Shaheed Sukhdev College of Business Studies

University of Delhi

Post Graduate Diploma in Cyber Security and Law

Sakshi Garg • Roll no-23726 • Subject-Web Applications • Semester-1

# Cross Site Scripting

It is a web security vulnerability that allows an attacker to compromise the data present on the website
It allow attacker any action that user is able to perform and to access any user data

**What are the type of XSS Attack**

1. Stored XSS (Persistent XSS):
In this type of attack, the malicious script is permanently stored on the target server, typically in a database.
When a user accesses a specific page or resource that retrieves the stored data, the injected script gets executed.

2. Reflected XSS (Non-Persistent XSS):
In a reflected XSS attack, the injected script is included in the URL or a form input. The server reflects the injected script back to the user's browser, and it's executed in the context of the user's session. This type of XSS doesn't persist on the target server.

3. DOM-based XSS:
DOM-based XSS occurs when the client-side script manipulates the Document Object Model (DOM) of a web page. The attack is typically executed on the client side, making it difficult to detect on the server. Attackers manipulate the DOM by injecting malicious code that is then executed by the victim's browser.

4. Self-XSS (User-Induced XSS):
In self-XSS attacks, the attacker tricks the user into running malicious code in their own browser. This is often done through social engineering techniques, such as luring the user to paste and execute seemingly harmless code into the browser's developer console.

5. Blind XSS (Third-Party XSS):
Blind XSS occurs when the injected script does not directly affect the attacker but impacts other users. The attacker typically relies on someone with elevated privileges (such as an administrator) executing the malicious code unintentionally.

6. Multipart/form-data XSS:
This type of XSS attack involves exploiting vulnerabilities in the way a web application handles file uploads. Attackers can inject malicious scripts into files that are then uploaded to the server, and when other users download or access those files, the scripts are executed.

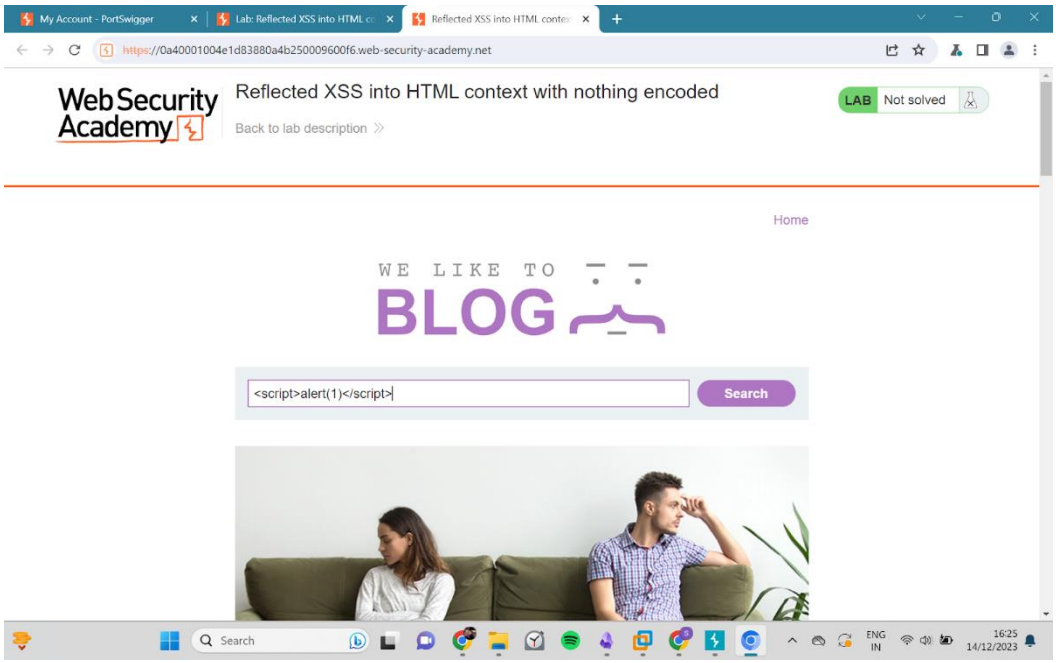
**Impact of XSS vulnerability:**
Cross site scripting attacks can have devastating consequences. Code injected into a vulnerable application can exfiltrate data or install malware on the user's machine. Attackers can masquerade as authorized users via session cookies, allowing them to perform any action allowed by the user account.
XSS can also impact a business's reputation. An attacker can deface a corporate website by altering its content, thereby damaging the company's image or spreading misinformation. A hacker can also change the instructions given to users who visit the target website, misdirecting their behavior. This scenario is particularly dangerous if the target is a government website or provides vital resources in times of crisis.

Severity Level - Medium

**Prevention-**
- Run untrusted HTML input through an HTML sanitization engine.
- Blacklist specific HTML tags that are at risk, such as the iframe, link, and script tags.
- Prevent client-side scripts from accessing cookies.

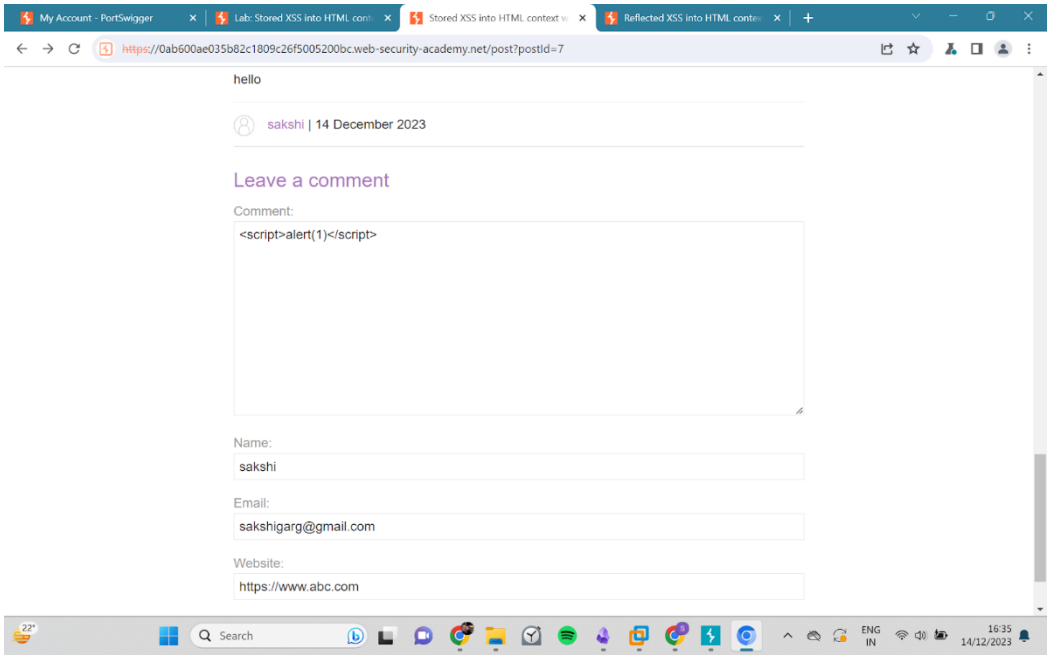
**POC**

Step 1- Open the website -> copy and paste the given query in the search bar



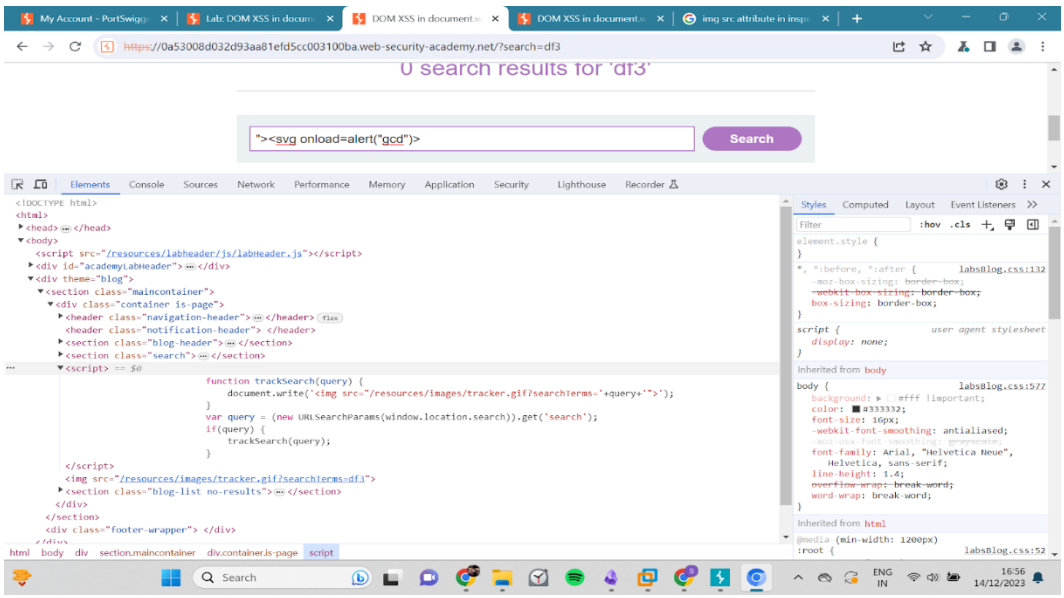

## Lab 2: Stored access into HTML context with nothing encoded

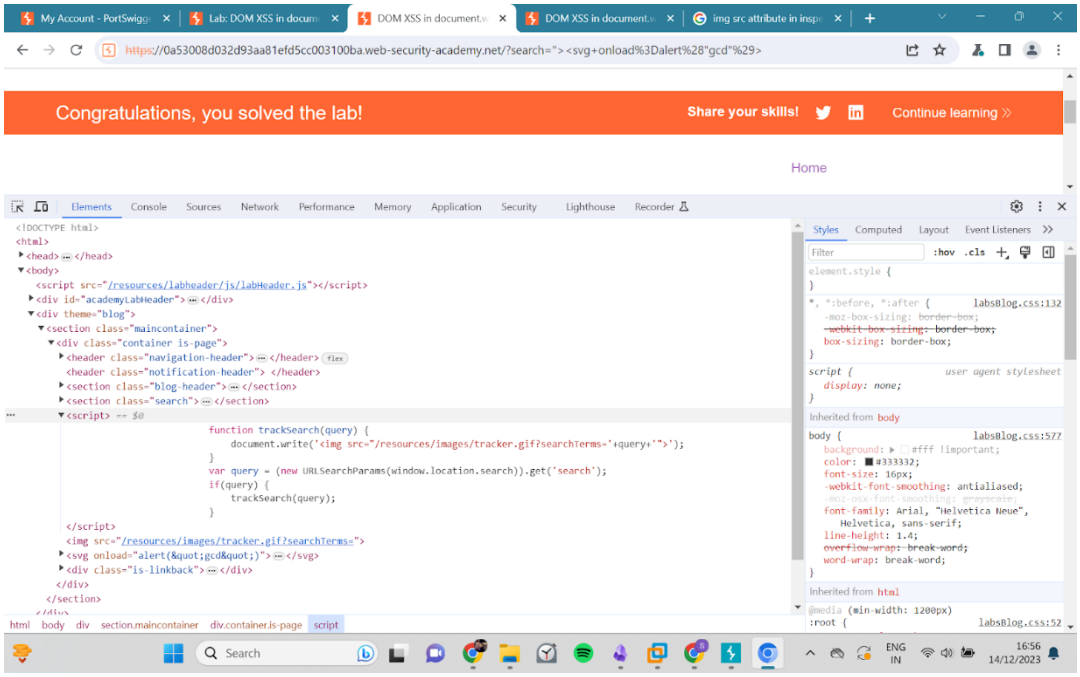
Access the lab -> paste the script in the comment box




# Lab 3: DOM XSS in document.write sink using source location.search

Severity - HIGH




My Account - PortSwigg
Lab: DOM XSS in docum
DOM XSS in document.w
DOM XSS in document.w
img src attribute in insp
https://0a53008d032d93aa81efd5cc003100ba.web-security-academy.net/?search="><svg+onload%3Dalert%28"gcd"%29>
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Styles
Computed
Layout
Event Listeners
Filter
:hov .cls
element.style {
}
*, *:before, *:after { labsBlog.css:132
-moz-box-sizing: border-box;
-webkit-box-sizing: border-box;
box-sizing: border-box;
}
script { user agent stylesheet
display: none;
}
Inherited from body
body { labsBlog.css:577
background: #fff !important;
color: #333332;
font-size: 16px;
-webkit-font-smoothing: antialiased;
font-family: Arial, "Helvetica Neue",
Helvetica, sans-serif;
line-height: 1.4;
overflow-wrap: break-word;
word-wrap: break-word;
}
Inherited from html
@media (min-width: 1200px)
:root { labsBlog.css:52
<!DOCTYPE html>
<html>
<head> </head>
<body>
<script src="/resources/labheader/js/labHeader.js"></script>

<section class="maincontainer">

<header class="navigation-header"> </header> flex
<header class="notification-header"> </header>
<section class="blog-header"> </section>
<section class="search"> </section>
<script> == $0
function trackSearch(query) {
document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');
}
var query = (new URLSearchParams(window.location.search)).get('search');
if(query) {
trackSearch(query);
}
</script>
<img src="/resources/images/tracker.gif?searchTerms=">
<svg onload="alert("gcd")"> </svg>

</section>
html body div section.maincontainer div.container.is-page script
Search
ENG
IN
16:56
14/12/2023