Cache Poisoning

في بداية الموضوع خلينا نفترض إن عندنا موقع اسمه resturant.com الموقع ده كان بيجيله 15 زائر في اليوم والموقع شغال وفل الفل، بعد فترة الناس الي بدأت تزور الموقع تعدت الـ 1000 زائر ولكن مواصفات السيرفر مش مستحملة كم الضغط ده وبالتالي بنلجأ إننا نضيف كمان كذا سيرفر تاني عليهم نفس محتوى السيرفر الأصلي وبعدين بنجيب سيرفر أسمه (load balancer – proxy server) وظيفته إنه بيتسلم الريكوست من اليوزر ويبعتها لسيرفر من السيرفرات الأصلية الي عليها الموقع وبالتالي أحنا كده وزعنا الضغط من على سيرفر واحد لأكتر من سيرفر.

- الـ Load balancer بيبعت الريكوست على حسب الـ Configuration وفي أكتر من configuration ليه زي **round robin** والي بيبعت الريكوست بالترتيب على السيرفرات أو الـ **First responder** يعني أسرع سيرفر هيرد هبعتله الريكوست.

Cacheing = هو مصطلح يقصد بيه تقليل الضغط من على السيرفر الأصلي عن طريق الـ load balancer server كمثال بسيط، لو في يوزر بيطلب حاجة معينة متكررة ف هو مش كل ما اليوزر يطلب الحاجة دي يروح الـ load balancer للسيرفر الأصلي ويقوله هات كذا، لا احنا نخلي الحاجة دي على البروكسي سيرفر ولما يطلبها اليوزر يديهاله بدون ما يروح للسيرفر الأصلي وبكده هو بيقلل الضغط من على السيرفر الأصلي والحاجات دي ممكن تكون ملفات js أو css أو ملف معين في شات وهكذا.

- الـ keys هي الحاجة الي بيعتمد عليها الـ proxy server عشان يـ cache حاجة معينة، يعني لو في أكتر من ريكوست راح لنفس الحاجة سواء ملف أو صورة ف هو بيتعمله كاش، وهنا البروكسي سيرفر بيتعامل مع keys زي مثلا نفس البارامترز بتيجي كل مرة أو الـ user-agent أو الـ Host header وهكذا.,

.

تعريفها هي ثغرة بتحصل لما ببعت نفس الريكوست عدد مرات معين وبالتالي الصفحة الفلانية بيحصلها cached وبقي أي حد بيطلب نفس الريكوست بتعرضله الـ cached page بدون ما تروح للـ back-end server.

كمثال لو أنا بطلب الصفحة الرئيسية لفيس بوك وكان في misconfiguration في الـ Host header ف لما بتكتب xss payload بيشتغل ولكن self xss ف إني أخليها تظهر لكل اليوزر هنا يجي دور الـ cache poisoning في إني أبعت نفس الريكوست عدد مرات معين لحد ما الصفحة تبقى cached ولما أي يوزر تاني يطلبها هتتنفذ الـ XSS.

الهدف منها

الهدف هو إنك بتقدر تعدل في محتوى الموقع بحيث تقدر تستخدم الاتاك ده في إنك تضر باقي اليوزرز كمثال

- هو إنك تحول الـ Self-XSS أو Reflected-XSS لـ Stored-XSS.
- Disclosure of sensitive information
- Dos Attack
- Unauthenticated access to sensitive pages
- Password reset

المراحل الي بتمر بيها

- الثغرة بتعدي على مرحلتين: الأولى إنك تسبب ضرر في الريسبونس، أو بمعنى أصح يكون فيها malicious payload، والتانية إنك تعمل cache للريسبونس دي وبالتالي توصل لباقي اليوزر.
- ودي المراحل الي بتمر بيها عشان تكتشف الثغرة:

1- بتشوف ايه الـ Header الي بتكون unkeyed يعني مش بيـ check عليها وبتأثر على الـ response وممكن تستخدم أداة زي param Miner.

2- تشوف تأثيرك على الـ response لاي مدى زي تقدر تكتب js payload وهكذا.

3- وهو إنك تعمل cache للـ response دي.

- أول حاجة الاتاكر بيحاول يحصل على ريبونس من سيرفر الباك إند تكون مسممة وده عن طريق الـ Host header في الغالب
- بعد ما يتأكد إنه الريسبونس الي جاتله من السيرفر كانت مسممة ف يبدأ يعملها cache بحيث كل يوزر يزور الصفحة دي هيتعرضله الريسبونس المسممة.

مهم

- قولنا إن عملية الـ caching بتعتمد على الـ KEYs سواء كانت params أو Headers معينة زي الـ HOST أو الـ USER-agent وخلافه.
- لكن الـ proxy server أحيانا بياخد الـ IP بتاع اليوزر عشان يعرف هو هيرجع الـ Response لمين بالظبط ف هنا اليوزر بيبعت الـ IP بتاعه وبالتالي الـ proxy server برضو بيبعت الـ IP بتاعه للـ back-end server ف هنا في Header بتضاف احنا مش بنشوفها زي X-Forwarded-For أو client-ip أو X-client-ip: وهكذا ودي بتُسمى الـ unKeyed.

- الـ unKeyed header في كتير منهم والي هما بنستخدمهم في تخطي الـ Rate limit (إرجع لصفحة الـ Rate).
- في ثغرة اسمها Local Route poisoning ودي الي بنستخدم فيها الـ X-Original-URL أو X-Rewrite-URL لما بكون داخل ع صفحة ويديني Forbbiden ف بتلاعب عليه، ودي بتفيد في ثغرات الـ authentication والـ Privilege Escalation.

Cache Control

· الـ Cache-Control عبارة عن هيدر في الريكوست والريسبونس وبيحمل أوامر ومش ضروري يكون نفس الأمر في الريكوست هو الي في الريسبونس.

· بيكون شكله كالآتي Cache-Control: max-age=200sc وهي case sensitive للحروف.

· الأوامر الي ممكن نستخدمها في الريكوست هي (max-age, (max-stale,min-fresh,no-cache,no-transform,no-store,only-if-cahce)

· الأوامر الي ممكن نستخدمها في الريسبونس هي (no-cache,no-transform,no-store,public,private,must-revalidate,proxy-revalidate)

الصفحة بيحصلها cahced لو

· الـ status code = 301,302,308,307,410

· لو الـ Cache-Control: public or have max-age or s-max-age

· لو موجود Expires فيه

الأوامر المختلفة:

· Public: دي معناها إن الصفحة هيحلصها كاش بواسطة أي كاش حتى لو هي مش cacheable.

· Private: في الحالة دي الصفحة هيحصلها كاش بواسطة المتصفح بس حتى لو هي مش cacheable.

· No-Cache: نفس حوار الـ Public ولكن دي بتعدي على السيرفر الأول يعملها validate ويتأكد منها.

· No-Store: هنا الصفحة مش هيحصلها كاش خالص حتى لو في أوامر تاني مع الأمر ده.

· Must-Revalidate: مش بتشكل فرق مع الـ No-Store لإن الأمر ده عشان يتنفذ ويتعمل عليها Check سواء هي valid أو لا ف لازم يتعملها تخزين الأول وده بيتناقض مع الـ No-Store.

· Must-revalidate: معناه إن أول ما الريسورس يبقى قديم فالكاش مش هيبقله غير لما يروح يعمله revalidate من جديد في الـ main server.

- No-transform: معناها إن البروكسي ميقدرش يعدل في الـ Response body.

سيناريوهات

- السيناريو المشهور هو الـ Cache poisoning الي بتتحول لـ XSS عن طريق إن في Unkeyed header بيتطبع في الريسبونس وبالتالي لو قيمت كانت بايلود جافا سكريبت ف هيطبع برضو وبالتالي لما يحصل cache للصفحة دي أي يوزر هيدخلها بعدين هيظهرله alert.
- تاني سيناريو وهو إن الـ Unkeyed Header بيتم من خلاله create URL سواء يجيب ملف js أو يجيب صورة وهكذا، ف لما بيطبع في الريسبونس بيروح الاتاكر عامل ملف زي الي بيجيبه الموقع وحاطط فيه البايلورد بتاعه وبعدين يخلي قيمة الـ Unkeyed header بالـ URL بتاع موقعه وبالتالي هيجيب الملف الي موجود ع موقعه وبعدها تعمل للصفحة cache وتظهر لكل اليوزرز.
- أحيانا مش بيكون header هو الي بيسبب الـ cache poisoning لا ممكن تكون الكوكيز هي الي بتطبع في الريسبونس وبالتالي الاتاك هيجي منها هي (أحيانا وأنت بتجرب تكتب حاجة تشوفها بتطبع ولا لا مش هتلاقيها بتطبع ف جرب تكتب كود جافا سكريبت ع طول وتشوف هيطبع ولا لا)، ف جرب تغير قيمة الكوكي وشوف لو الي كتبته بيطبع في الريسبونس وبالتالي الـ cookie هي الي هتبقى unKey header.
- أحيانا التطبيق بيجبر الـ connection يكون HTTPS ف لو حاولت تبعت الريكوست ببرتوكول تاني زي HTTP,FTP الموقع بيعمل ريديركت لنفسه ويخلي الاتصال HTTPS وده بنختبره عن طريق X-Forwarded-Scheme: ftp وبعدها بيجي دور X-Forwarded-Host وتغير الهوست وتخليه URL الموقع بتاعك (لاحظ إن لازم يكون الـريكوست لملف جافا سكريبت عشان تقدر تحط البايلود بتاعك في ملف جافا سكريبت شبيه بيه.
- أحيانا لما السيرفر يجيله ريكوست بـ protocol غريب عن الي مستني يجيله زي http, ftp بيعمل redirection على الـ Host الي موجود بس بيخليه HTTPS، ف ممكن نكتب السيرفر بتاعنا في الـ Host وبالتالي السيرفر هيحول اليوزر للدومين بتاعنا.
- الثغرات الي بتحصل بتكون بسبب خطأ في تصميم الـ Cache server نفسه، أو بسبب الطريقة الي بيتم بيها إستخدام الـ cache server في موقع معين.
- دي الثغرات الي بتحصل نتيجة الـ design flaw in cache server:
  - السيناريو المشهور هو الـ Cache poisoning الي بتتحول لـ XSS عن طريق إن في Unkeyed header بيتطبع في الريسبونس وبالتالي لو قيمت كانت بايلود جافا سكريبت ف هيطبع برضو وبالتالي لما يحصل cache للصفحة دي أي يوزر هيدخلها بعدين هيظهرله alert.
  - تاني سيناريو وهو إن الـ Unkeyed Header بيتم من خلاله create URL سواء يجيب ملف js أو يجيب صورة وهكذا، ف لما بيطبع في الريسبونس بيروح الاتاكر عامل ملف زي الي بيجيبه الموقع وحاطط فيه البايلورد بتاعه وبعدين يخلي قيمة الـ Unkeyed header بالـ URL بتاع موقعه وبالتالي هيجيب الملف الي موجود ع موقعه وبعدها تعمل للصفحة cache وتظهر لكل اليوزرز.
  - أحيانا مش بيكون header هو الي بيسبب الـ cache poisoning لا ممكن تكون الكوكيز هي الي بتطبع في الريسبونس وبالتالي الاتاك هيجي منها هي (أحيانا وأنت بتجرب تكتب حاجة تشوفها بتطبع ولا لا مش هتلاقيها بتطبع ف جرب تكتب كود جافا سكريبت ع طول وتشوف هيطبع ولا لا)
  - أحيانا التطبيق بيجبر الـ connection يكون HTTPS ف لو حاولت تبعت الريكوست ببرتوكول تاني زي HTTP,FTP الموقع بيعمل ريديركت لنفسه ويخلي الاتصال HTTPS وده بنختبره عن طريق X-Forwarded-Scheme: ftp وبعدها بيجي دور X-Forwarded-Host وتغير الهوست وتخليه

URL الموقع بتاعك (لاحظ إن لازم يكون الـريكوست لملف جافا سكريبت عشان تقدر تحط البايلود بتاعك في ملف جافا سكريبت شبيه بيه.

- ممكن يكون التطبيق فيه ثغرة DOM-XSS بسبب إنه بيعتمد على داتا جاية من اليوزر, ف ممكن وقتها تحاول تعمل cache للصفحة دي وبكده هتنفذ الـ DOM-XSS على كل اليوزر.
- أحيانا بنضيف هيدر Access-control-allow-origin: * للصفحة الي احنا عملنالها create عشان نسمح لأي origin إنه يعمل request للملف بتاعنا.

- ملحوظة: الصفحة الي فيها set-cookie مينفعش يحصلها cache.
- دي الثغرات الي بتحصل نتيجة الـ implementation flaws:
  - الميثدولوجي هنا بتختلف عن الـ Classic cahce poisoning لإنها بتعتمد على فهمك للـ cache كويس، وطبعا بتختلف من موقع لموقع لإن كل موقع بيعمل Configuration على حسب هو عايزه ازاي والميثدولوجي بتكون كالآتي:

- - Identify a suitable cache oracle
- - Probe key handling
- - Identify an exploitable gadget

.

Cache Deception

- الثغرة دي بإختصار بتتلاعب بيها على البروكسي سيرفر, مثال لما بتيجي تـ access صفحة حساسة زي Info.php الي بيكون فيها معلوماتك ف هو أكيد مش هيعمل للصفحة دي cache خالص ف أنت بتروح تضيف في آخر الـ URL حاجة زي example.css أو مثلا example.js ف الـ URL بيكون كده

Site.com/info.php/example.css

فـ لما الريسبونس تعدي على الـ proxy server هو بيعتبر الـ example.css هي الصفحة والـ info.php ده directory ف بيروح يـ create ملف اسمه info.php ويعمل cache جواه للـ example.css ف لو بعت الـ URL كله ليوزر تاني ودخل عليه وهو authenticated هيحصل cache للصفحة بما فيها معلومات اليوزر وبالتالي تقدر تدخل ع الصفحة وتعرف المعلومات.

أماكن تواجدها

- في الصفحات الحساسة زي المعلومات الشخصية، تغيير الاميل, ورقم التليفون وهكذا.

تكتشفها ازاي

- بتروح لاي صفحة حساسة ولنفترض المعلومات الشخصية
- بتضيف في آخر الـ URL حاجة زي example.css أو أي حاجة مش موجودة
- بتبعت الريكوست بتاع 1000-10,000 مرة
- بتدخل من متصفح تاني خالص باميل تاني وتـدخل ع نفس الـ URL الي آخر example.css وتشوف هل المعلومات حصلها cache ولا لا.