For many businesses, it’s not until after they suffer from a security breach that web security becomes a priority. An effective web security approach must be defensive and proactive. Let’s look at 5 most common web security vulnerabilities and how you can prevent them.

1. **Injection Flaws**

An injection flaw occurs when you fail to filter untrusted input. It can also happen if you pass unfiltered data to SQL server (SQL Injection), to LDAP server (LDAP injection), to the browser (Cross Site Scripting), or anywhere else. The result is that the attacker is able to inject commands to all these entities. Thus, they can hijack a user’s browser and cause loss of data.

It is preferable to filter anything your application receives from an untrusted source according to a whitelist. It is not recommended to use a blacklist since they are usually easier to bypass and getting them right is not easy. This is the reason why many antivirus software programs typically fail with blacklists.

1. **Missing Function-Level Access Control**

This is an authorisation failure. This means that whenever a function is called on a server, authorisation does not perform properly. Many a times developers tend to think that the server-side generated the UI and that the functionality that is not supplied by the server is inaccessible by the client. However, it is not that simple. Attackers can forge a request to the hidden functionality. The fact that UI does not make the functionality easy to access does not stop them.

To prevent this, authorization must always be done on the server side.

1. **Cross-Site Scripting (XSS)**

This vulnerability allows attackers to compromise user interactions with a vulnerable application. They give JavaScript tags to your application on input and the user’s browser executes it. The attacker can hence carry out any operations that the user can perform, and also access their data. In case the user has privileged access, the attacker can also gain full control over the application.

A simple preventive measure is to not return html tags to the client. This also has added advantage to defend against html injections. It is also imperative to filter user input validity as you receive it.

1. **Insecure Direct Object Reference**

A direct object reference means that an internal database key or file is exposed to the user. As a result, in case of broken or missing authorization, an attacker can access your application. Thus, it’s important to conduct authorisation consistently and properly, along with whitelisting the choices. Many a times, this entire problem is easily avoidable if you store the data internally and don’t rely on the client to pass it via CGI parameters.

1. **Security Misconfiguration**

Server and application misconfiguration is very common. The reason is that there are many ways you can go wrong while configuring. For instance, have a directory listing enabled on the server which can leak your sensitive data, running outdated software, not changing default passwords and keys and running the app with debug enabled in production. You can prevent this by preferably having an automated “build and deploy” method which runs tests on deployment.

We are Aardwolf Security provide detailed analysis of your application’s design, authorisation mechanisms, networking and external data sources. Get in touch with us today to get a quote for your Web Application Assessment.