Cross-site scripting

CVE-2020-9371

Aleksander Avastu

Introduction

Cross-site scripting (abbr. XSS) is a type of software security risk mostly associated with web applications which provides malware designers with the possibility of injecting client-side code into a website. Such code is most commonly used for redirecting users onto a scam website, bypassing security and accessing private information.

I hereby allow the University of Tartu to publicly display this paper until the year 2025.

Basic intro to XSS

1) Web service lacks proper sanitization for data given by users
2) User inputs malicious code instead of proper data for an input field
   a. This is usually done by using a <script> tag in the input field
3) Server executes the malicious code due the lack of proper sanitization

These attacks are usually caused by the server executing server-side code with user submitted parameters without properly sanitizing the user data.

Types of XSS

XSS attacks are most commonly divided into two categories: persistent and non-persistent attacks. Whilst persistent attacks are usually more dangerous, either can be used for a multitude of malicious intents.

Non-persistent XSS attacks are the most common type of security risk for a web application. This can happen when user inputted data is used in server-side code to display data in another view without proper sanitization.
Persistent XSS attacks aim to save the injected script into the server database. This allows the malicious code to be executed by every user of the web service. This is the greatest threat with persistent XSS attacks. This means that instead of infecting only one user’s session with malicious code, all users will be vulnerable.

**XSS prevention**

The only thing required for the prevention of XSS attacks is sanitization of user inputted data. Most web frameworks these days come with built in functionality for sanitizing and validating inputs. That is it. That is all you have to do.

It can be hard remembering to always sanitize the inputs but most frameworks require special actions to access user inputted data without sanitizing it first.

For example, the PHP framework Laravel can access data from the server side `{{ $data }}` wit double curly braces, but this automatically sanitizes the input. In order to access unsanitized data, the developer must use `{!! $data !!}` which very clearly lets the developer know that they are using some potentially dangerous data.

**Most famous XSS attack**

The most famous XSS attack would have to be the Samy virus. The Samy virus was created by Samy Kamkar in 2005. The virus was an XSS worm, meaning that the virus was being spread by other users (a persistent XSS attack). The Samy virus was a rather harmless worm, designed for the popular at the time website called MySpace. The worm had two main functions in addition to self replicating. The first function was to update the bio of the user viewing the infected page with the phrase “but most of all, samy is my hero”. The second function of the worm was to send the user Samy Kamkar a friend request. Due to the nature of the worm, the virus spread faster as time went on. Withing 20 hours of the release of the worm, over one million accounts had been infected.
How to prevent the specific CVE

The problem brought out in the description of CVE-2020-9371 was that the calendar name field allowed for XSS. In order to fix this security issue the developers have to sanitize the input of the calendar name field. Since this an PHP application written with the Wordpress framework the underlying field can be sanitized using the PHP function htmlspecialchars($input) or the Wordpress function sanitize_text_field($input).

I tried my best to find the correct line in the project but since it does not have a public git repository and search function here (https://plugins.trac.wordpress.org/browser/appointment-booking-calendar/tags/1.3.34) is basically non-existent, I could not find the corresponding line since the master file called cpabc_appointments.php imports quite a few files, which in turn also import a great number of files.