Server-Side Request Forgery (SSRF)

What is a Server-Side Request Forgery Attack?

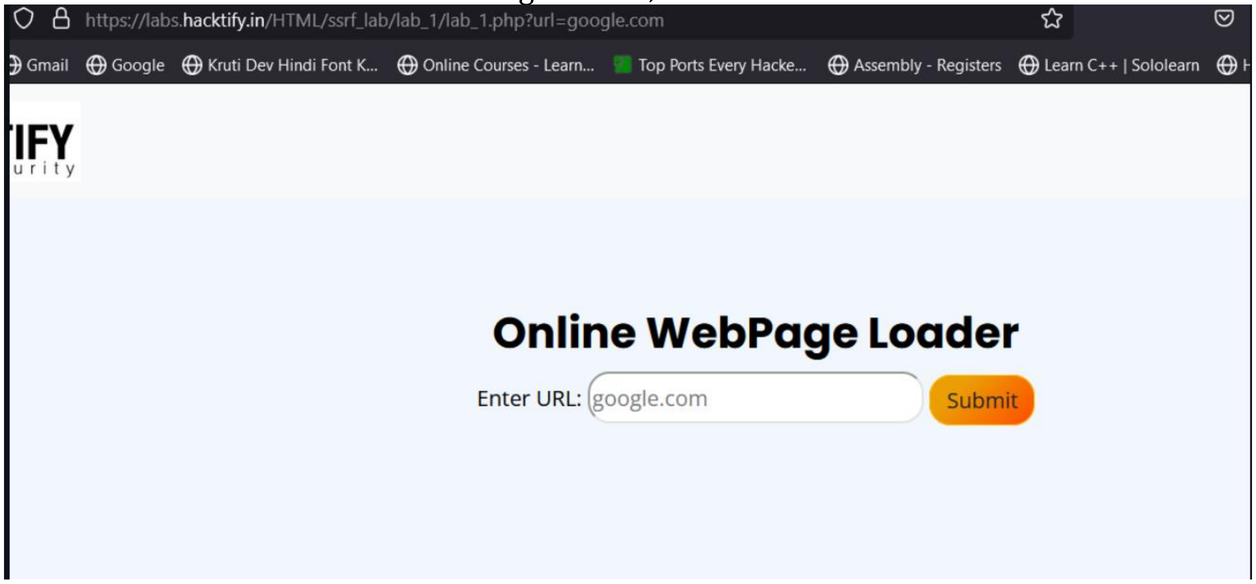
Server-side request forgery (SSRF) is a web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing. In a typical SSRF attack, the attacker might cause the server to connect to internal-only services within the organization's infrastructure. In other cases, they may be able to force the server to connect to arbitrary external systems, potentially leaking sensitive data such as authorization credentials. In simple words, Server-Side Request Forgery (SSRF) refers to an attack, wherein an attacker can send a crafted request from a vulnerable web application. SSRF is mainly used to target internal systems behind WAF (web application firewall), that are unreachable to an attacker from the external network. Additionally, it's also possible for an attacker to mark SSRF to access services from the same server that is listening on the loopback interface address (127.0.0.1).

Severity:

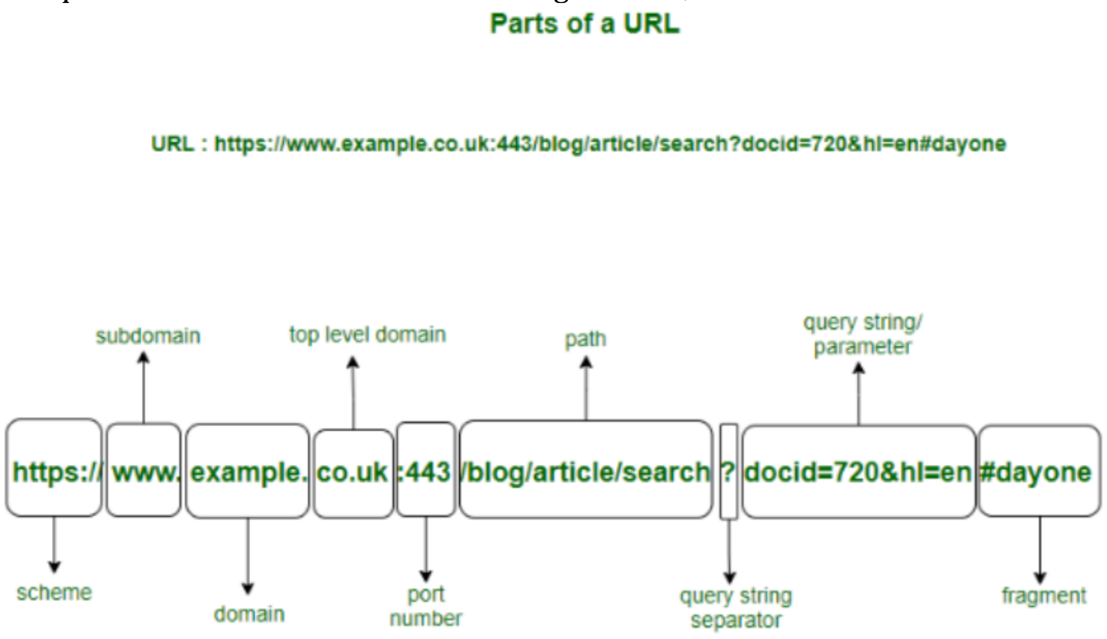
The severity of SSRF varies and depends on a case-to-case basis.

Lab 1: Get The 127.0.0.1

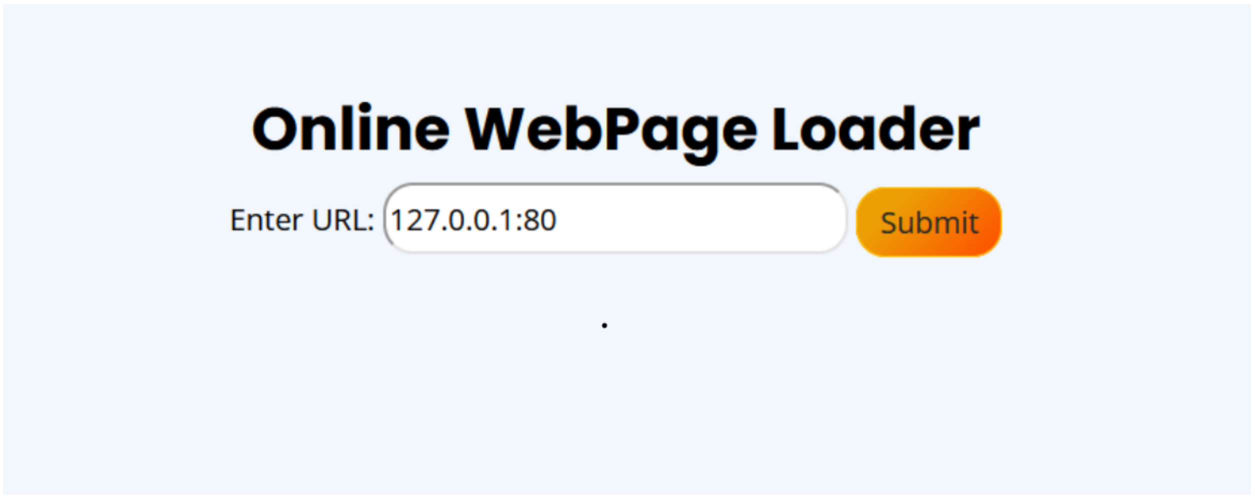
Observations: On accessing the lab we can observe that we are given a search bar, in which we can enter a URL which is something like this,



Solution: Before starting to play with the labs we should first know that SSRFs can be tricky due to the blacklisting and whitelisting implemented by developers, and we'll try to understand the same with this lab too.... Let's start by entering what is suggested in the search bar and we get the following response which is no response, Now as the name of the lab suggests, "Get the 127.0.0.1", 127.0.0.1 is the local or loopback address that points back to the host itself, so we can understand that it needs us to access the server itself, but now the question arises, ‘how can we request the server without knowing what kind of service is enabled?’, So the simple answer to it is first we would have to find the service running (or open port) on our victim server. (Pretty easy right??) :)   Now we’ll discuss two ways to scan the ports, as we are aware that every service running in a system is assigned one of the 65536 available ports, in my previous job I was asked, “How will you scan for open ports in a scenario where you don’t want to run a nmap scan or other scripts?” Take your time and answer this to me on my discord, Huh!!! I’ll give you the answer, when we make a request our actual URL looks something like this,



So here to scan open ports on a site we could just change the port number or add a port number of our choice at the end of the domain name such as (www.example.com:3030), Now the two ways of scanning ports are, Checking for commonly known ports by adding them to the URL. (www.example.com:xx) where “xx => Port numbers” And by automating the process using burp Intruder or some script.   Back to our lab, let’s just search for the following, Payloads: localhost or http://localhost/ And we’ll notice that we are not getting any response, Ooooh! wait are we forgetting something let’s use the following payloads, Payload: localhost:80 or http://localhost:80/ Still, no response, if we remember we had talked about blacklisting and whitelisting concept in our previous write-ups so remembering that and taking hints from the Lab name, Payload: 127.0.0.1:80



Upon sending this we get the following response, As this is the Apache index.html page we got the 127.0.0.1 Hence, we solved this lab!