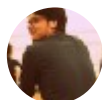


# How I Found XSS By Searching In Shodan



D1vy4n5hu 5hukl4

Follow

Aug 4 · 3 min read

*It is just another XSS blog.*

Anyways I was pentesting for a private program, I started with information gathering and meanwhile started dirbuster in the backend. I started with google and shodan. The moment I searched **companyname.com** in the search. A result appeared with a response in the shodan query. I opened the link and it was having an undeveloped WordPress page with a search bar. I entered XSS payload and it popped up. Then I changed the request method

from POST to GET and it worked like charm.

We will assume that the website is *https://companyname.com*.

## Description:

A code injection attack requires an attacker to enter malicious code (typically JavaScript or HTML) into an application in a data field that will be displayed to another user of the application. When the victim user views that data field, the JavaScript executes in the victim's browser and can perform malicious actions. One common attack would be for the JavaScript or HTML code to send the victim's cookies to the attacker.

## Impact:

The attacker injects a malicious JavaScript code or HTML code in the vulnerable parameter/user search field. Here it was a **reflected XSS**, which was discovered by Shodan query. This could be used to steal session token and cookie as it was part of the parent website which was running on the same server.

# Steps to reproduce:

Search the *companyname.com* in Shodan.

The screenshot shows the Shodan search engine interface. The search bar contains 'companyname.com'. The results show 3 total results. The top result is '301 Moved Permanently' from a domain in the United States, dated Sat, 06 Jul 2019 03:44:06 GMT. The second result is '200 OK' from Amazon.com, dated Wed, 17 Jul 2019 11:16:18 GMT. The third result is '301 Moved Permanently' from a domain in the United States, dated Sat, 03 Aug 2019 03:00:14 GMT. The second result is highlighted with a red box, showing the following details:

```
HTTP/1.1 200 OK
Server: [redacted]
Date: Wed, 17 Jul 2019 11:16:18 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Link: <https://www.[redacted].sp-json/>; rel=
Link: <https://www.[redacted]; rel=shor...
```

Shodan Query Result

1. Visit URL:

<https://www.companyname.com/eiy/redacted?s=test>

2. In the search bar enter the payload “><img src=x onerror=alert(document.domain)>

3. XSS alert popped up.

4. Proof Of Concept:

<https://www.companyname.com/eiy/redacted?s=%22%3E%3Cimg+src%3Dx+onerror%3Dalert%28document.domain%29%3E>



Reflected XSS

## Remediation:

To prevent data validation attacks such as cross-site scripting, I recommend a defence in depth strategy that includes both input validation and output sanitization.

The first step towards thwarting any data validation attack is to validate the input to prevent acceptance of any characters that may have special meaning within the application or the final destination of the data (in this case, the browser). The recommended way of handling input validation is to deny by default and only accept inputs that have expected values (i.e. a whitelist). Input validation routines should always check the data for length, range, type and format (perhaps using regular expressions).

Sanitization of malicious characters in the HTML served by the application is an equally important measure to prevent XSS. For instance, the character `<` would be encoded as `<` and, although appearing to the user as the less-than character, would not be interpreted by the client browser as the start of an HTML tag.

The best way to prevent XSS is to enforce output encoding, precisely Hex or URL encoding on user-controlled parameters which prevent the code to execute in the HTML page.

## Reference:

[https://www.owasp.org/index.php/XSS \(Cross Site Scripting\) Prevention Cheat Sheet](https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet)

<http://www.softwaremag.com/L.cfm?Doc=1006-12/2006>

<https://www.acunetix.com/websitesecurity/cross-site-scripting/>


Thanks

@justm0rph3u5

. . .

---

 Read this story later in [Journal](#).

 Wake up every Sunday morning to the week's most noteworthy stories in Tech waiting in your inbox. [Read the Noteworthy in Tech newsletter](#).

Security

Bug Bounty

Information Security

Ethical Hacking

Cross Site Scripting



45 claps



...



WRITTEN BY

**D1vy4n5hu 5hukl4**

Follow

Security Engineer | Threat Hunter | DevSecops | Linux Administrator



**Noteworthy - The Journal Blog**

Follow

The Official Journal Blog

[See responses \(1\)](#)

## More From Medium

More from Noteworthy - The Journal Blog

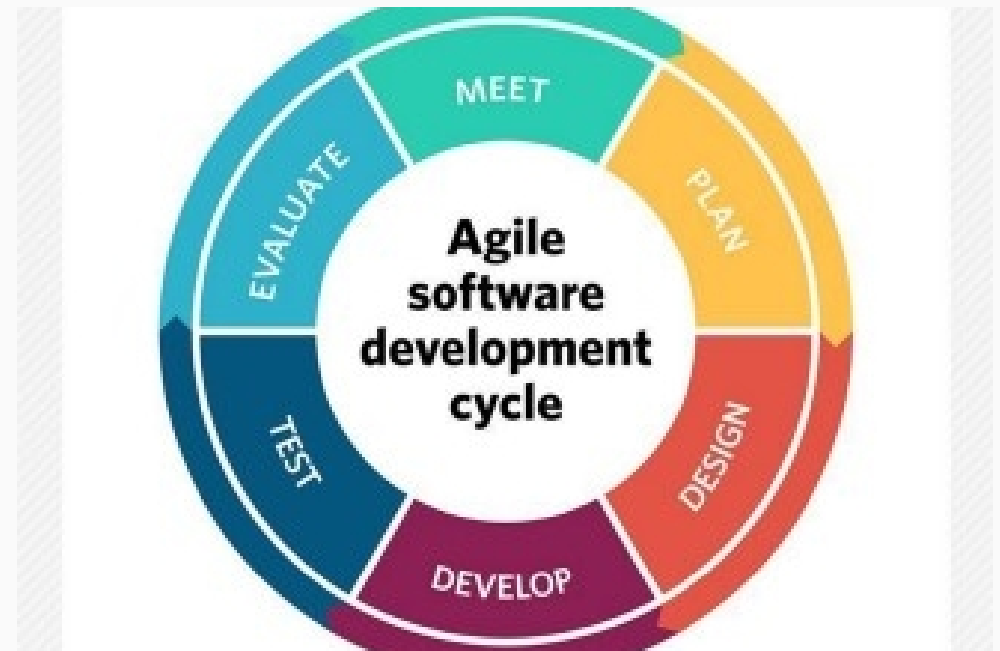
### How Agile Fails in Practice



Sam Redmond in Noteworthy - The Journal Blog  
Aug 22 · 16 min read ★



1.5K





More from Noteworthy - The Journal Blog

## Why you should choose Angular for your next front-end project



Sam Redmond in Noteworthy - The Journal Blog

Aug 30 · 7 min read ★



525



More from Noteworthy - The Journal Blog

## Next Steps: A Personalized Guide to Learning Web Development



Bridge School in Noteworthy - The Journal Blog

Aug 26 · 9 min read ★



549



## Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. [Watch](#)

## Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. [Explore](#)

## Become a member

Get unlimited access to the best stories on Medium — and support writers while you're at it. Just \$5/month. [Upgrade](#)

---

# Medium

[About](#)[Help](#)[Legal](#)