|  |  |
| --- | --- |
| **Overall Risk Level** | Moderate |
| **Impact Level** | High |
| **Likelihood of Occurrence** | Low |
| **Application/Application URL** | Amazon  https://www.amazon.com/ |
| **Security Control** | CWE-79: Improper Neutralization of Input During Web Page Generation (‘Cross-site Scripting’) |
| **Recommended Fix Action** | Assume all input is malicious. Use an “accept known good” input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. |
| **Reporting Date** | Jan 01, 2020 |

Vulnerability Description:

Stored Cross-site Scripting (XSS) is the most dangerous type of Cross Site Scripting. Web applications that allow users to store data are potentially exposed to this type of attack. This chapter illustrates examples of stored cross site scripting injection and related exploitation scenarios. Stored XSS occurs when a web application gathers input from a user which might be malicious, and then stores that input in a data store for later use. The input that is stored is not correctly filtered. As a consequence, the malicious data will appear to be part of the web site and run within the user’s browser under the privileges of the web application. Since this vulnerability typically involves at least two requests to the application, this may also called second-order XSS. Stored XSS does not need a malicious link to be exploited. A successful exploitation occurs when a user visits a page with a stored XSS. Stored XSS is particularly dangerous in application areas where users with high privileges have access. When the administrator visits the vulnerable page, the attack is automatically executed by their browser. This might expose sensitive information such as session authorization tokens…

Vulnerabilitiy Finding Summary:

In this case, the Amazon application is vulnerable to Stored XSS via the "k" parameter using the following Payload:   
"><details open ontoggle=prompt`0`>.

Detailed Vulnerability Step Actions:

The Penetration Testing Team navigated to the following URL:  
https://www.amazon.com/

The team entered the following Payload into the search bar:  
"><details open ontoggle=prompt`0`>

The team clicked on the submit button

Upon the page loading the team was able to verify the execuction of JavaScript code  
due to a prompt boxing prompting with the value of 0