**Lỗ hổng DOM-based:**

**Khái niệm DOM:**

DOM là cây các thành phần của 1 trang web do trình duyệt tạo ra.

Những trang web có thể dùng Javascript để tác động vào các đối tượng của DOM và thuộc tính của nó. Thực tế đây là cách những trang web hiện đại hoạt động.

Tuy vậy, Javascript xử lí dữ liệu thiếu an toàn có thể bị tấn công. Những lỗ hổng DOM-based sinh ra khi Javascript lấy giá trị kẻ tấn công kiểm soát được là source và cho nó vào 1 hàm nguy hiểm là sink.

**Lỗ hổng Taint-flow:**

Rất nhiều lỗ hổng DOM có thể bắt nguồn từ cách Javascript tác động vào dữ liệu kẻ tấn công kiểm soát được.

**Taint-flow là gì?**

Để khai thác hay giảm thiểu những lỗ hổng loại này, đầu tiên bạn phải làm quen cơ bản với mối quan hệ giữa sources và sinks.

**Sources:**

1 source là 1 thuộc tính Javascript nhận dữ liệu mà kẻ tấn công kiểm soát đc. Ví dụ thuộc tính location.search là source vì nó đọc dữ liệu từ chuỗi truy vấn. Tóm lại, bất cứ thuộc tính nào có thể được kiểm soát bởi kẻ tấn công là source tiềm năng. Nó bao gồm cả Url tham chiếu (document.referrer), cookies của người dùng (document.cookies) và thông báo web

**Sinks:**

1 sink là 1 hàm Javascript nguy hiểm hoặc đối tượng Dom có thể gây ra ảnh hưởng xấu nếu dữ liệu kẻ tấn công được gửi đi.

Ví dụ hàm eval() là 1 sink vì nó xử lí đối số như Javascript.

Ví dụ về html sink là document.body.innerHTML vì nó cho phép kẻ tấn công tiêm html độc hại và thực thi code Javascript.

Về cơ bản, những lỗ hổng DOM-based sinh ra khi trang web gửi dữ liệu từ 1 source đến 1 sink, sau đó xử lí dữ liệu theo cách ko an toàn .

Source phổ biến là Url, được truy suất qua đối tượng location. Kẻ tấn công có thể tạo thành 1 đường dẫn gửi nạn nhân đến 1 ứng dụng có lỗ hổng với payload trong chuỗi truy vấn. Xem đoạn code sau:

goto = location.hash.slice(1)

if (goto.startsWith('https:')) {

  location = goto;

}

Đây là lỗ hổng Dom based chuyển hướng vì source location.hash được xử lí theo cách không an toàn. Nếu url chứa phân đoạn hash (#) mà bắt đầu với https, đoạn mã này sẽ lấy giá trị của location.hash và gán nó cho thuộc tính location của window. Kẻ tấn công có thể khai thác lỗ hổng này bằng url sau:

https://www.innocent-website.com/example#https://www.evil-user.net

Khi nạn nhân vào Url trên, Javascript sẽ đặt giá trị cho location là https://www.evil-user.net, tự động chuyển hướng nạn nhân đến trang web độc hại. Hành vi này có thể dễ bị khai thác để thực hiện 1 cuộc tấn công lừa đảo.

**Một số sources phổ biến:**

document.URL

document.documentURI

document.URLUnencoded

document.baseURI

location

document.cookie

document.referrer

window.name

history.pushState

history.replaceState

localStorage

sessionStorage

IndexedDB (mozIndexedDB, webkitIndexedDB, msIndexedDB)

Database

Một vài loại dữ liệu cũng có thể được dùng như sources để khai thác lỗ hổng taint-flow:

* Reflected-data
* Stored-data
* Web messages

Những sink có thể dẫn đến lỗ hổng DOM-based:

|  |  |
| --- | --- |
| **DOM-based vulnerability** | **Example sink** |
| DOM XSS | Document.write() |
| Open redirection | window.location |
| Cookie manipulation | document.cookie |
| Javascript injection | eval() |
| Document-domain manipulation | document.domain |
| WebSocket-URL poisoning | WebSocket() |
| Link manipulation | element.src() |
| Web message manipulation | postMessage() |
| Ajax request-header manipulation | setRequestHeader() |
| Local file-path manipulation | FileReader.readAsText() |
| Client-side SQL injection | ExecuteSql() |
| HTML5-storage manipulation | sessionStorage.setItem() |
| Client-side XPATH injection | document.evaluate() |
| Client-side JSON injection | JSON.parse() |
| DOM-data manipulation | element.setAttribute() |
| Denial of service | RegExp() |