JWT

تعريف

- الـ JWT هو إختصار لـ Json Web Token وده عبارة عن JSON Format لنقل الداتا بين طرفين سواء بين عميل وسيرفر أو سيرفر وسيرفر أو عميل وعميل، وشرط تكون الـداتا معمول لها encode64.

  - الـ JWT بتُستخدم كتير ف الـ Authentication, Session management, and access control، وبالتالي الثغرات الي في الـ JWT هتضر بالموقع كامل.
  - الـ JWT بيِّستخدم لنقل الداتا زي ما قولنا، وأي نوع من الداتا، ولكن الأشهر إنها بتُستخدم لنقل info (claims عن اليوزر في الـ Authentication وغيرها.
  -

- الـ JWT بيتكون من 3 حاجات وهما الـ Header.Payload.Signature

- الـ Header عبارة عن metadata عن الـ token نفسه، بيتكون من جزئين الجزء الأول هو نوع تشفير الـ signature والجزء الثاني هو نوع التوكين وغالبا بيكون JWT.

- الـ payload عبارة عن الداتا المرسلة (claims) وبعض المعلومات الإضافية زي مدة انتهاء التوكين وهكذا.

- الـ signature عبارة عن

  - الـ Header encoded base64URL.
  - Payload encoded base64URL
  - secert

وكل ده بيتم تشفيره بطريقة محددة (نوع التشفير بيتم تحديده في الجزء الأول من الـ Header)، واحيانا بيعمل encode للـ hash result، وده عشان يتأكد إن الداتا محصلش فيها تغيير.

- الـ JWT أي حد يقدر يعملها decode ويقرأ الداتا الي فيها وكمان يقدر يعدل عليها، ده بيعتمد على الـ signature المُستخدمة.

JWT | JWS | JWE

- الـ JWT مش entity مستقل لوحده، هو مجرد claims بيحصلها نقل بين طرفين، ف هو مواصفاته محدودة، ولكن بتزيد مواصفاته لما بندمج مواصفات الـ JSON web Signature والـ JSON web Encryption معاه.
- بإختصار الـ JWT هو عبارة عن JWS أو JWE، لما الناس بتستخدم كلمة JWT فهي ف الغالب تقصد الـ JWS
- الـ JWS متشابهة جداً، ولكن المختلف إن الـ Content بيكون encrypted مش مجرد encoded.

كيفية عمله

- بيسجل اليوزر الدخول باليوزر نيم والباسورد
- بيتحقق السيرفر من الـ credentials وبعدين بيبعت الـ JWT للمتصفح وبيتم تخزينه في الـ Authorization Header.
- لما اليوزر يدخل على صفحة حساسة بيتم إرسال الـ JWT مع الريكوست عن طريق Authorization Header بالشكل التالي Authorization: Bearer <Token>
- بيتحقق السيرفر من الـ Signature وبعدين بيرجع الريسبونس للكلاينت.

ما هي هجمات JWT

- الـ JWT attacks هي عبارة عن modified token بيبعته الـ attacker للسيرفر بهدف إنه يعمل bypass للـ authentication, access control بإنه ينتحل شخصية يوزر هو authenticated اصلا.

إيه الـ Impact بتاع الـ JWT attacks

- لو الـ attacker قدر يعمل valid token بقيم عشوائية، ف ممكن يقدر يعلي الصلاحيات بتاعته، أو ينتحل شخصية يوزر تاني وبالتالي هتكون account take-over.

الهجمات المحتملة

- أحيانا السيرفر بينشئ التوكين ولكن مش بيحفظ أي معلومات عن التوكين الي عمله إنشاء، وبالتالي هو ميعرفش المعلومات الصحيحة ولا الـ signature للتوكين ده، ف حتى لو حصل تغيير في البيانات والـ Signature مبقتش valid ف هيقبل الريكوست لإنه مش عارف الـ valid signature إيه اصلا، ف أي تغيير في البيانات هيعدي.
- 1. في بعض المكتبات للـ JWT بتستخدم فقط طريقة للـ decode ومش بتعمل verify للـ singature.

2- تسرب المعلومات المهمة: وده بيحصل لما الـpayload بيكون plaintext وبيحمل معلومات حساسة (يعني الـ alg بتكون none لوحدها),أو بمجرد ما تعمل decode للـ payload بتلاقي معلومات مهمة.

2- تعديل الـ Signature alg لـــ none, لإن بعض مكتبات الـ JWT بتدعم الـ none alg وبالتالي السيرفر مش هيقدر يتأكد من وجود تغيير ولا لا

· الـ signature الي موجودة في الـ JWTs عشان تمنع اليوزر من إنه يعدل في الداتا والسيرفر يكون مطمن إن مفيش حاجة هيحصلها تعديل ف كل الـ JWTs المفروض يكون معمولهم sign.

· فالـ Header JWT بتتحدد الـ alg الي هتعمل sign للـ data.

· بعد ما بتخلي الـ alg=none بتروح تمسح الـ Signature وتبعت الريكوست عادي.

- أحيانا مش بينفع مجرد تعديل البيانات أو تغيير الـ alg=none، ف مش قدامك غير إنك تعرف الـ secret عشان تـ signature valid token وتكون معدل البيانات.
- أحيانا الديفلوبر بياخد الكود كوبي بيست من الانترنت وبالتالي مش بيغير الـ secret في الكود بتاعه، ف ممكن يكون حاجة شائعة أو معروفة، أو مستخدم كتير قبل كده في أكتر من موقع.
- 3. تالت سيناريو إنك تعمل brute force للـ secret عن طريق أداة hashcat بالأمر ده:
  - `<hashcat -a 0 -m 16500 <jwt> <wordlist`
  - بعد ما بتلاقي الـ Key بتعدل البيانات براحتك وبعدين بتعمل sign للتوكين وتبعت الريكوست عادي.
- طبقا لمواصفات الـ JWS، فالـ alg header هو الوحيد المهم، ولكن من الناحية العملية ف الـ JWT headers بتحتوي على parameters تاني، وفي منهم مهم للـ Attackers زي:
  - مJWK: وده بيمثل الـ Key.
  - مJKU: وده عبارة عن URL بيستخدمه الـ servers علشان توصل للـ Keys الي فيهم correct key.
  - مKid: وده عبارة عن Key id بيستخدمه الـ servers عشان يعرفوا الـ correct key لما يكون في مجموعة keys.
- طبعا دول user-controllable parameters ودورهم إنهم يقولوا للسيرفر إيه الـ Key الي تستخدمه عشان تعمل signature للـ token، وأنت بتستغل ده عن طريق إنك تعمل RSA private key وتحط الـ matching key بتاعه ف الـ JWK header.
- الأفضل طبعا إن السيرفر يكون عنده whitelist بالـ public keys الي هيستخدمها عشان يعمل verify للـ signature، ولكن الـ misconfigured servers بتستخدم أي key بيكون ضمن الـ JWK parameter.

- 4. في السيناريو ده السيرفر بيكون بيـ support الـ JWK وبيكون موجود الـ Kid header، ف عشان تستغل ده لصالحك، بتروح تنشأ Key على حسب الـ alg المستخدمة وفالغالب RSA-key وبعدين بتضيف الـ JWK header وبتخلي قيمة الـ Kid داخل الهيدر ده زي قيمة الـ Kid header الأساسي وبعدين بتعمل sign للـ token وتبعت الريكوست :D.
- 5. لو مش هيكون JWK direct ف هيستخدم JKU عشان يوصل للـ keys وهيختار الي يتطابق مع الـ kid ولاحظ إن الصفحة الي هيجيب منها لازم تنتهي /.well-known/jwks.json
- 6. أحيانا الـ kid بياخد قيمته من Local file يكون فيه verification keys، أو ريكورد من الداتا بيس، ف هنا بتخليه يجيب قيمة ملف /dev/null والي قيمته بترجع بـ null وبالتالي الـkey هيكون encod64 null byte (AA==)I وبكده أنت قدرت تعمل verify للـ signature.
  - لو الـ server بيجيب الـ verification keys من الداتا بيس، ف الـ kid هنا مصاب بالـ SQL injection

Algorithm confusion attacks

- الهجوم ده بيحصل لما الـ attacker يقدر يخلي الـ server يستخدم alg مختلفة غير الي الديفلوبر متوقعها.
- الهجوم ده بيحصل بسبب أخطاء في تنفيذ بعض مكتبات الـ JWT، وكل عملية Verification بتختلف عن التانية طبقا للـ algorithm المستخدمة، ولكن في بعض المكتبات بتستخدم طريقة واحدة لعملية الـ verification، والطريقة دي بتعتمد على الـ alg header الي بيجي مع الـ token وبيحدد طريقة الـ verification هتتم ازاي.
- الخطأ الي بيحصل بيكون بسبب كود زي ده:

```
}(function verify(token, secretOrPublicKey

    ;()algorithm = token.getAlgHeader

        }("if(algorithm == "RS256

Use the provided key as an RSA public key //

        }("else if (algorithm == "HS256 {

Use the provided key as an HMAC secret key //

                                        {

                                    {
```

- الي بيحصل هنا إنه بيقبل الـ key على حسب الـ alg، فـ لو RS256 هيعتبر الـ key هو الـ public key والدنيا تمام، ولكن لو غيرنا الـ alg = HS256 ف هيعتبر الـ Key هو الـ secret وبالتالي لو الـ attacker حاول ولقي الـ public key هيقدر يعمل بيه sign للتوكين.
- الخطوات الي بيمر بيها الهجوم ده كالآتي:

- 1. تحاول تلاقي الـ Public key وممكن تشوف endpoints زي دول:
  - `/jwks.json`
  - `/.well-known/jwks.json`
- 2. تحول الـ Public key للـ format الصحيح عشان يتطابق مع الـ secret lcoal copy الي بيستخدمه السيرفر، وبيكون متخزن في ملف من ملفات النظام، أو في داتا بيس.
- 3. تعدل الـ JWT request بتاعك.
- 4. تعمل sign للتوكين من خلال الـ Public key وتبعت بعدين عادي.