Open Source Github Repo Taken: xpleaf/Blog_mini: An Open Source Blog System that developed with Flask. (github.com)

Bug found is Cross-Site Scripting Attack (XSS)

CVE-2019–12308

Background:

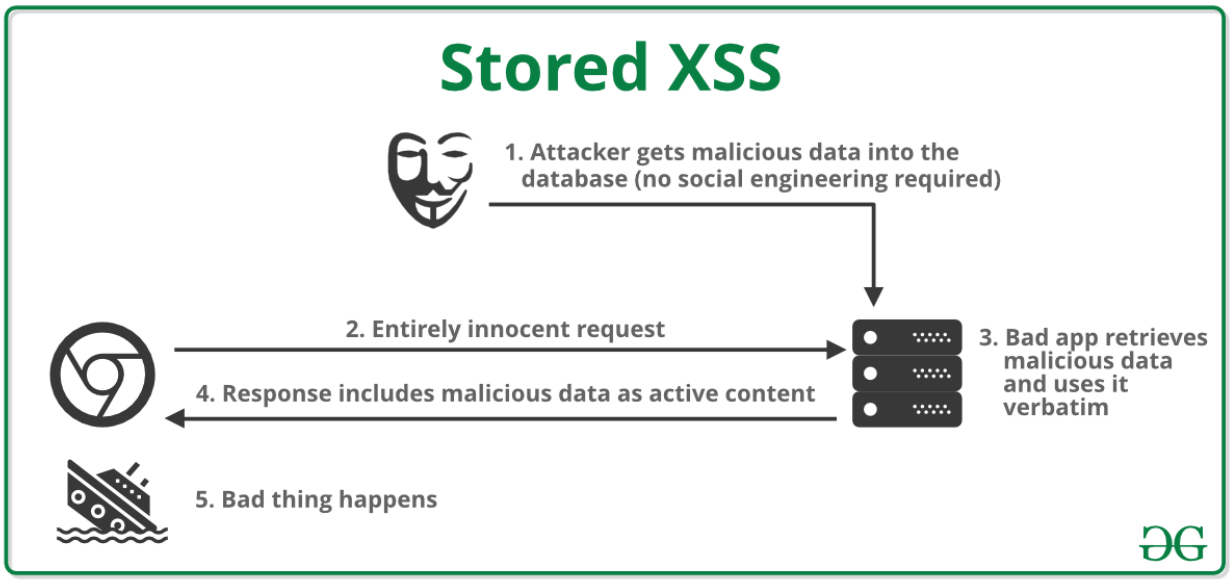
Cross Site Scripting (XSS) : It is a type of security vulnerability that can be located in a few internet applications. XSS attacks enable attackers to inject client - side scripts into web pages considered by different users. A move-website online scripting vulnerability can be used by attackers to bypass get entry to controls including the equal-starting place policy. cross-website online scripting carried out on web sites accounted for more or less eighty four% of all protection vulnerabilities documented by means of Symantec up until 2007. XSS outcomes vary in range from petty nuisance to good sized protection chance, depending at the sensitivity of the facts handled with the aid of the vulnerable website online and the character of any protection mitigation applied by using the website online's proprietor community.

It can be of multiple types:

i) Reflected XSS
ii) Stored XSS
iii) Dom based XSS
iv) Self XSS
v) Mutated XSS

It is a kind of Stored XSS Vulnerability

Stored XSS:



Source: GFG

Stored pass-web site scripting (also known as 2nd-order XSS) occurs while an utility gets data from a relied on source and integrates that facts into its modern day HTTP responses in an insecure manner.

suppose a website lets in users to publish remarks on blogs, that are exhibited to other customers. customers publish feedback the use of the HTTP request. If an script is stored in any manner in blog then when a website renders working on javascript it runs the script by default reading stored data of the user. This injection can be misused on various manner.

About Open-Source Project:

> It's a Chinese maintained repository, used for mini-blog system.
>
> Blog_mini is an open source blogging system, developed in Python, with a concise interface and powerful back-office management, with it, one can easily set their personal blog site!

Bug reported PR: [Cross Site Scripting Vulnerability in Latest Release · Issue #43 · xpleaf/Blog_mini (github.com)]

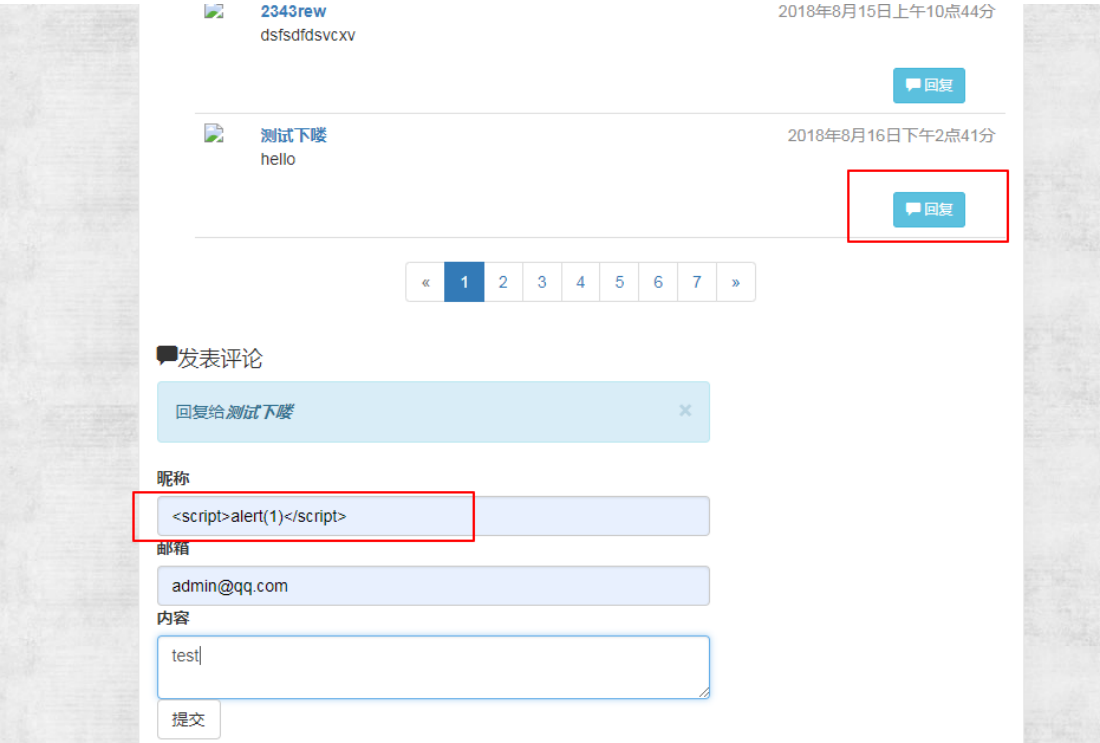
Steps to follow:

i) Select one article detials, like: **http://122.152.231.228:8080/article-detials/4**

ii) **Find the article** comment or **create new comment**.

iii) Reply the comment, and in the nickname section enter the XSS payload as: <script>alert(1)</script>

iv) then click submit button.



IMPACT:

These can have a huge impact on the hosted website, a user can run or inject JavaScript code and this may lead to deletion of some article or Updation or changes in any article. For example, if I rename the name position with some malicious script to install or download any malicious script to the users, Then, whoever opens that website. The website will default run that script, and it will install JavaScript embedded malicious file to the user's PC, which can lead to high risk to users PC. This can impact the business loss of the website of the company. This can be used to have spreading misinformation. Attacker can redirect the user to some URL leading to poisoning of the cookies, Keylogging, etc. leading to Account hijacking, data leaking, etc.

Solution:

Just by encoding the input data entered in the html data before storing we can resolve this issue easily by replacing characters such as <, >, /,&,",'with < etc as these are the basic characters that are used while injecting stored XSS error using javascript function or for Django python we can encode url using below library (Python3.7)

```
function escapeOutput(toOutput){
    return toOutput.replace(/\&/g, '&')
        .replace(/\</g, '<')
        .replace(/\>/g, '>')
        .replace(/\"/g, '"')
        .replace(/\'/g, ''')
        .replace(/\//g, '/');
}

```

```
import urllib.parse
url = 'website.php?name=alert("XSS")'
print(urllib.parse.quote(url))
# URL OUTPUT website.php%3Fname%3D%3Cscript%3Ealert%28%22XSS%22%29%3C/script%3E
```

Using above functions this will make the code uninjectable and will be stored and displayed as normal text instead of a JavaScript alert function.