# PortSwigger SSRF Lab Notes

Server-side request forgery (" SSRF "), bir bilgisayar güvenliği [exploit](https://tr.wikipedia.org/wiki/Exploit) türü olup, saldırganın bir sunucunun fonksiyonlarını kötüye kullanarak o sunucunun alanında bulunan ama aksi takdirde saldırganın doğrudan erişemeyeceği bilgilere erişmesine veya bu bilgileri manipüle etmesine neden olur

1. Basic SSRF against the local server

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at http://localhost/admin and delete the user carlos.

 Solution

1. Browse to /admin and observe that you can't directly access the admin page.
2. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
3. Change the URL in the stockApi parameter to http://localhost/admin. This should display the administration interface.
4. Read the HTML to identify the URL to delete the target user, which is:

***http://localhost/admin/delete?username=carlos***

1. Submit this URL in the stockApi parameter, to deliver the SSRF attack.
2. Basic SSRF against another back-end system

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, use the stock check functionality to scan the internal 192.168.0.X range for an admin interface on port 8080, then use it to delete the user carlos.

 Solution

1. Visit a product, click **Check stock**, intercept the request in Burp Suite, and send it to Burp Intruder.
2. Change the stockApi parameter to ***http://192.168.0.1:8080/admin***then highlight the final octet of the IP address (the number 1) and click **Add §**.
3. In the **Payloads** side panel, change the payload type to **Numbers**, and enter 1, 255, and 1 in the **From** and **To** and **Step** boxes respectively.
4. Click **Start attack**.
5. Click on the **Status** column to sort it by status code ascending. You should see a single entry with a status of 200, showing an admin interface.
6. Click on this request, send it to Burp Repeater, and change the path in the ***stockApi to: /admin/delete?username=carlos***
7. Blind SSRF with out-of-band detection

This site uses analytics software which fetches the URL specified in the Referer header when a product page is loaded.

To solve the lab, use this functionality to cause an HTTP request to the public Burp Collaborator server.

**Note**

To prevent the Academy platform being used to attack third parties, our firewall blocks interactions between the labs and arbitrary external systems. To solve the lab, you must use Burp Collaborator's default public server.

A screenshot of a computer

AI-generated content may be incorrect.

 Solution

1. Visit a product, intercept the request in Burp Suite, and send it to Burp Repeater.
2. Go to the Repeater tab. Select the Referer header, right-click and select "Insert Collaborator Payload" to replace the original domain with a Burp Collaborator generated domain. Send the request.
3. Go to the Collaborator tab, and click "Poll now". If you don't see any interactions listed, wait a few seconds and try again, since the server-side command is executed asynchronously.

A screenshot of a computer

AI-generated content may be incorrect.

1. You should see some DNS and HTTP interactions that were initiated by the application as the result of your payload.

**🔎 What to Look For**

* **Parameters/headers that trigger backend requests** → here it’s the **Referrer header**, consumed by analytics software.
* **No direct response (blind SSRF)** → exploit must rely on out-of-band interaction (e.g., Burp Collaborator).
* **Indicators of failure** → if no DNS/HTTP interaction shows up, check if you need a full URL (e.g., http://).

**✅ Key Points**

* Vulnerable functionality: **Referrer header** on product pages.
* Attack method: Replace Referrer with a **Burp Collaborator domain** to test for SSRF.
* Since it’s blind, success isn’t visible in the HTTP response → you must use **Burp Collaborator polling**.
* Mistake to avoid: payload must include full scheme (http://your-collaborator-id.burpcollaborator.net).
* Successful exploitation: Collaborator logs show a DNS/HTTP request from the target server → confirms **blind SSRF**.
* Impact: Blind SSRF alone is “proof of concept” → in real scenarios, it can pivot to internal services or chained exploits (like Shellshock in the next lab).

1. SSRF with blacklist-based input filter

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at http://localhost/admin and delete the user carlos.

The developer has deployed two weak anti-SSRF defenses that you will need to bypass.

 Solution

1. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
2. Change the URL in the stockApi parameter to ***http://127.0.0.1/*** and observe that the request is blocked.
3. Bypass the block by changing the URL to: ***http://127.1/***
4. Change the URL to ***http://127.1/admin*** and observe that the URL is blocked again.
5. Obfuscate the ***"a" by double-URL encoding it to %2561 to*** access the admin interface and delete the target user.

**🔑 Why the obfuscations work**

**1. http://127.1/ instead of http://127.0.0.1/**

* But in IP addressing, **127.1 = 127.0.0.1** (the OS expands it to the loopback address).
* Since the filter only checks for the **blacklisted string** 127.0.0.1, it doesn’t catch 127.1.
* Result: request still goes to localhost, bypassing the block.

**2. Double URL-encoding (%2561)**

* The filter also blocks paths containing /admin.
* If you send:
* /%2561dmin
  + %25 = %
  + %2561 → decoded once by the filter into %61.
  + %61 = a.
* After the filter runs, the backend decodes again and sees /admin.
* The filter only looked at the **first decoded version**, so it misses the trick.

1. SSRF with filter bypass via open redirection vulnerability

This lab has a stock check feature which fetches data from an internal system.

To solve the lab, change the stock check URL to access the admin interface at http://192.168.0.12:8080/admin and delete the user carlos.

The stock checker has been restricted to only access the local application, so you will need to find an open redirect affecting the application first.

 Solution

1. Visit a product, click "Check stock", intercept the request in Burp Suite, and send it to Burp Repeater.
2. Try tampering with the stockApi parameter and observe that it isn't possible to make the server issue the request directly to a different host.
3. Click "next product" and observe that the path parameter is placed into the Location header of a redirection response, resulting in an open redirection.
4. Create a URL that exploits the open redirection vulnerability, and redirects to the admin interface, and feed this into the stockApi parameter on the stock checker:

***/product/nextProduct?path=http://192.168.0.12:8080/admin***

1. Observe that the stock checker follows the redirection and shows you the admin page.
2. Amend the path to delete the target user:

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