

Assignment 8.1

Course name: XML attacks

Developers:

- Anders Carlsson
- Oleksii Baranovskyi

Performer:

- FB-01 Sakhnii Nazar

Table of Contents

Task 1. XXE injection	1
Task 2. WSDL testing	3

Task 1. XXE injection

Purpose: understand how to detect and exploit XXE-injection

After the work the student must

- know: how detect XXE-injection vulnerabilities;

Tasks:

- analyze provided VM (192.168.56.11);
- detect XML communication;
- exploit XXE-injection.

Technical equipping of the workplace:

- OWASP Burp Suite
- OWASP Zend Attack Proxy
- Chrome developer tools
- Mozilla developer tools

Solution:

Launch VM. Open <http://192.168.56.11> in browser. Analyze HTTP-parameters for GET/POST. Find XML-based communication.

TASK 1

What page is vulnerable to XXE-injection? Prove it (screenshot).

Answer:

Для знаходження потенційно вразливих веб-сторінок до **XML External Entity** ін'єкції необхідно скористатися сканером веб-вразливостей **Burp Suite**, який дасть можливість дізнатися, де на веб-сайті відбувається передача даних у форматі **XML**.

The screenshot shows the Burp Suite interface. On the left, the 'Site map' tab is active, displaying a tree structure of the website. The 'Contents' tab is also visible, showing a list of HTTP requests and responses. A specific request is highlighted, showing the raw XML data. A blue box with text is overlaid on the screenshot, indicating that the XML parser 'soap' was found after sending a request to the /echo_xml endpoint.

Хоча даний XML-парсер "soap" вдалось знайти лише після надсилання деякого довільного повідомлення на веб-сторінці /echo_xml

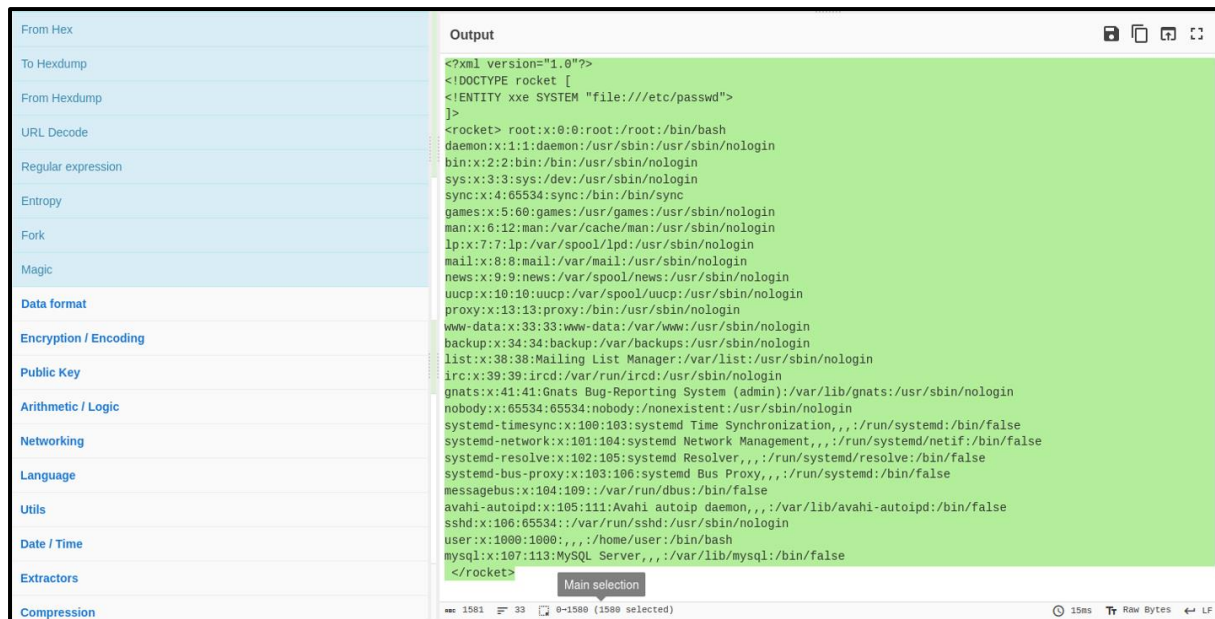
TASK 2

How can you exploit XXE-injection? Prove it (screenshot).

Answer:

The screenshot shows the Burp Suite interface with the 'Request' and 'Response' tabs active. The 'Request' tab displays the raw XML data, which includes a payload designed to exploit XXE-injection. The 'Response' tab shows the server's response, which includes the XML data. A green box highlights the payload in the request, and a red box highlights the response, indicating the successful exploitation of the XXE-injection.

The screenshot shows the Burp Suite interface with the 'Operations' and 'Input' tabs active. The 'Operations' tab displays the final exploit payload, which is a long string of XML data. The 'Input' tab shows the raw XML data, which includes the payload. A green box highlights the payload in the input, and a red box highlights the response, indicating the successful exploitation of the XXE-injection.



Task 2. WSDL testing

Purpose: understand how to detect and work with WSDL schemas

After the work the student must

- know: how detect and use XML schemes (WSDL);

Tasks:

- analyze provided VM (192.168.56.11);
- detect WSDL schema definition;
- analyze announced methods;
- use methods.

Technical equipping of the workplace:

- OWASP Burp Suite
 - OWASP Zend Attack Proxy
 - Chrome developer tools
 - Mozilla developer tools
 - Boomerang - SOAP & REST Client
- <https://chrome.google.com/webstore/detail/boomerang-soap-rest-clien/eipdnjedkpcnlmmdfdkgfpljanehloah>)

Solution:

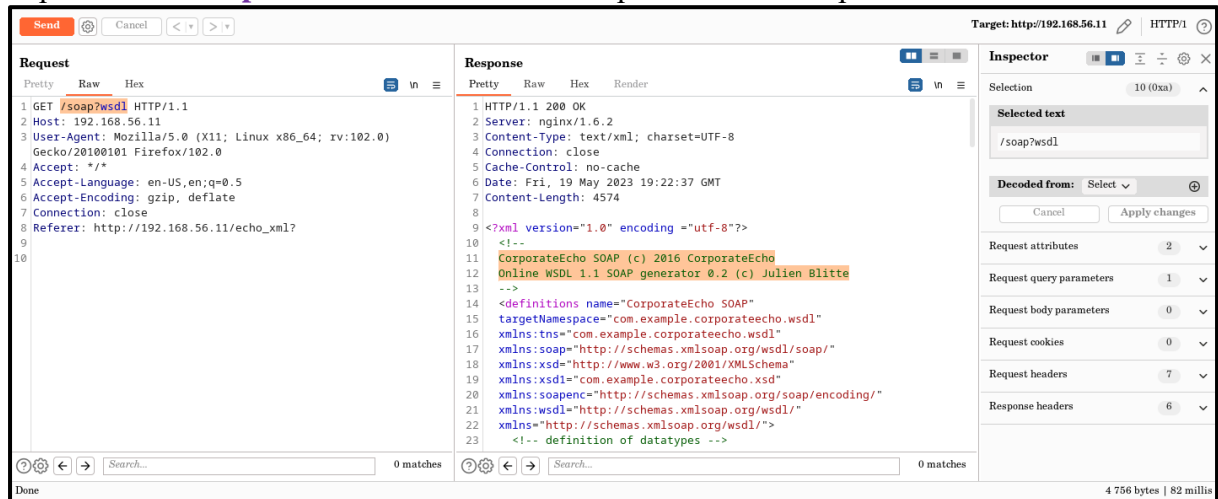
Launch VM. Open <http://192.168.56.11> in browser. Analyze HTTP-parameters for GET/POST. Find WSDL. Analyze results. Detect worked methods.

TASK 1

What page announces WSDL? Prove it (screenshot).

Answer:

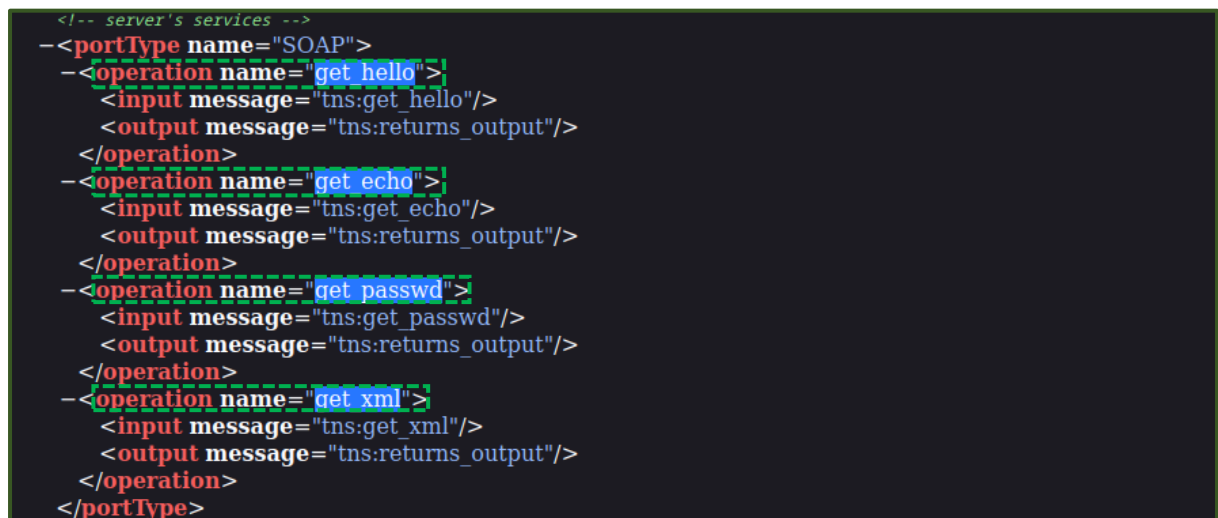
Так само, тільки якщо надіслати деякий запит до веб-сторінки `/echo_xml`, то в історії перехоплень **Burp Suite** можна помітити звернення до мови розміток “WSDL”.



TASK 2

Which methods are announced via WSDL? Prove it (screenshot).

Answer:



TASK 3

What method is possible to execute? Prove it (screenshot).

Answer:

The screenshot shows a SOAP client interface with a target URL of `http://192.168.56.11`. The request is a SOAP message with the action `com.example.corporateecho.wsd1/get_xml`. The response is a SOAP message with the action `com.example.corporateecho.wsd1/get_hello`. The response body contains the text `Hello, Nazar`. The interface includes a 'Send' button, a 'Cancel' button, and a 'Target' field. The 'Request' and 'Response' tabs are visible, showing the raw XML data. The 'Inspector' panel on the right shows the selected text 'Hello, Nazar'.

The screenshot shows a SOAP client interface with a target URL of `http://192.168.56.11`. The request is a SOAP message with the action `com.example.corporateecho.wsd1/get_echo`. The response is a SOAP message with the action `com.example.corporateecho.wsd1/get_echo`. The response body contains the text `Glory To Ukraine!`. The interface includes a 'Send' button, a 'Cancel' button, and a 'Target' field. The 'Request' and 'Response' tabs are visible, showing the raw XML data. The 'Inspector' panel on the right shows the selected text 'Glory To Ukraine!'.

The screenshot shows a SOAP client interface with a target URL of `http://192.168.56.11`. The request is a SOAP message with the action `com.example.corporateecho.wsd1/get_passwd`. The response is a SOAP message with the action `com.example.corporateecho.wsd1/get_passwd`. The response body contains the text `Query error!`. The interface includes a 'Send' button, a 'Cancel' button, and a 'Target' field. The 'Request' and 'Response' tabs are visible, showing the raw XML data. The 'Inspector' panel on the right shows the selected text 'Query error!'.

The screenshot shows a SOAP client interface with a target URL of `http://192.168.56.11`. The request is a SOAP message with the action `com.example.corporateecho.wsd1/get_xml`. The response is a SOAP message with the action `com.example.corporateecho.wsd1/get_xml`. The response body contains the text `PFCE70NUVBFHJvY2tldCBBIDwhRUSUSVRZIHh4Z5BTWVNUROgInZpbGUGLy8vZXRjL2hvc3RuYW11j4gXT4KPHJvY2tldDQgJnh4Z2TsgPC9yZnZnZXQ+IA==`. The interface includes a 'Send' button, a 'Cancel' button, and a 'Target' field. The 'Request' and 'Response' tabs are visible, showing the raw XML data. The 'Inspector' panel on the right shows the selected text 'Decoded from: Base64'.