|=| XXE |=|

XML

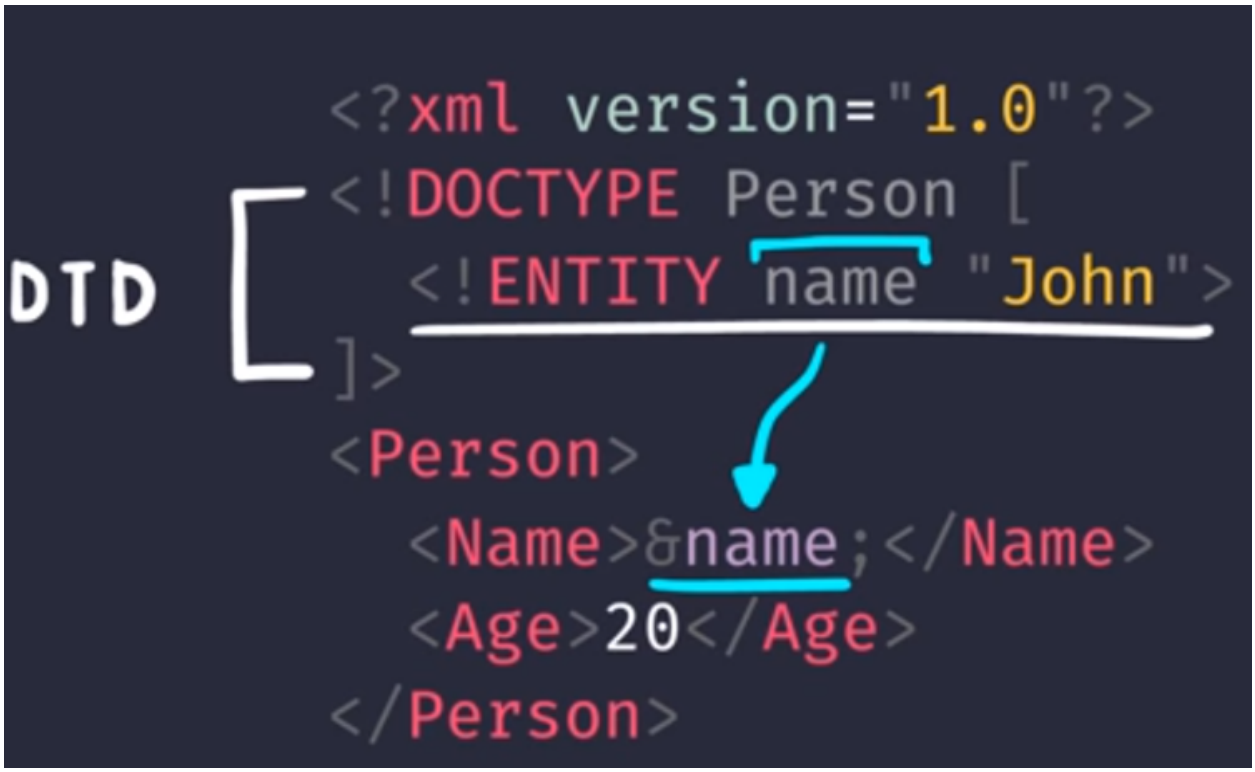
- طيب قبل ما نبدأ نعرف الثغرة وازاي نلاقيها وكل ده خلينا نعرف هي ايه الـ XML.
- الـ XML هي اختصار لـ Extensible Markup language, وهي لغة زيها زي الـ HTML ولكن الفرق الجوهري بين الاتنين إن الـ HTML بتُستخدم لعرض البيانات على صفحات الويب ولكن الـ XML بتُستخدم لنقل وأحيانا تخزين البيانات.
- الـ XML زي الـ JSON وبظهور الـ JSON بدأ الـ XML يختفي
- الـ HTML بتتكون من predefiend tags يعني تاجات معرفة مسبقا أو محجوزة وأنت بتستخدمها زي (dev, strong, head) لكن الـ XML أنت بتسمي التاج براحتك ويفضل يكون أسمه بيوصف الداتا الي جوا التاج.

- هنا مثال لكود XML :

- في كود XML بيكون في Root tag واحد بس وجواه بيكون باقي الـ Tags, كمان الـ start/end tag بيكونوا case sensitive يعني لازم نفس الاسم.

|=| Entity |=|

- الـ entity ممكن نفكر فيها كـ variable بتخزن جواه قيمة وبتقدر تستدعيها في أماكن كتير ومختلفة داخل الـ XML Document وبيتم تعريف الـ Entity في مكان محدد وهو الـ DTD.
  - DTD = Document Type Definition
- تاج الـ DOCTYPE هو الي بيعرف الـ XML parser إن الجزء ده هو الـ DTD وجواه بتكون الـ Entities.
- طريقة كتابة الـ Entity:



- قولنا إن الـ Entity زي الـ variable ولكن بتختلف في إنها مش بس بتخزن قيمة انت بتحطها بس, ولكن ممكن تخزن قيمة من Local file, أو من موقع خارجي وده بيُسمى External Entity وده بيسبب Attacks كتير.

- المشكلة في الـ Entity والي بتسبب ثغرة XXE هو الـ Attribute الي اسمه SYSTEM والي بدوره بيقول للـ Entity إن الداتا بتاعته هيجيبها من local file أو remote source بشكل عام، وده مثال للكود الضعيف الي ممكن تستغله.

XXE Injection

|=| تعريفها |=| هي ثغرة بتحصل لما يكون في Entity بياخد القيمة بتاعته أو الداتا من ملف على السيرفر أو من موقع خارجي وبالتالي هنا ممكن اغير اسم الملف وأعرض محتوى ملف تاني أو أعرض موقع خارجي غير الي موجود في الكود الأصلي وبكده هخلي السيرفر الي عليه ملف الـ XML يبعت ريكوست لسيرفر خارجي وبالتالي هيحدث SSRF Attack.

|=| أسباب حدوثها |=|

- هو Attribute الـ SYSTEM والي بدوره بيجيب الداتا من ملف على السيرفر أو موقع خارجي وهنا ممكن اغير براحتي، وهنا مثال للكود الضعيف:

```
<?"xml version="1.0" encoding="UTF-8?>

<stocklevel><ProductID>215</productID></stocklevel>
```

- ف هنا مفيش أي حماية ضد الثغرة وبالتالي لو الـ Application بيبعت ريكوست زي ده في ممكن نضيف عليه ونستغله كالآتي:

```
<?"xml version="1.0" encoding="UTF-8?>

<[ <"DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd!>

<stocklevel><ProductID>&xxe;</ProductID></stocklevel>
```

- هنا قدرنا نستخدم الـ SYSTEM Entity عشان نرجع محتوى ملف الـ /etc/passwd/ وتقدر تستغلها بأكتر من شكل زي ما مكتوب ف القسم الي تحت.

|=| أنواعها |=|

- In-band: وهنا أنت بتعمل ريكوست بالـ Payload بتاعك وبتظهرلك الـ Response قدامك
- Error: وهنا ممكن تعتبرها Blind XXE بس مش blind كاملة لانه بيظهرلك بعض الايرورز الي بتفهم منها الـ Response.
- OOB: ودي Fully Blind لإنه مش بيظهر أي شيء لا ايرورز ولا حاجة.

وممكن نصنف أنواعها بناءاً على الـ Results أو الإستغلال زي:

- إستغلال الـ XXE عشان ترجع محتوى ملف على نفس السيرفر (هنا الـ Entity قيمته بتكون ملف ع السيرفر)
- إستغلال الـ XXE عشان ننفذ SSRF attack (هنا الـ Entity قيمته بتكون URL لموقع خارجي)
- إستغلال الـ XXE عشان نرجع داتا مهمة على شئ لينا كنترول عليه. (Out-Of-Band)
- إستغلال الـ Blind XXE عشان نرجع داتا بس عن طريق الـ Errors.
- إستغلال الـ XXE في تنفيذ RCE لو الـ expect:// كان متاح ومش blocked.

- إستغلال الـ XXE في تنفيذ Dos attack أو بيسموه Billion Laughs Attack وده مثال للـ exploit code:

Request	POST http://example.com/xml HTTP/1.1 <?xml version="1.0" encoding="ISO-8859-1"?> <!DOCTYPE mytype [<!ELEMENT mytype ANY> <!ENTITY name "John "> <!ENTITY name2 "&name;&name;"> <!ENTITY name3 "&name2;&name2;&name2;&name2;"> <!ENTITY name4 "&name3;&name3;&name3;&name3;">]> <foo> Hello &t3; </foo>
Response	Response HTTP/1.0 200 OK Hello John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John John

- وده بيخلي الـ XML parser يستخدم مساحة كبيرة من الـ Memory وبالتالي بيحصل overload ليها.

ملحوظة:

- الـ Entity في منه 3 أنواع وهما:

- الـ General Entity والي هو متعارف عليه, يعني بتعرف Entity في الـ DTD وبتستدعيه في كود الـ XML.
- الـ Parameter Entity وده بيبدأ بـ % ووظيفته إن قيمته ممكن تكون Entity تاني يعني Entity جوا Entity والـ parameter Entity مش بتقدر تستدعيه إلا جوا الـ DTD وبالتالي هو بيستدعي الـ Genral Entity الي جواه وهنا بتقدر تستدعي الـ Gernal Entity في كود الـ XML.
- الـ Predefined Entity ودول الـ < > # @ وغيرهم لإنهم أحيانا بيجيبوا ايرور في الكود ف بنستخدم الـ Entity بتاعهم.

|=| إزاي تعمل تيست للـ XXE |=|

- أول حاجة عشان تعمل تيست للـ XXE هو إنه تجرب تـ insert الـ XXE metacharacters زي الـ
  - الـ Single quote (')
  - الـ Double quotes (")
  - الـ Angualr parentheses (<>)
  - الـ Comment (<!-- ->)
  - الـ Ampersand الـ (&)
  - الـ CDATA section delimiters == ودي بتُستخدم عشان تحط جواها Tags بس الـ XML parser يعتبرها داتا عادية كأنها text، ودي ممكن توصلك لـ XXS attack.
    - CDATA to XSS
- بعد ما خلصت الخطوة الأولى وعرفت الـ Structure بتاع الـ XML document هيجي دور إنك تـ inject code
- لو لقيت بارامترز وعايز تعرف إذا كانت مصلا بتستخدم XML أو في ثغرة XXE، جرب عن طريق البرب سويت تغير شكل الريكوست لـ json وتبعت لو قبلها غالبا بيتسخدم XML، او غير الريكوست لـ XML، لو رفض الأتنين ف جرب مثلا تكتب في قيمة البارامتر &entity; وتشوف لو ظهرلك إيرور.
- جرب تستخدم الـ XIclude لو لقيت إنك ملكش access على الـ XML document كله ولكن ليك access على جزء منه.
-

|=| أماكن تواجد الـ XXE |=|

- XML APIs
- SOAP APIs
- في أي مكان يتم فيه تحليل ملف(Microsoft office docx / xlxs / pptx / إلخ). دي مجرد ملفات مضغوطة مليئة بملفات XML.
- RSS feed parser (الـ RSS feed مجرد XML file)
- SAML authentication
- فالـ HTML parsing (زي تحويل الـ HTML لـ PDF)
- لو في Feature/Functionality بتتعامل مع الـ sitemap.xml

- لو في Feature/Functionality بتتعامل مع الـ SVG files.
- ممكن في الـ Registeration بإنه يضيف <user> في ملف xmlDB كـ node.
  - لو قابلك سيناريو الـ Registeration وبيتسخدم XML أمشي مع الخطوات دي (Link)
- من الـ Real-world scenarios الي ممكن تقابلها:
  - الـ Andriod development tools: كتير منهم بيتعامل مع الـ XML parser بطريقة تخلي الـ attacker يستخدم الـ External entities بشكل شئ، ومن التوولز دي الـ Andriod studio, APKtool ولكن في الإصدارات الجديدة إتصلحت المشاكل دي.
  - الـ WordPress: بتحصل ثغرات الـ XXE في الورد بريس وده مش كويس لإن 40% من تطبيقات الويب بتستخدم الورد بريس وبالتالي ممكن تكون معرضة لهجمات الـ XXE.

|=| إستغلال الثغرة |=|

- أول حاجة بيكون عن طريق إسترجاع الملفات الحساسة زي ملف الباسورد وكأنها LFI.
- تاني طريقة وهي إرسال ريكوست لموقع خارجي ويعرضلي الـ Response أو اعمل access لـ other back-end systems وكأنها SSRF.
- تالت حاجة وهو الـ Blind والي مش بيعرض فيها Response زي ما عرفنا, في هنا بنلجأ لطريقة الـ OOB = Out Of Band وهو إنك تستخدم سيرفر خارجي تقدر تتحكم فيه وتقدر تشوف الريكوستات الي جاتله, ف بتعمل Entity بيروح يـ access السيرفر الخارجي بتاعك وتشوف هل في ريكوست جاله ولا لا، لو في يبقى في Blind XXE.
- لو الموقع بيسمح تـ upload ملف svg اكتب ملف جواه الكود الي عايزه يتنفذ وارفعه وهيتم تنفيذه.
- ممكن تنفذ Billion laugh attack أو Remote code execution.
- أحيانا مش بيظهرلك كود XML وبيكون عبارة عن parameters ف هتجرب &entity; ولو حصل إيرور وعرفت إنه بيستخدم XML ف هنا أنت ليك access على جزء بس من كود الـ XML مش كله ف جرب XInclude.
- ممكن ميظهرش أي شئ خاص بالـ XML وبيظهر رسالة خطأ عادية، بس أنت قدامك كود XML وعارف إنه بيستخدمه, هنا بتحاول توصل لـ DTD موجود بالفعل وتكون عارف entity معين جواه وتبدأ تعدل عليه.

|=|Code review for XXE |=|

- لو في واحد من الـ JAVA APIs دول مش configured كويس ف ممكن يبقى مصاب بالـ XXE

```
javax.xml.parsers.DocumentBuilder
javax.xml.parsers.DocumentBuildFactory
org.xml.sax.EntityResolver
org.dom4j.*
javax.xml.parsers.SAXParser
javax.xml.parsers.SAXParserFactory
TransformerFactory
SAXReader
DocumentHelper
SAXBuilder
SAXParserFactory
XMLReaderFactory
XMLInputFactory
SchemaFactory
DocumentBuilderFactoryImpl
SAXTransformerFactory
DocumentBuilderFactoryImpl
XMLReader
Xerces: DOMParser, DOMParserImpl, SAXParser, XMLParser
```

- شوف الـ Source code ودور في الـ docType او الـ DTD لو في external entities مصابة
- الـ JAVA OPI reader لو كان أقل من 3.10.1 ف غالبا مصاب بالـ XXE.
  - ممكن تعرف الـ version بتاع الـ JAVA OPI reader من الـ JAR filename
    - poi-3.8.jar
    - Poi-ooxml-3.8.jar
- .

|=| إزاي تمنعها |=|

- عشان تمنع الـ Billion Laughs Attack محتاج تحط Limit للـ XML parser من الميموري يستخدمها.

- يُفضل تستخدم libxml_disable_entity_loader لو بتستخدم PHP ووظيفة الـ Function دي إنها بتمنع إستخدام الـ XML external entities والي بتسبب ثغرة XXE.
- أعمل Manual configuration للـ XML parser وخليه يمنع الـ DTD الي بيتم تعريف الـ entities فيه.
- إستخدم الـ WAFs ودي بتمنع كتير من الـ XXE inputs.
- الاتنين فيتشرز دول لازم يكونوا قيمتهم False:
  - external-general-entities
  - external-parameter-entities
  - ويُفضل إنك تستخدم disallow-doctype-decl وتكون قيمتها "true" عشان تتأكد إن الـ attacker مش هيقدر يضيف DOCTYPE.
- بما إنها واحدة من الـ input validation vulnerabilities based-input ف لازم تعمل كويس.
- وممكن تقرأ أكتر عن الـ mitigation من خلال اللينك ده لإنه طويل جداً Mitigation XXE

|=| Tools |=|

- GitHub - luisfontes19/xxexploiter: Tool to help exploit XXE vulnerabilities
  - (XML Injection Fuzz Strings (from wfuzz tool
-

|=| مصادر |=|

- https://brightsec.com/blog/xxe-vulnerability/
- https://brightsec.com/blog/xxe-attack/
- https://www.bugcrowd.com/blog/how-to-find-xxe-bugs/
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/07-Input_Validation_Testing/07-Testing_for_XML_Injection

- https://application.security/free-application-security-training/owasp-top-10-xml-entity-injection
- https://book.hacktricks.xyz/pentesting-web/xxe-xee-xml-external-entity
-