Server-Side Request Forgery is a serious web vulnerability that results from bad programming. It allows attackers to send requests from a server to other external and internal resources, and receive responses.

Hackers mostly use SSRF attack for targeting internal systems behind the firewalls that are otherwise inaccessible from external networks. They also leverage SSRF attacks for accessing services available through loopback interface of exploited server.

An SSRF vulnerability exists when the criminal gains partial or full control of the web application’s request. A common example is when the attacker controls the third-party service URL requested by the web application.

## How to Mitigate Server-Side Request Forgery

Simply applying regular expressions and blacklists to user input is not a good approach for mitigating SSRF. Generally, blacklisting is a poor way of controlling security. Attackers still find ways to bypass them. They can use HTTP redirecting, alternate IP encoding, or use wildcard DNS service.

## DNS Resolution and Whitelisting

The best approach towards avoiding SSRF is to whitelist the IP address or DNS name that your application has to access. If whitelisting is not suitable and one has to rely on blacklisting, it’s necessary to properly validate user input then.

In case one has to adopt blacklisting approach, the way of mitigating correctly varies according to the application in question. In simpler words, it means that there is no one universal method to fix Server-Side Request Forgery. The type of mitigation method one adopts depends upon the business requirements and application functionality.

## Handling Response

In order to prevent the response data from going into the hands of the attacker, it is important to ensure that response is same as expected. It is also necessary to make sure that the raw response body from the server’s request is not delivered to the client.

## Disabling URL Schemas that are Unused

If an application is only using HTTP and HTTPS for making requests, only these URL schemas should be allowed. By disabling other unused schemas, attackers cannot use the web application for making requests using potentially dangerous schemas like dict://, file://, and ftp://.

## Authenticating Internal Services

Some of the services such as MongoDB, Memcached, and Elasticsearch don’t require authentication by default. Hence, an attacker can use SSRF vulnerability and access any such services without needing authentication. Thus, for ensuring security of the web application, it is important to enable authentication where possible.

To find out more about how to prevent SSRF vulnerabilities in your web application book a free consultation with Aardwolf Security and talk to our experts.