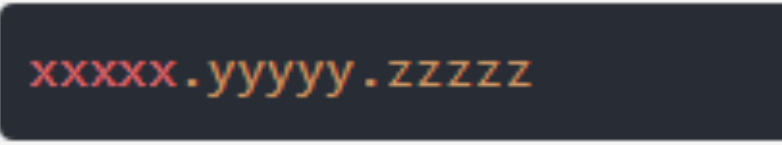
- في شكل مضغوط ، يحتوي JWT على ثلاثة أجزاء مفصولة بنقاط (.) ، وهي:

- Header

- Payload

- Signature

- عادة ما يكون هيكل JWT على النحو التالي:



xxxxx.yyyyy.zzzzz

ثغرة ال JWT (تكملة...)

- يتكون JWT من ثلاث اجزاء يفرق بينها علامة DOT وهي (Header + Payload + Signature)

Header	جزء مخصص يقوم بتعريف نفسه للسيرفر, اظهار نوعه, الهاشيق اللي يستخدمه, الخ... (هذه المعلومات تكون متاحة للقراءة من اي شخص)
Payload	هو الجزء الذي يحتوي على المعلومات (claims) المأخوذة من الكلاينت للسيرفر و هو اللي يحتوي على المحتوى اللي يتحقق منه السيرفر من أجل المصادقة (هذه المعلومات تكون متاحة للقراءة من اي شخص)
Signature	جزء مهمته يقوم بأخذ (Header + Payload) ويقوم بتطبيق خوارزمية HMAC على الجزئين, لو قام المهاجم بتغيير Payload سوف يتنافى مع Signature ويرد له بالرفض التام ف هنا نفهم ان وظيفته هي التأكيد على المصادقة (هذه المعلومات غير متاحة للقراءة من اي شخص)

ثغرة ال JWT (تكملة...)

- فيما يلي نقدم هذه الأجزاء الثلاثة بشكل منفصل:

- Header

- يتكون العنوان عادةً من جزأين: نوع الرمز المميز ، أي JWT و نوع الخوارزمية المستخدمة في عملية التوقيع ال Signature و هي المثال ده HMAC HS256

- على سبيل المثال:

```
1 | {  
2 |   "alg": "HS256",  
3 |   "typ": "JWT"  
4 | }
```

- يتم ترميز JSON الخاص بجزء الرأس بواسطة Base64Url لتكوين الجزء الأول من JWT

ثغرة ال JWT (تكملة...)

- payload هو المقطع الثاني من الكود وهو الذي يركب البيانات الفعلية التي نحتاج اليها أي بمعنى أوضح هي البيانات المشفرة التي نريد ارسالها . يمكن أن يكون معلومات المستخدم مثل معرف المستخدم username والاسم والبريد الإلكتروني ... الخ (باختصار ال payload هو كود البيانات التي نرسلها او نستقبلها)
- يحتوي على

ISS	المصدر	هي لتعريف المصدر للسيرفر
Sub	الموضوع	هي لتعريف اسم الموضوع او الهدف من المصادقة للسيرفر
aud	التعرف على المستلم	الجمهور
exp	وقت انتهاء صلاحية token	وهو يحدد متى تنتهي صلاحية JWT
nbf	بداية صلاحية token	تحدد متى يبدأ صلاحية JWT
iat	Timestamp	فقط ضع Timestamp

ثغرة ال JWT (تكملة...)

```
{  
  "iss": "exportdeveloper.com",  
  "exp": 1426420800,  
  "company": "export developer",  
  "awesome": true  
}
```

- مثال على Payload
- يتم ترميز JSON الخاص بجزء Payload بواسطة Base64Url لتكوين الجزء الثاني من JWT

ملحوظة: على الرغم من أن هذه المعلومات محمية من العبث ، يمكن لأي شخص قراءتها. ما لم يتم تشفيرها ، لا تضع معلومات سرية في عناصر الحمولة Payload أو رأس Header الخاص بال JWT

ثغرة ال JWT (تكملة...)

- signature أو التوقيع . يتم إنشاؤه من خلال الجمع بين Header المشفر و Payload المشفر وتوقيعه باستخدام خوارزمية تشفير قوية ، مثل HMAC SHA-256. يحتفظ الخادم (server) بالمفتاح السري الخاص بالتوقيع بحيث يتمكن من التحقق من الرموز المميزة الحالية وتوقيع رموز جديدة.

```
1 HMACSHA256(  
2   base64UrlEncode(header) + "." +  
3   base64UrlEncode(payload),  
4   secret)
```

ثغرة ال JWT (تكملة...)

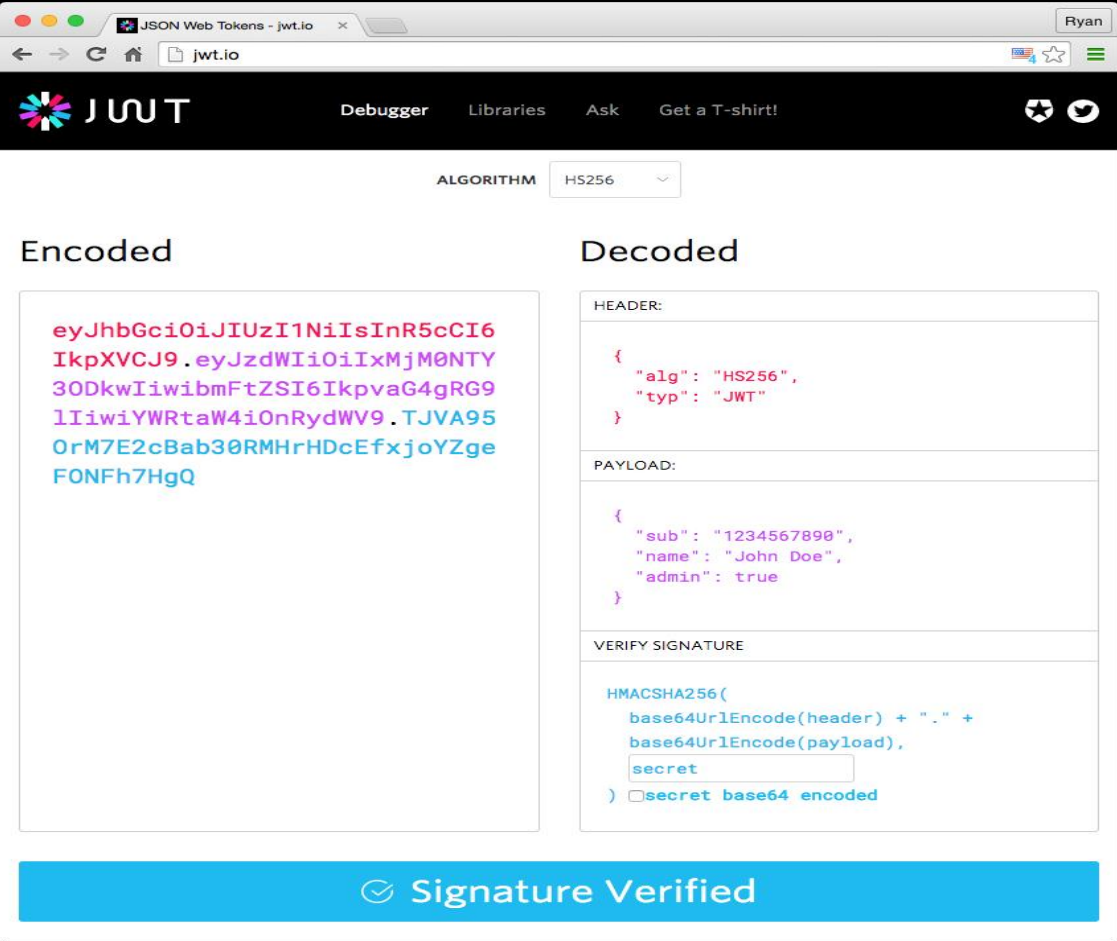
- ممارسة JWT

- ناتج JWT هو ثلاث سلاسل Base64-URL مفصولة بنقاط ، والتي يمكن تمريرها بسهولة في بيئات HTML و HTTP ، وهي أصغر حجمًا من المعايير القائمة على XML مثل SAML
- مثال JWT التالي ، الذي يحتوي على ترميز البيانات والحمل السابق ، يستخدم المفتاح السري للتوقيع.

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4  
gRG9lIiwiaXNTb2NpYWwiOnRydWV9.  
4pcPyMD09o1PSyXnrXCjTwXyr4BsezDI1AVTmud2fU4
```

ثغرة ال JWT (تكملة...)

- يمكننا استخدام المصحح jwt.io لفك شفرة JWT والتحقق منها وإنشائها:



The screenshot shows the JWT.io website interface. At the top, there's a navigation bar with the JWT logo, a 'Debugger' tab, and links for 'Libraries', 'Ask', and 'Get a T-shirt!'. Below the navigation bar, the 'ALGORITHM' is set to 'HS256'. The 'Encoded' section displays a long string of base64-encoded characters. The 'Decoded' section shows the token's structure, including the 'HEADER' and 'PAYLOAD'. The 'VERIFY SIGNATURE' section shows the HMACSHA256 algorithm being used to verify the token's signature. A blue banner at the bottom indicates 'Signature Verified'.

Encoded

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiYXNjaWV9.TjVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ
```

Decoded

HEADER:

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

PAYLOAD:

```
{  "sub": "1234567890",  "name": "John Doe",  "admin": true}
```

VERIFY SIGNATURE

```
HMACSHA256(  base64UrlEncode(header) + "." +  base64UrlEncode(payload),  secret  )
```

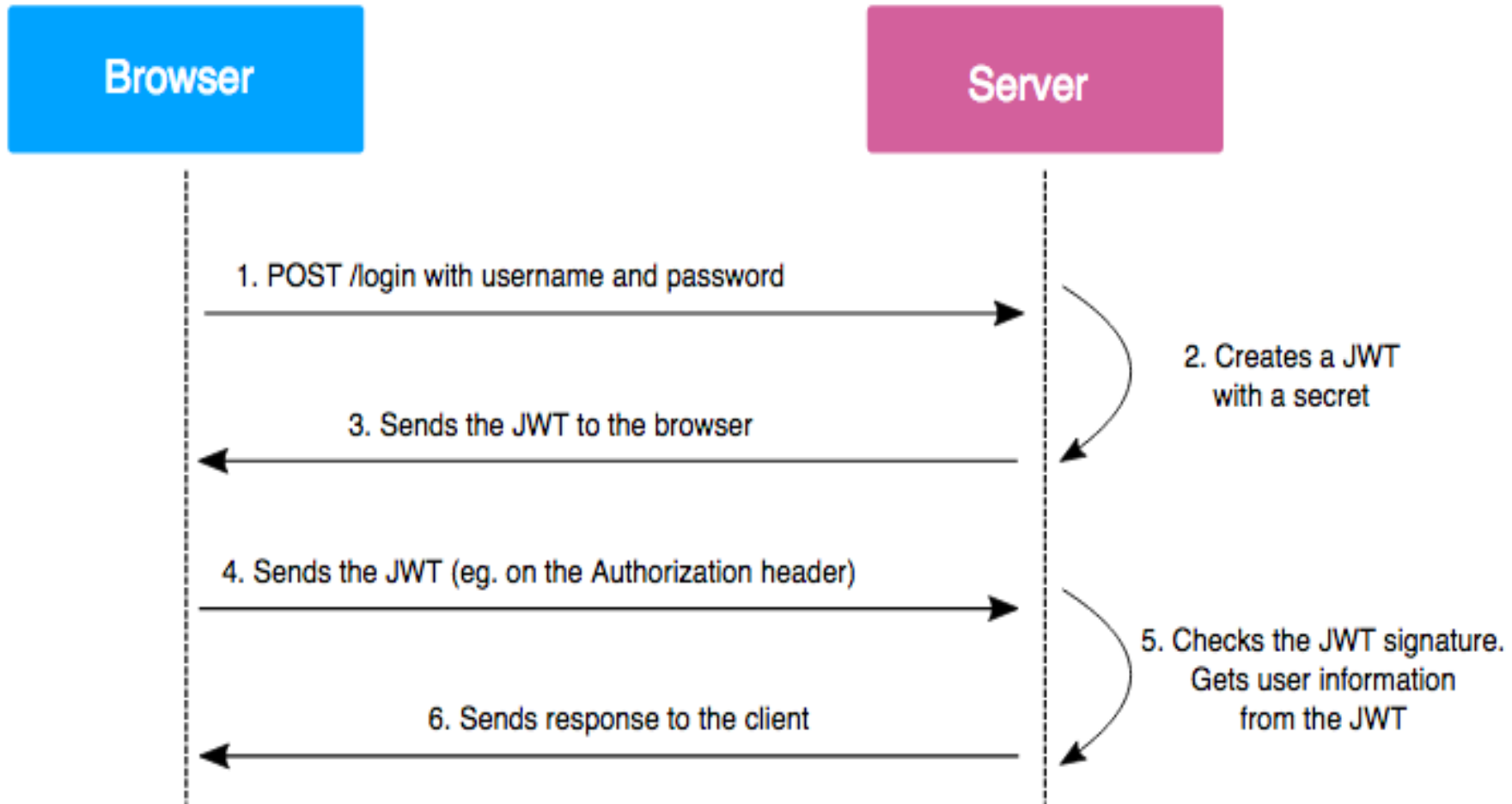
Signature Verified

ثغرة ال JWT (تكملة...)

- مبدأ عمل JWT
- في المصادقة ، عندما يسجل المستخدم الدخول بنجاح باستخدام بيانات الاعتماد الخاصة به ، سيتم إرجاع JSON Web Token ويجب تخزينه محلياً (عادةً في التخزين المحلي ، ولكن يمكن أيضاً استخدام ملفات تعريف الارتباط) بدلاً من إنشاء خادم جلسة بالطريقة التقليدية إعادة ملف تعريف الارتباط.
- عندما يريد المستخدم الوصول إلى مسار أو مورد محمي ، يجب على وكيل المستخدم المتصفح استخدام نظام الاستضافة لإرسال JWT ، عادة يكون في رأس الطلب Authorization Bearer

```
Authorization: Bearer <token>
```

ثغرة ال JWT (تكملة...)



ثغرة ال JWT (تكملة...)

• أسئلة:

① هل JWT آمن؟

• لا لأن طريقة الترميز Base64 قابلة للعكس ، أي أنه يمكن تحليل محتوى الرمز الصادر من خلال الترميز. بشكل عام ، لا نوصي بوضع معلومات حساسة في الحمولة Payload ، مثل كلمة مرور المستخدم.

② هل يمكن تزوير محتويات JWT Payload؟

• لا لأن أحد مكونات JWT هو Signature ، والذي يمكن أن يمنع تعديل محتوى الحمولة Payload. لأن التوقيع يتكون من Base64 مع رأس Header وحمولة Payload .

ثغرة ال JWT (تكملة...)

احد الاستغلات التي تتم على هذه الثغرة هو وضع اجورزم التوقيع Signature ب none و تغيير في ال Payload كما في المثال

Request

Raw Params Headers Hex JSON Web Tokens

```
Headers = {  
  "alg" : "none",  
  "typ" : "JWT"  
}  
  
Payload = {  
  "sub" : "1234567890",  
  "name" : "John Doe",  
  "admin" : true  
}  
  
Signature = ""
```

- ☐ Do not automatically modify signature
- ☐ Recalculate Signature
- ☐ Keep original signature
- ☒ Sign with random key pair

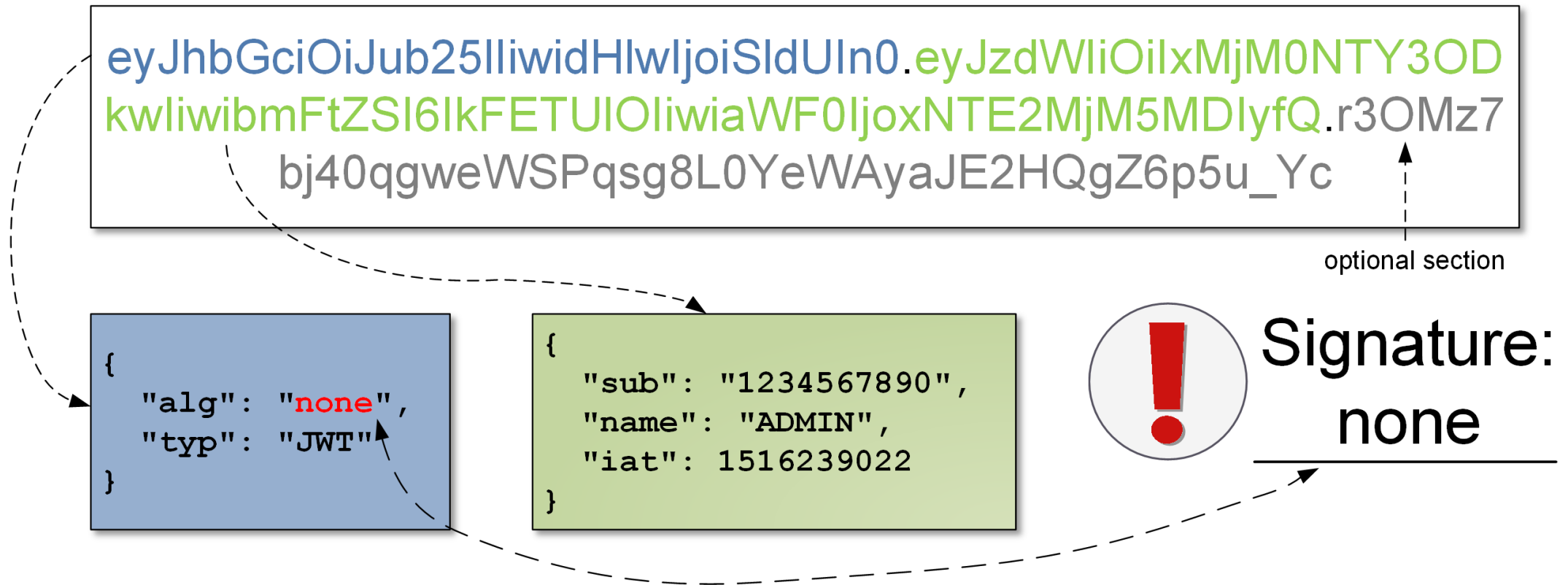
Secret / Key for Signature recalculation:



Alg None Attack:

Alg: none

ثغرة ال JWT (تكملة...)



ثغرة ال JWT (تكملة...)

احد الاستغلات التي تتم على هذه الثغرة ايضا هو وضع اجورزم للتوقيع Signature في Header و تغيير في ال Payload و لكن حذف Signature Section كما في المثال

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiJlMjM0NTY3ODkwIiwibmFtZSI6IjE2MzQ1Njc4OTAiLCJpYXQiOiIxNTY2MzQ1NjIyIn0.

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

```
{  
  "sub": "1234567890",  
  "name": "ADMIN",  
  "iat": 1516239022  
}
```



No signature section

تم بحمد الله انتهاء الفصل الثاني و العشرين