DAY6_advent_of_the_cyber

This year, Santa wanted to go fully digital and invented a "Make a wish!" system. It's an extremely simple web app that would allow people to anonymously share their wishes with others. Unfortunately, right after the hacker attack, the security team has discovered that someone has compromised the "Make a wish!". Most of the wishes have disappeared and the website is now redirecting to a malicious website. An attacker might have pretended to submit a wish and put a malicious request on the server! The security team has pulled a back-up server for you on `10.10.64.95:5000`. Your goal is to find the way the attacker could have exploited the application.

By Swafox

What is XSS?

Cross-site scripting (XSS) is a web vulnerability that allows an attacker to compromise the interactions that users have with a vulnerable application. Cross-site scripting vulnerabilities normally allow an attacker to masquerade as a victim user, and carry out any actions that the user is able to perform. If the victim user has privileged access within the application (i.e admin), then the attacker might be able to gain full control over all of the application's functionality and data. Even if a user is a low privileged one, XSS can still allow an attacker to obtain a lot of sensitive information.

Why does it work like that?

XSS is exploited as some malicious content is being sent to the web browser, often taking the form of JavaScript payload, but may also include HTML, Flash, or any other type of code that the browser may execute. The variety of attacks based on XSS is almost limitless, but all of them come down to exactly two types: stored and reflected.

Types of XSS

Stored XSS works when a certain malicious JavaScript is submitted and later on stored directly on the website. For example, comments on a blog post, user nicknames in a chat room, or contact details on a customer order. In other words, in any content that persistently exists on the website and can be viewed by victims.

```
<!-- Normal comment--> <p>      Your comment goes here </p> <!--Malicious comment--> <p>      <script>          evilcode()
     </script> </p>
```

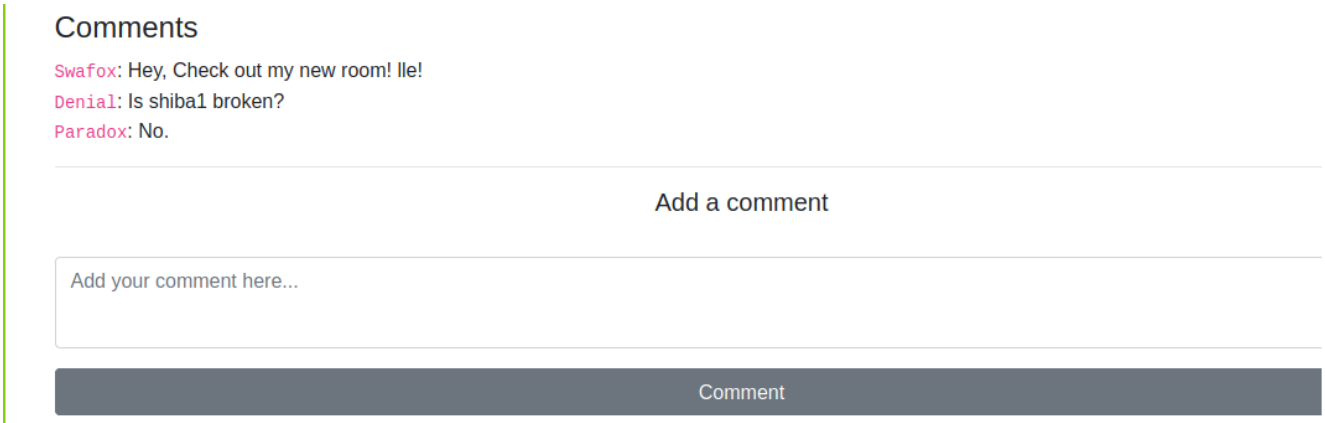
Let's say we have a website with comments (Code above). A normal comment is put under `<p></p>` tags and displayed on the website. A malicious user can put `<script></script>` tags in that field to execute the `evilcode()` function every time a user sees this comment.

Stored XSS gives an attacker an advantage of 'injecting' malicious JavaScript behind images. By using `<img>` attribute it is possible to execute custom JS code when the image is viewed or clicked. For example:

```
<img src='LINK' onmouseover="alert('xss')">
```

In this case, an attacker embeds an image that is going to execute `alert('xss')` if the user's mouse goes over it.

Say we have a web application that allows users to post their comments under the post.



An attacker can exploit this by putting an XSS payload instead of their comments and force everyone to execute a custom javascript code.
This is what happens if we use the above `<img>` payload there:

A malicious picture executes a custom `alert('xss')` once being viewed. This is the most common example of stored XSS.

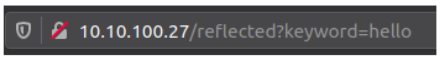
Reflected is another type of XSS that is carried out directly in the HTTP request and requires the attacker to do a bit more work. An example of this could be malicious javascript in the link or a search field. The code is not stored on the server directly, meaning that a target user should compromise himself by clicking the link.

Here's a quick example of an URL with malicious javascript included:

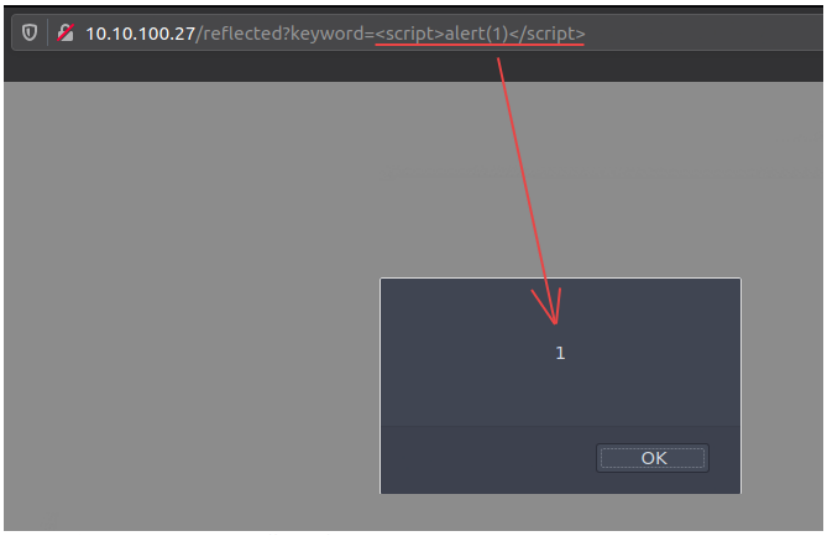
```
<https://somewebsite.com/titlepage?id=> <script> evilcode() </script>
```

Any user that clicks on the link is going to execute the `evilcode()` function, eventually falling under the XSS attack.

Let's say a website is using a query string **keyword** in its URL `10.10.100.27/reflected?keyword=hello` like so:



A search query is put after this keyword parameter. The XSS can be exploited by putting a payload **instead** of the search query.
The url starts with `10.10.100.27/reflected?keyword=`. By adding text onto the keyword, we can perform reflected XSS like `10.10.100.27/reflected?keyword=<script>alert(1)</script>` which results in an alert box with 1 on our screen.



Bingo! The XSS was successfully exploited!

How to detect XSS?

Both reflected and stored XSS vulnerabilities can be detected in a similar way: through the use of HTML tags, such as `<h1></h1>`, `<b></b>` or others. The idea is to try out inputting text with those tags and see if that produces any differences. Any change in text size or color immediately indicates an XSS vulnerability.

But sometimes, it might be challenging to find them manually, and of course, we cannot forget about the basic human error. Happily, there's a solution for that! OWASP ZAP is an open-source web application security scanner. It can automatically detect web vulnerabilities.
You can launch ZAP by going to 'Applications -> Web -> Owasp Zap' on the attack box:

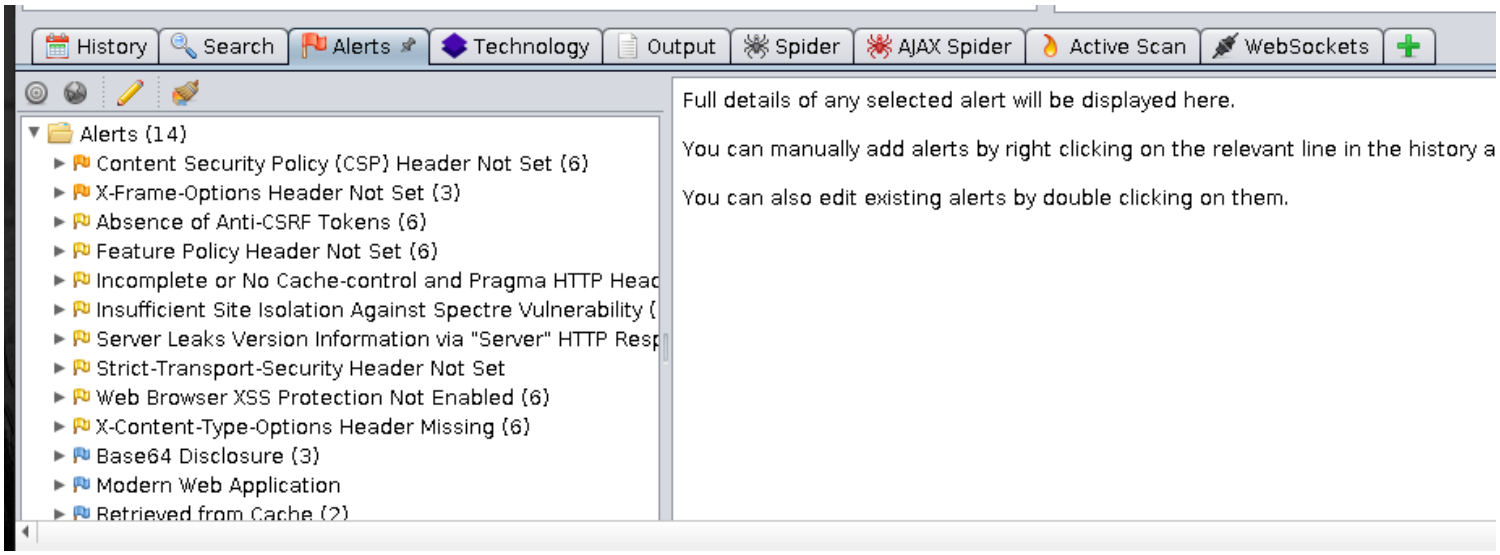
Bonus: Mitigating XSS

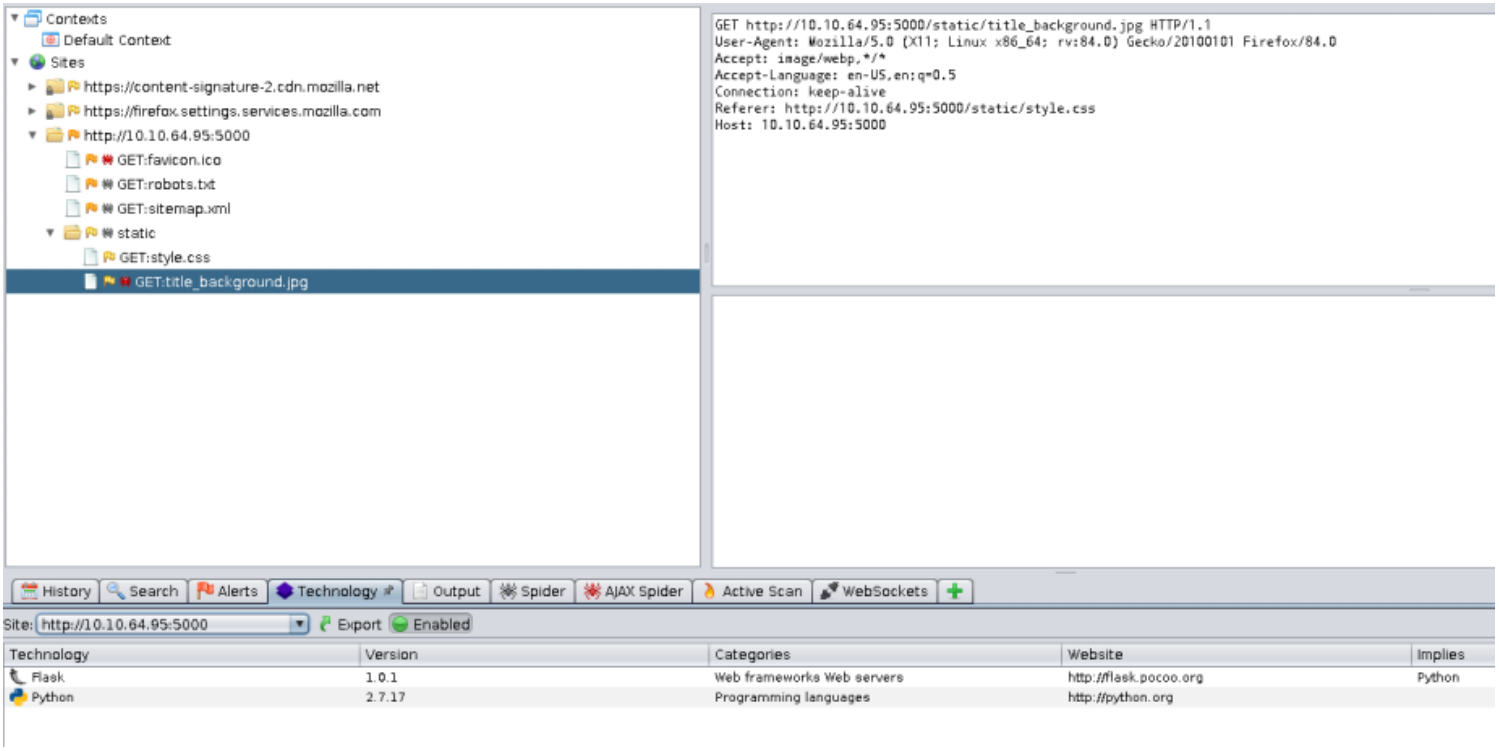
The rule is simple: all user input should be sanitized at both the client and server-side so that potentially malicious characters are removed.
There are libraries to help with this on every platform.
Smart developers should always implement a filter to any text input field and follow a strict set of rules regarding processing the inputted data.
For more info about this, check out OWASP's guide:

OWASP/CheatSheetSeries

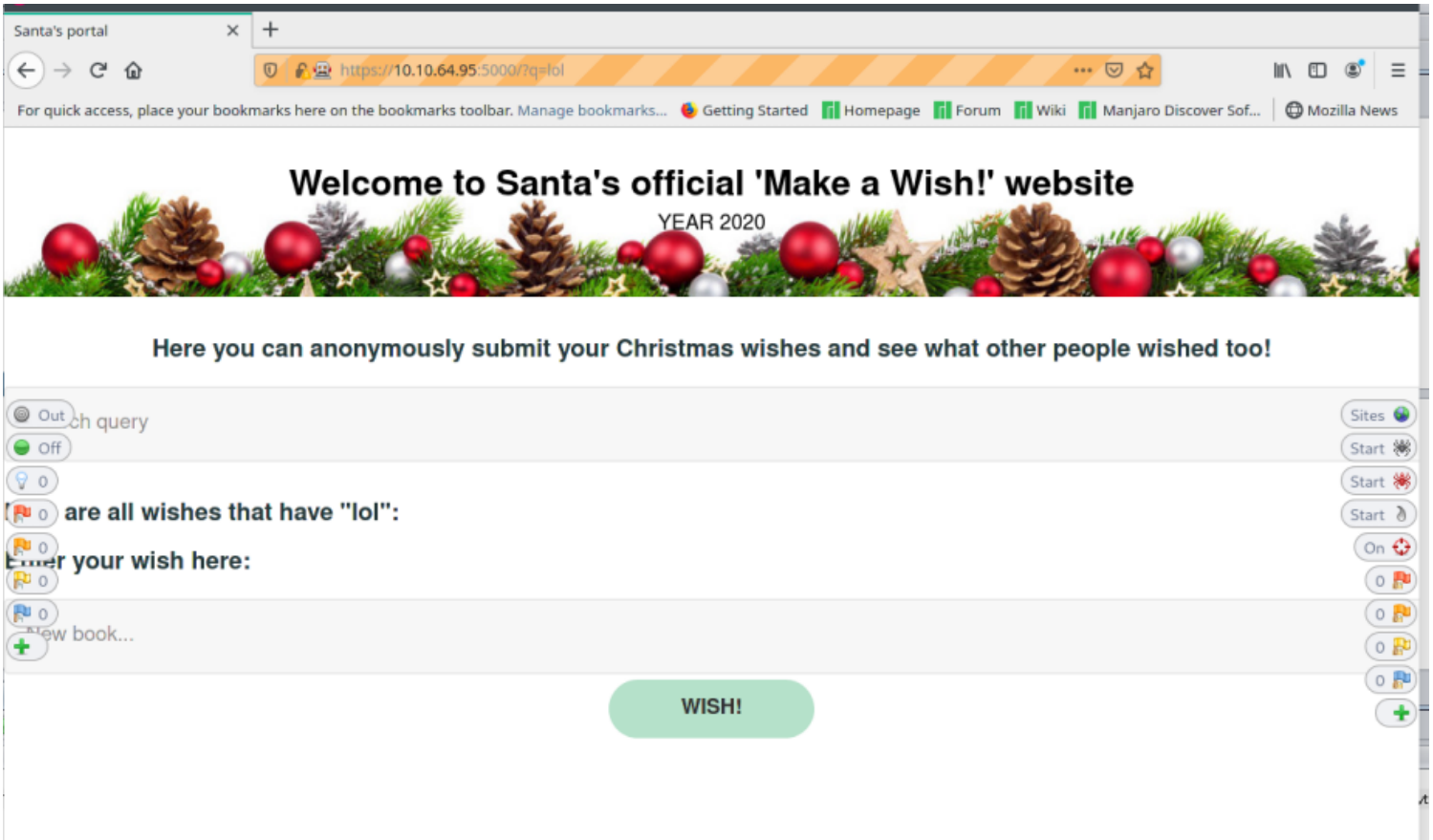
Now exercise:

Running zaproxy , we get:

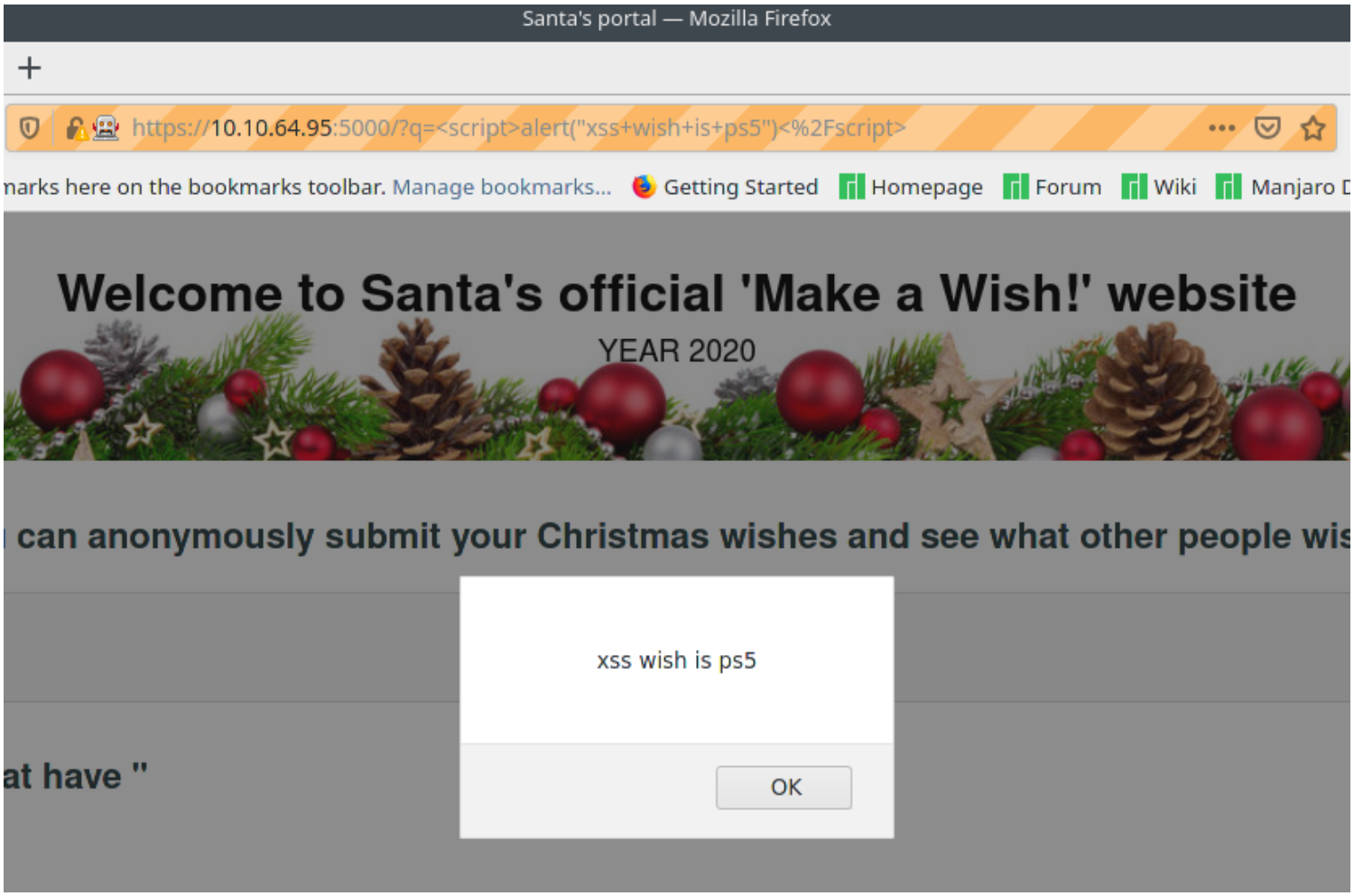




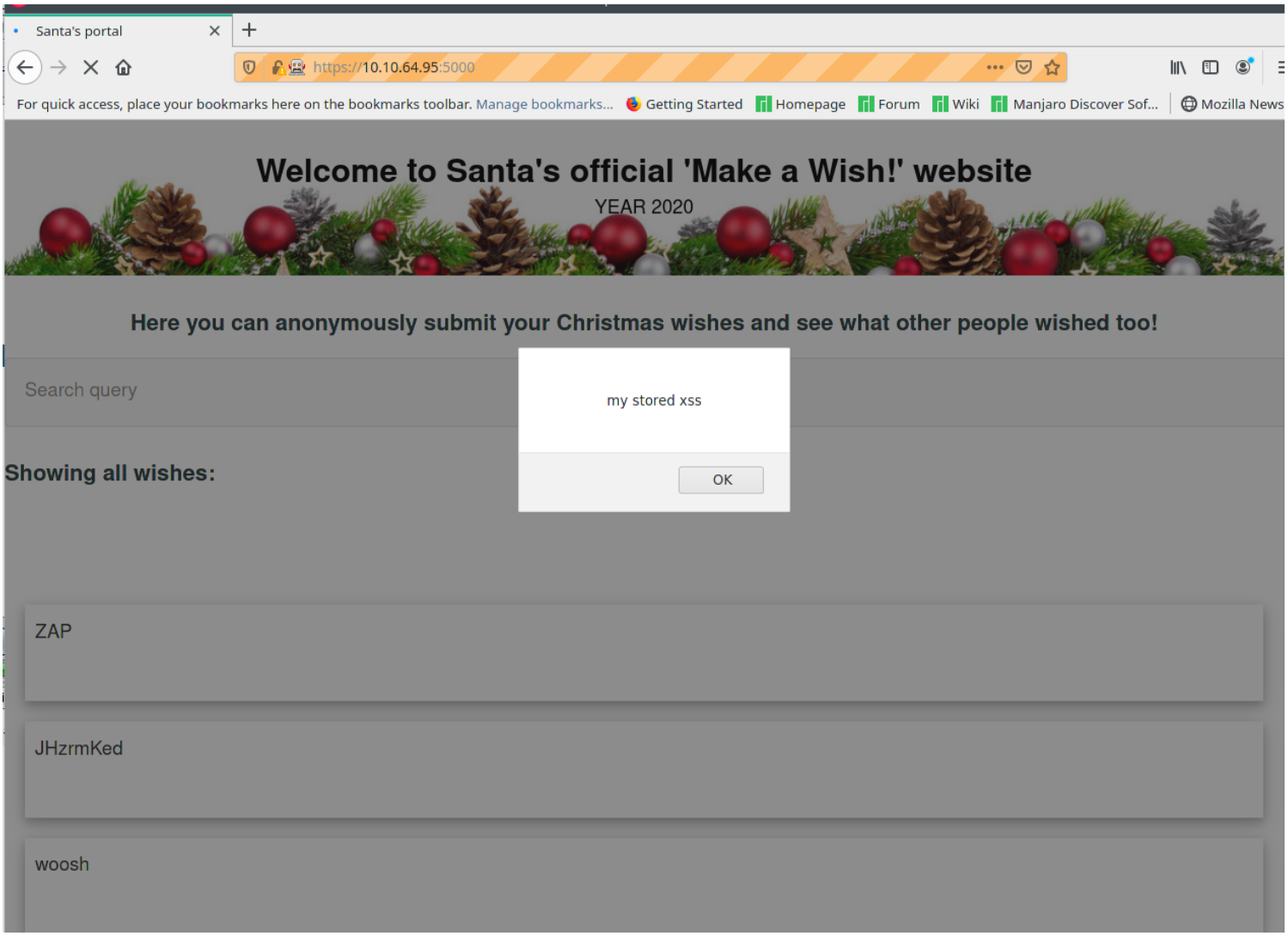
going to the given address:

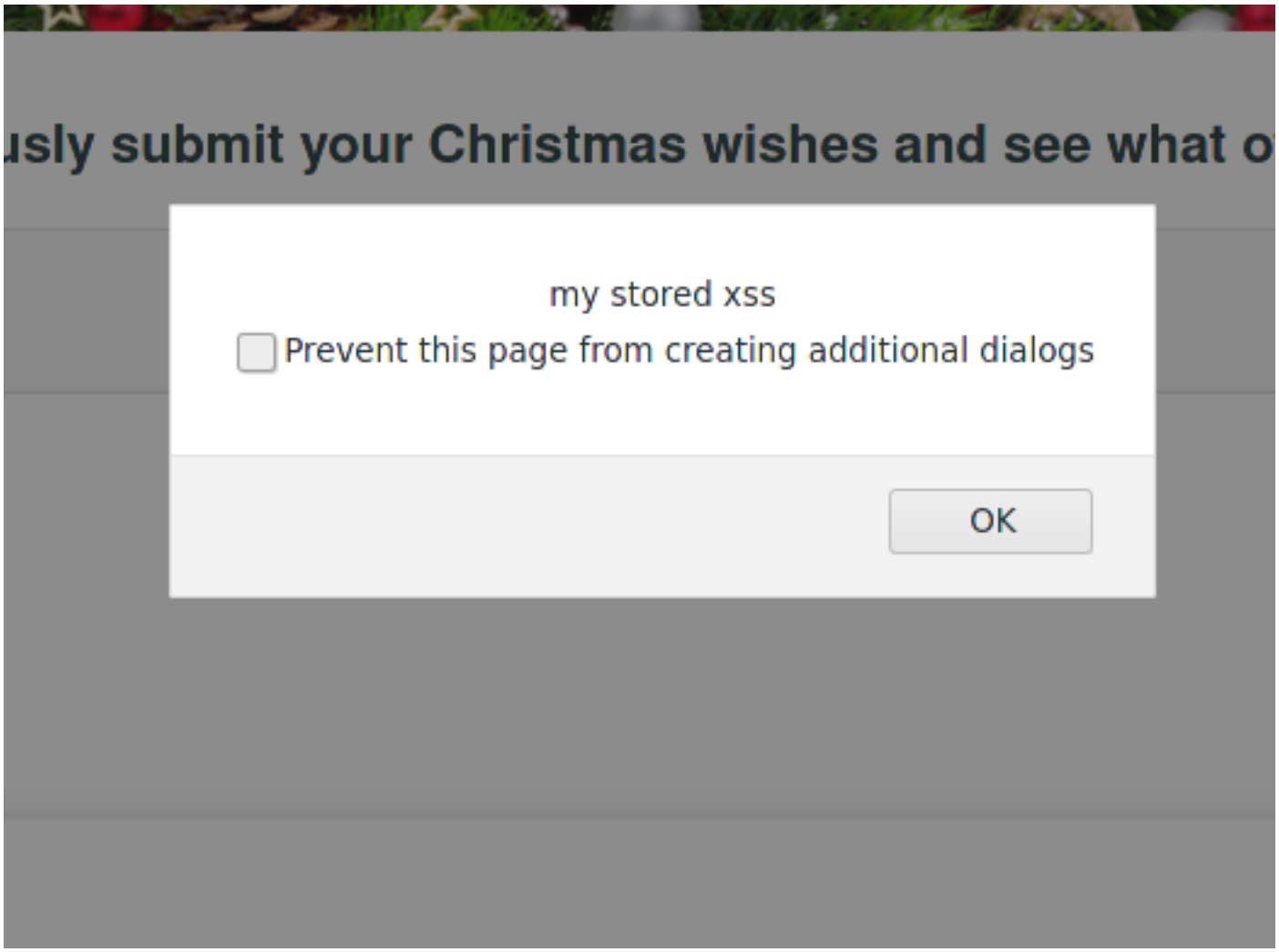


This site is vulnerable to “Reflected XSS”



This site is vulnerable to “Stored XSS” too:


Santa's portal
https://10.10.64.95:5000
For quick access, place your bookmarks here on the bookmarks toolbar. Manage bookmarks...
Getting Started
Homepage
Forum
Wiki
Manjaro Discover Sof...
Mozilla News
Welcome to Santa's official 'Make a Wish!' website
YEAR 2020
Here you can anonymously submit your Christmas wishes and see what other people wished too!
Search query
my stored xss
OK
Showing all wishes:
ZAP
JHzrmKed
woosh


usly submit your Christmas wishes and see what o
my stored xss
Prevent this page from creating additional dialogs
OK