# Prevention of PDF Generation Vulnerabilities

After discussing different ways to exploit HTML injection vulnerabilities in PDF generation libraries, let us discuss ways to prevent these types of vulnerabilities.

## Insecure Configurations

Many of the vulnerabilities we discussed in the previous sections result from the improper configuration of PDF generation libraries. There are many cases where the default settings of these libraries are insecure. While many of them have been discovered and fixed, we should not rely on the security of the default settings. Thus, reading the documentation, stepping through the configuration file, and configuring the PDF generation library according to our needs are all essential. For instance, many PDF generation libraries default the configuration to allow access to external resources. Setting this option to false effectively prevents SSRF vulnerabilities. In the DomPDf library, this option is called enable\_remote.

In some libraries, there are other configuration options that enable the execution of JavaScript and even PHP code on the server. While using features like these might be helpful for the dynamic generation of PDF files, they are also extremely dangerous, as the injection of PHP code can lead to remote code execution (RCE). For example, the DomPDF library has a configuration option called isPhpEnabled that enables PHP code execution; this option should be disabled because it's a security risk.

Generally, most libraries provide security best practices that we should follow when using them. For instance, [here](https://github.com/dompdf/dompdf/wiki/Securing-dompdf) are security best practices for DomPDF.

## Prevention

All vulnerabilities discussed previously result from user-supplied HTML tags being used as input to the PDF generation library. A web application can prevent these vulnerabilities by disallowing HTML tags in the user input. This can be achieved by HTML-entity encoding the user input, for example, by using the [htmlentities](https://www.php.net/manual/en/function.htmlentities.php) function in PHP. htmlentities will convert all applicable characters to [HTML entities](https://www.freeformatter.com/html-entities.html), as in < becoming &lt; and > becoming &gt;, making it impossible to inject any HTML tags, therefore preventing security issues.

However, in many cases, this mitigation might be overly restrictive as it may be desired for the user to be able to inject certain style elements, such as bold or italic text, or resources, such as images. In that case, the user must be able to insert HTML tags into the PDF generation input. We can mitigate the vulnerabilities we discussed by configuring the PDF generation library options properly by taking into consideration security all security problems. At the very least, we need to ensure the following settings are properly configured:

* JavaScript code should not be executed under any circumstances
* Access to local files should be disallowed
* Access to external resources should be disallowed or limited if it is required

In many cases, the HTML code relies on external resources such as images and stylesheets. If they are part of the template, the web application should fetch these resources in advance and store them locally. We can then edit the HTML elements to reference the local copy of these resources such that no external resources are loaded. This allows us to set strict firewall rules that prevent all outgoing requests by the web server running the web application. This will prevent SSRF vulnerabilities entirely. However, if users need to be able to load external resources, it is recommended to implement a whitelist approach of external endpoints that resources can be loaded from. This prevents the exploitation of SSRF vulnerabilities by blocking access to the internal network.