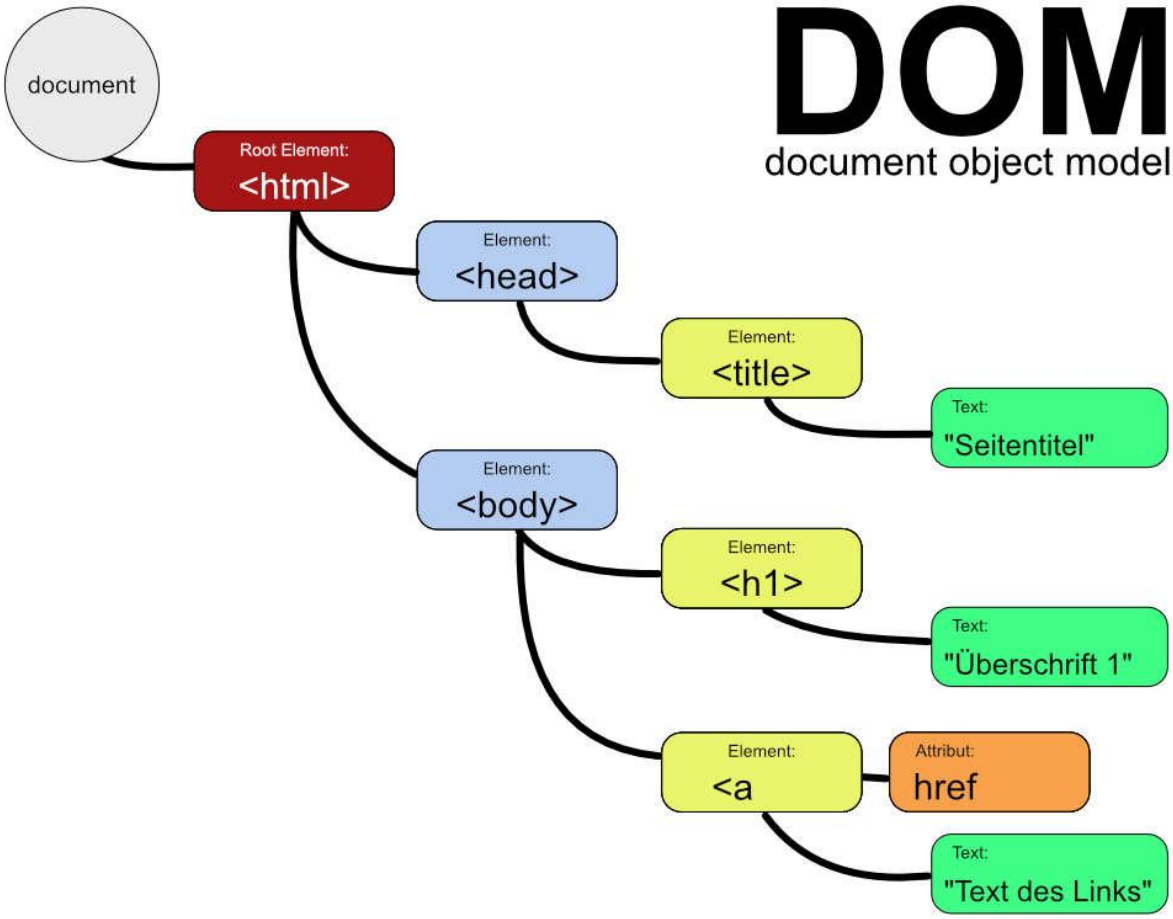
DOM-based vulnerabilities

تعريفها هي إختصار لـ Document Object Model وهو عبارة عن تمثيل هرمي للعناصر الموجودة في صفحة الويب كالآتي.



- الجافاسكريبت بتستخدم عشان تعالج (تنفذ، تعدل، تمسح) الـ DOM وده ف حد ذاته مش ثغرة، بل بالعكس كل المتصفحات بتستخدم الموضوع ده، إنما الثغرة بتحصل لما الجافاسكريبت تاخد من اليوزر قيمة هو قادر إنه يتحكم فيها وهنا ده بيُسمى الـ **Source** وبالتالي ممكن يحط malicious paylaods، بعدين تروح تدخل الـ input ده في function خطيرة وهنا بتُسمى **Sink**.

Taint-flow vulnerabilities

- معظم الثغرات المعتمدة على الـ DOM بتحصل بسبب مشاكل كيفية تعامل الـ Client-side مع الـ Attacker-controllable data يعني الداتا أو الـ input الي الـ Attacker يقدر يتلاعب بيه.

- فالي بيحصل إن الموقع بياخد الـ Soucre من اليوزر ويمرره للـ Sink function عشان يحصلها execution.
- وهنا لازم نعرف الفرق ما بين الـ Source, Sink:

Source

- هو عبارة عن JS property بتاخد قيمة الـ Attacker يقدر يتلاعب بيها، وده زي الـ location.search property بتاخد قيمة يقدر الـ Attacker يغيرها ويستغلها، فأي property يقدر الـ Attacker يتلاعب بيها فهي Source زي برضو (document.referrer, document.cookie) وكمان الـ Web messages.
- أشهر Source هو الـ URL والـ object الخاص بيه هو الـ location وده يقدر يستغله الـ attacker في إنه يحول الـ Victim لصفحة تاني، وهنا مثال للكود:

```
(goto = location.hash.slice(1
if (goto.startsWith('https:')) {  location = goto
                                {
```

Sink

- هو عبارة عن JS Function ممكن تسبب نتائج غير مرغوب فيها لو اتمررلها Attacker-controlable data زي eval() و document.body.innerHTML
- في نقدر نقول إن DOM-based vulnerabilities بتحصل لما نمرر قيمة من Source => Sink.

Common sources

The following are typical sources that can be used to exploit a variety of taint-flow vulnerabilities:

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

Which sinks can lead to DOM-based vulnerabilities?

The following list provides a quick overview of common DOM-based vulnerabilities and an example of a sink that can lead to each one. For a more comprehensive list of relevant sinks, please refer to the vulnerability-specific pages by clicking the links below.

DOM-based vulnerability	Example sink
DOM XSS LABS	`document.write()`
Open redirection LABS	`window.location`
Cookie manipulation LABS	`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()`

DOM-based XSS

- الـ DOM-based XSS هو أول نوع من ثغرات الـ DOM والي بيحصل بسبب Sink معينة وهي الـ document.write() وزي ما احنا عارفين عشان يحصل DOM-based vulnerability لازم Source & Sink.
- الـ DOM-XSS بتحصل بسبب إن الـ JS بتاخد قيمة من الـ Source والي غالبا بتكون URL زي window.location وتبعتها لـ Sink وغالبا بيكون eval() أو inner.HTML وده بيخلي الـ Attacker ينفذ اكواد جافاسكريبت خبيثة.
- يبقى عشان تنفذ الـ DOM-XSS ف أنت لازم تبعت داتا للـ Source => Sink => execute.

- من الحاجات الي ممكن نستخدمها كـ Sources برضو هي الـ Reflected Data, Stored Data, Web Message

1. Reflected Data
2. Stored Data

3. Web Message

Controlling the web message source

- هنا هنعرف تأثير الـ Web messages وازاي ممكن تكون source ونستخدمها عشان نحقق بيها Dom-based vunerability، ف لو الموقع مش بيتعامل بشكل صحيح مع الـ Web message الي جاية ف ده هيسبب الثغرة، كمثال ممكن الـ Attacker يعمل iframe فيه PostMessage() function بتبعت Web messages للـ eventListner ، وده ف حالة إن الصفحة مش بتتأكد من الـ origin الي جاي، وبعدين الـ eventListner() هيمرر الداتا للـ Sink function في الـ parent page وبالتالي الـ Web messages هنا كانت كـ source، وف الحالة دي كل الـ function الي بتتعامل مع الـ eventListner هتكون vulnerable وهتكون Sink.

سيناريوهات

- لما تلاقي صفحة فيها eventListener وبتقبل قيمة بدون ما تلفترها، جرب تحط الصفحة في iframe ولما يحصله load جرب تنفذ أمر، وخلي بالك الأمر
- لو الكود بيتنفذ داخل object.property زي location.href ف انت محتاج تمرر payload زي ده javascript:alert(1)I.
- لو بيستخدم JSON.parse ف انت محتاج تبعت الـ payload ويكون json.
- زي ما قولنا إن في الـ DOM-based Web message بتعتمد على إن الـ eventListener مش بيتحقق من الـ Origin أو الـ Host، ف ممكن تقابل بعض الـ restrictions زي :
  - ممكن يعمل تحقق عن طريق StartWith(), EndWith()I ف وقتها بتقدر تعمل bypass بسهولة زي https:normal-website.com.evil.com أو https:evil.comnormal-website.com

DOM clobbering

- الـ DOM clobbering هي مش ثغرة، ولكن تكنيك بنستخدمه عشان نقدر نحقن HTML code داخل صفحة، عشان تتلاعب بالـ DOM وكمان تغير في طريقة عمل الـ JS.
- الـ DOM clobbering مفيد لما تكون ثغرة الـ XSS صعب تتحقق، ولكن أنت بتقدر تتحكم في شوية HTML codes وعندهم id, name attribute بيكونوا whitelisted بالنسبة للـ HTML filter.
- أشهر مثال للـ DOM clobbering لما تستخدم الـ anchors element عشان تـ overwrite على global variable وبعدين التطبيق بيستخدم الـ global variable بطريقة غير آمنة زي مثلا إنه يعمل generate لـ dynamic URL.
  - لو لقيت OR operator || وهو جزء من global variable ومسموح لك تنفذ أكواد HTML ف تقدر تنفذ DOM clobbering.
- مثال تاني لإستخدام الـ DOM clobbering هو إنك تستخدم الـ form, input tags مع بعض عشان تـ clobber الـ properties، لما الديفلوبر يحاول يكون في السليم ف بيستخدم الـ attributes property عشان يـ loop على كل الـ attributes ولو لقي حاجة ضمن الـ blacklist زي onerror, onclick وغيرهم بيروح حاذفها.
  - ف أنت بتكتب كود زي ده <f<form onclick=alert(1)><input id=attributes>Click me ف بالرغم من إنه هيعمل Loop للـ attributes الي تبع الـ form ولكن مش هيقدر يحذف حاجة ضمن الـ form لإن الـ attributes property حصلها clobber بالـ input وبالتالي مش هيحذف الـ onclick property.
  -
  -
  -

How to test for DOM-based cross-site scripting

1.Testing HTML sinks

-