CompSec

Extending a Hand to Attackers

Browser Privilege Escalation Attacks via Extensions

USENIX Security 2023

Young Min Kim, Byoungyoung Lee

SEOUL NATIONAL UNIVERSITY

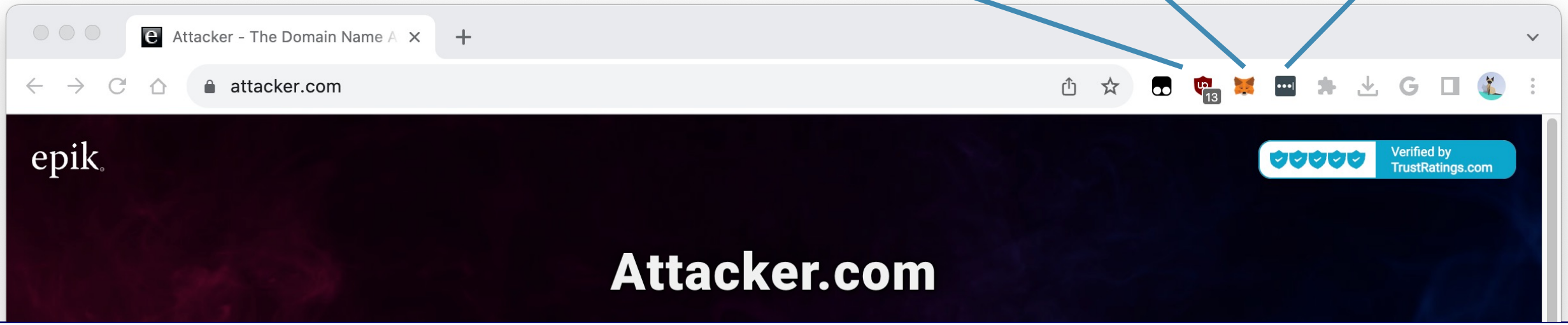
TL;DR: What's Possible?

Visit

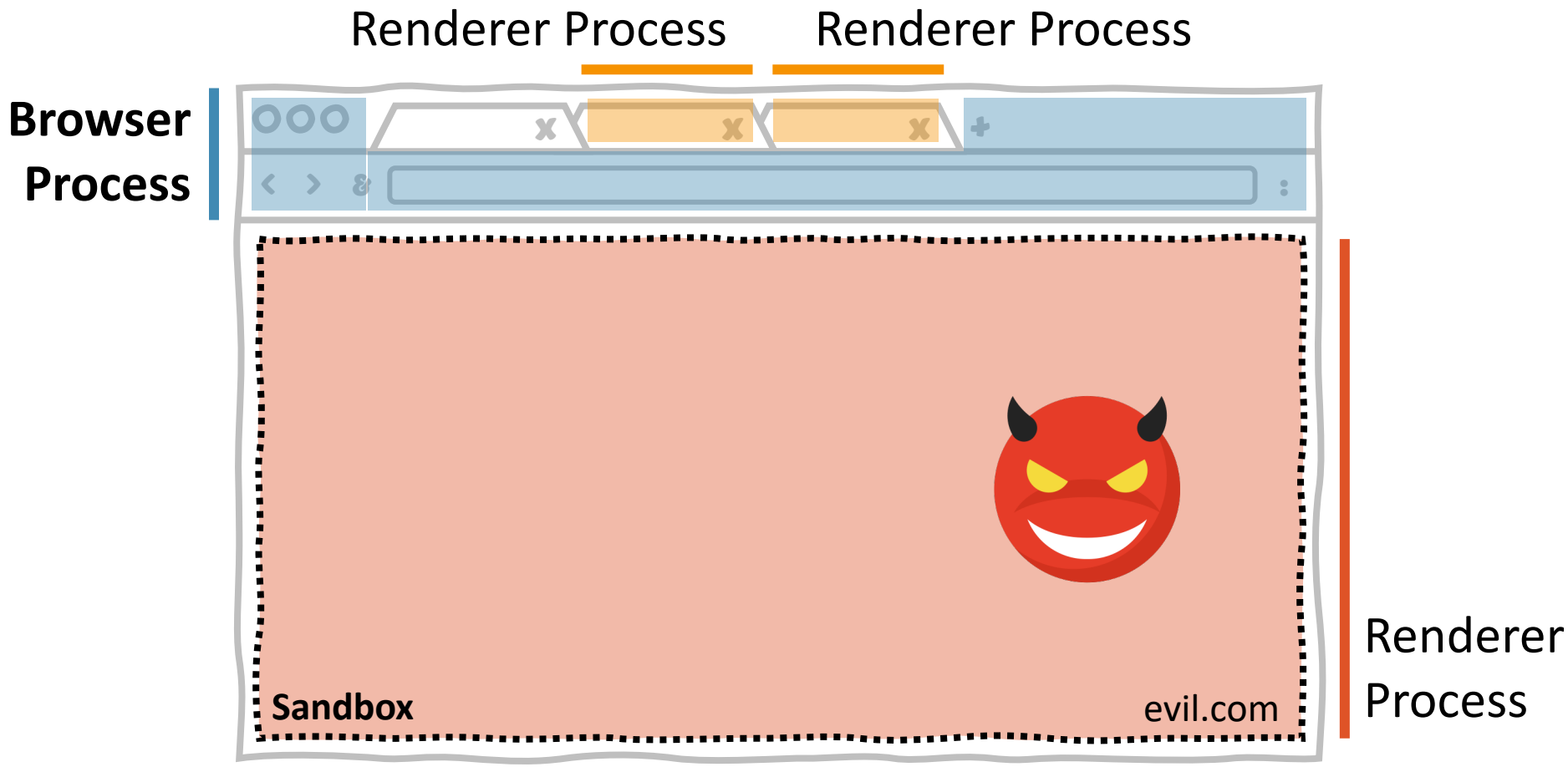
UXSS
(universal cross
-site scripting)

Steal
Coins

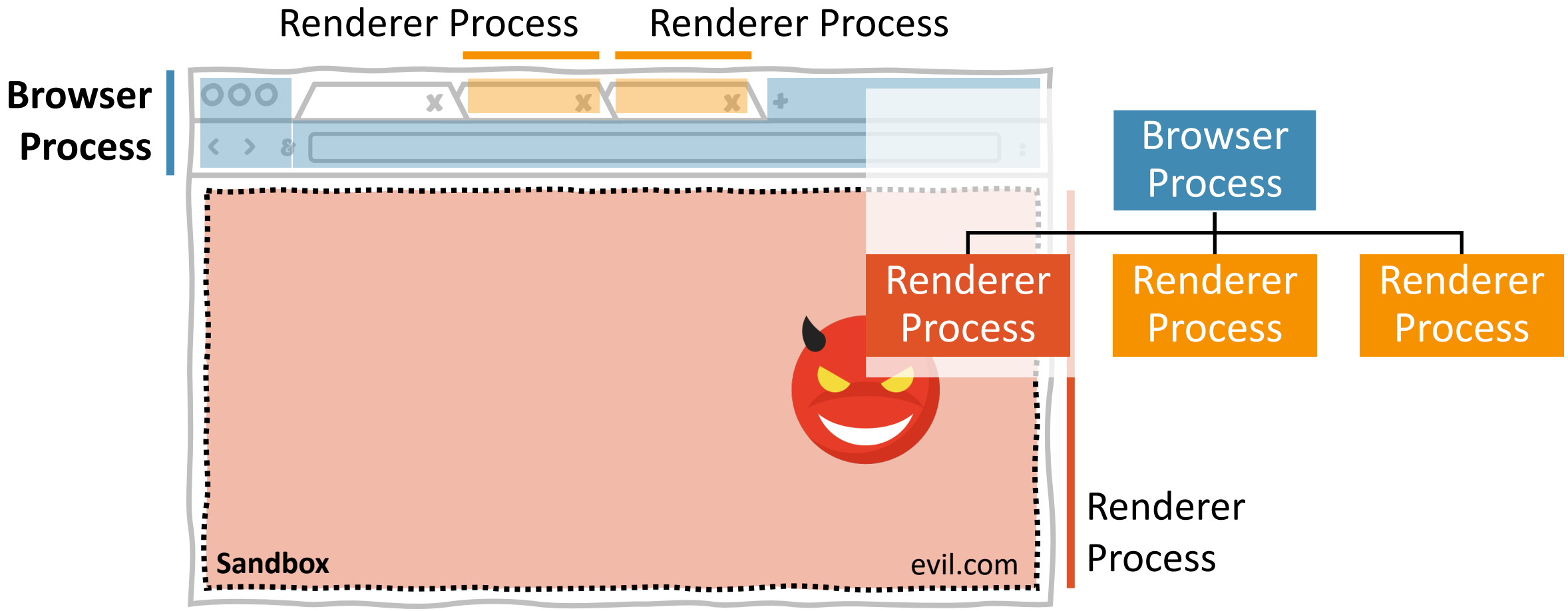
Steal
Passwords



Multi-Process Architecture

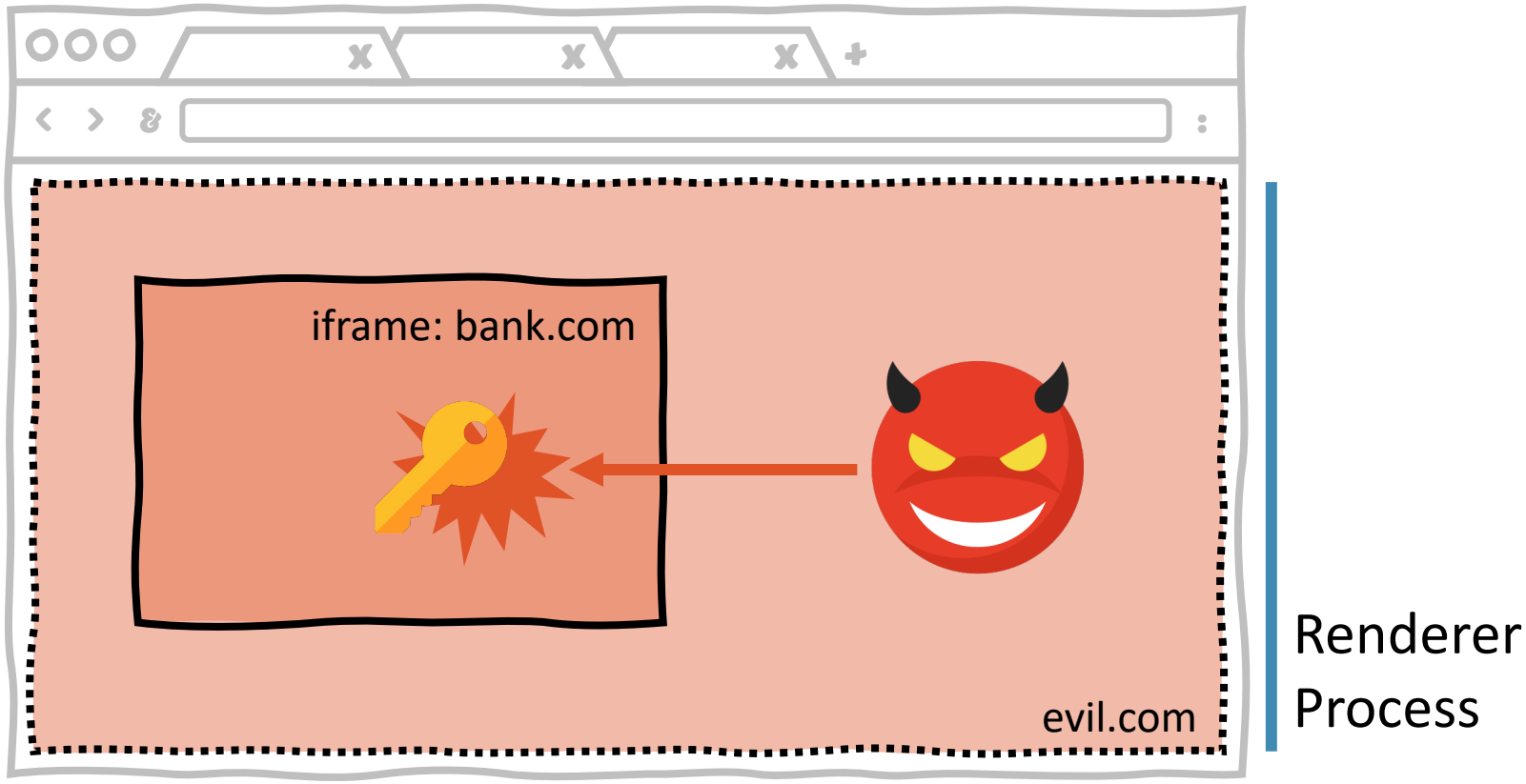


Multi-Process Architecture

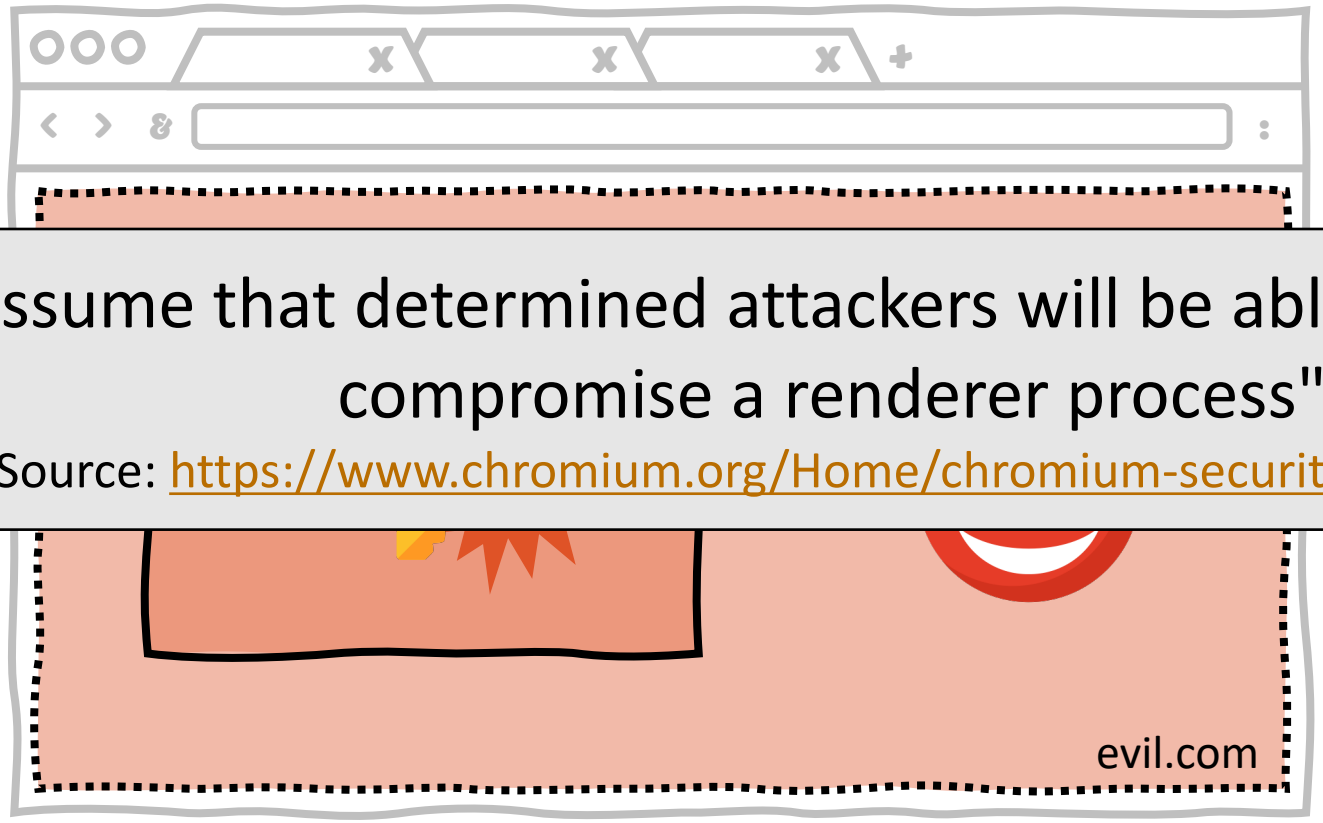


Threat Model

Threat Model: Attacker_{RW}



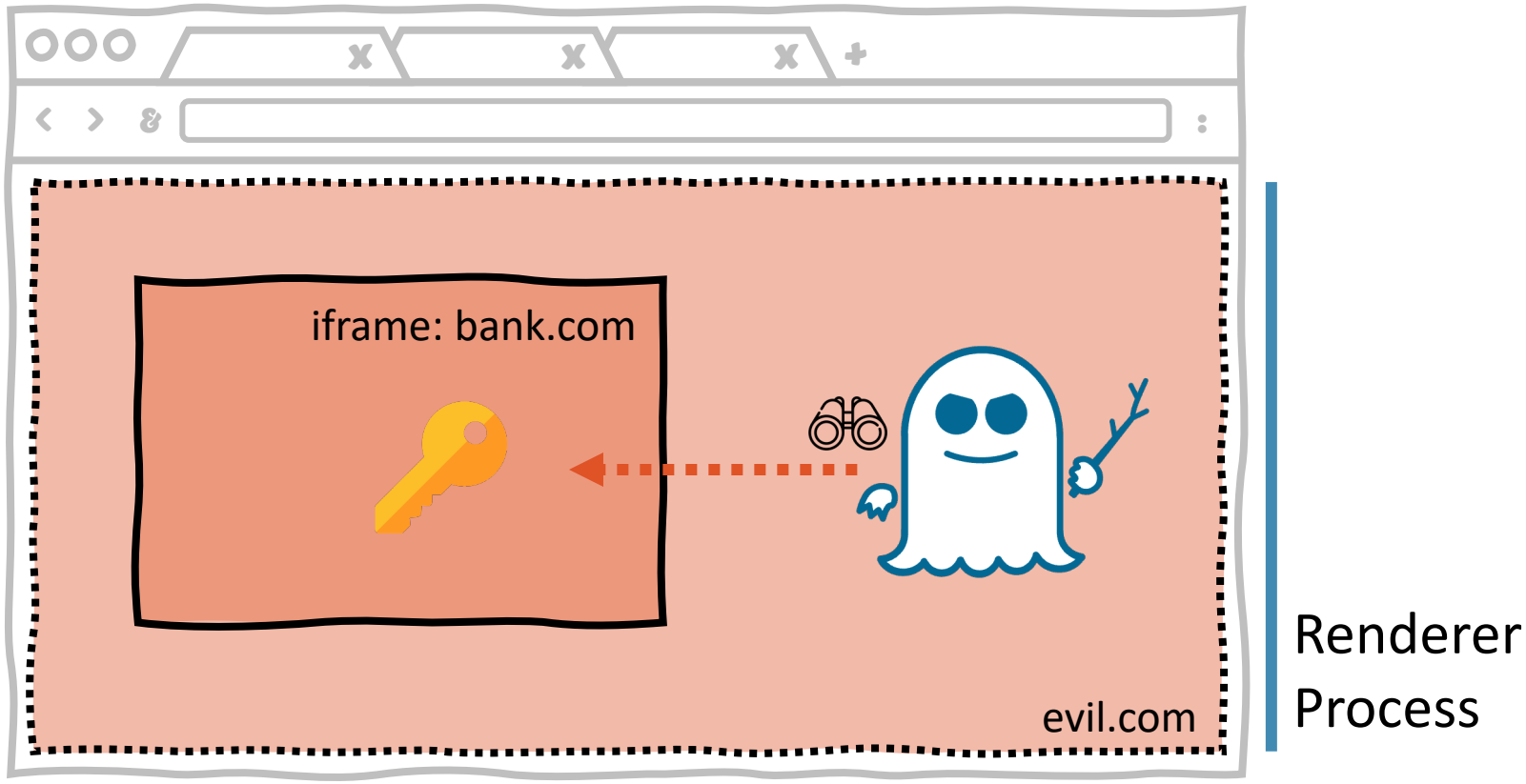
Threat Model: Attacker_{RW}



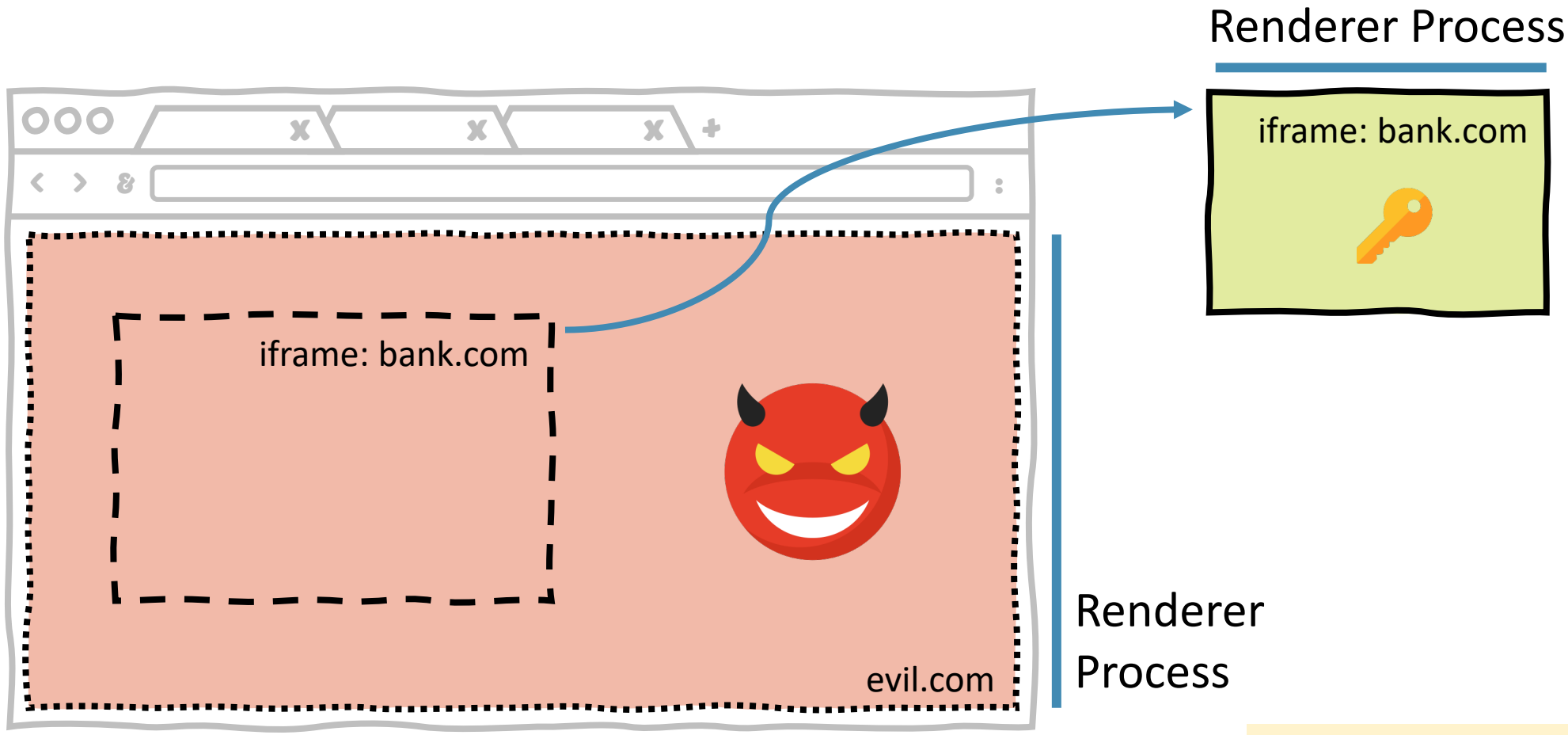
"We assume that determined attackers will be able to find a way to compromise a renderer process"

(Source: https://www.chromium.org/Home/chromium-security/site-isolation/)

Threat Model: Attacker$_R$



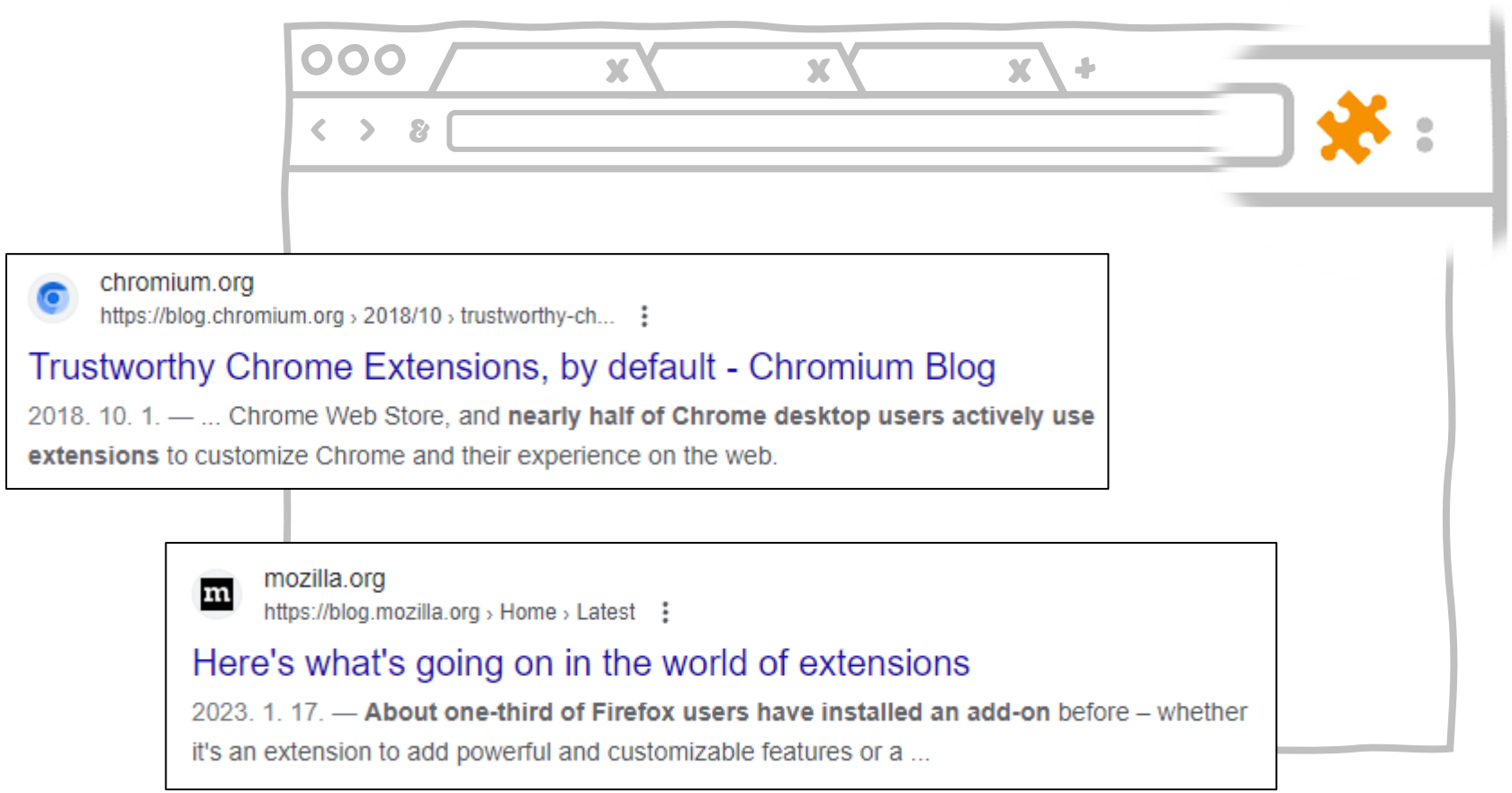
Site Isolation



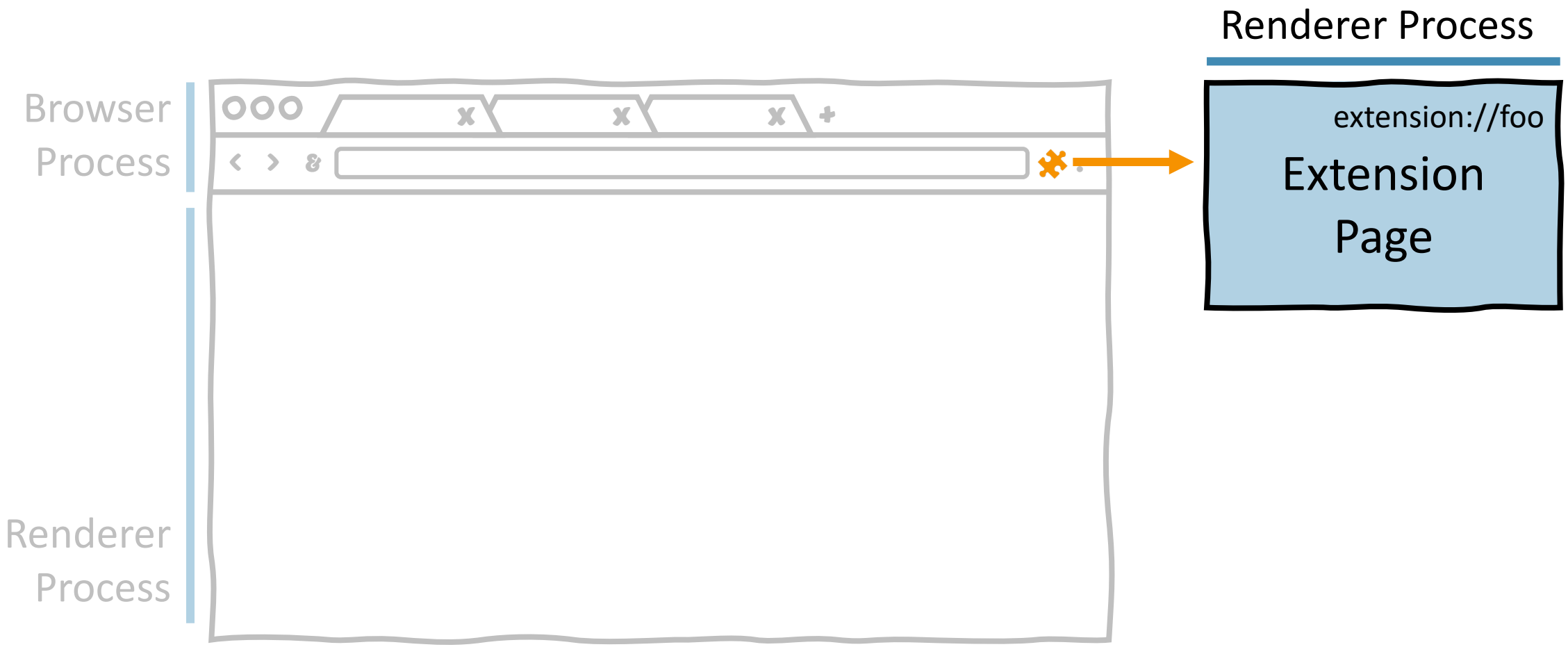
Reis et al. (Security '19)

What needs to be done in browser extensions to align with the site isolation?

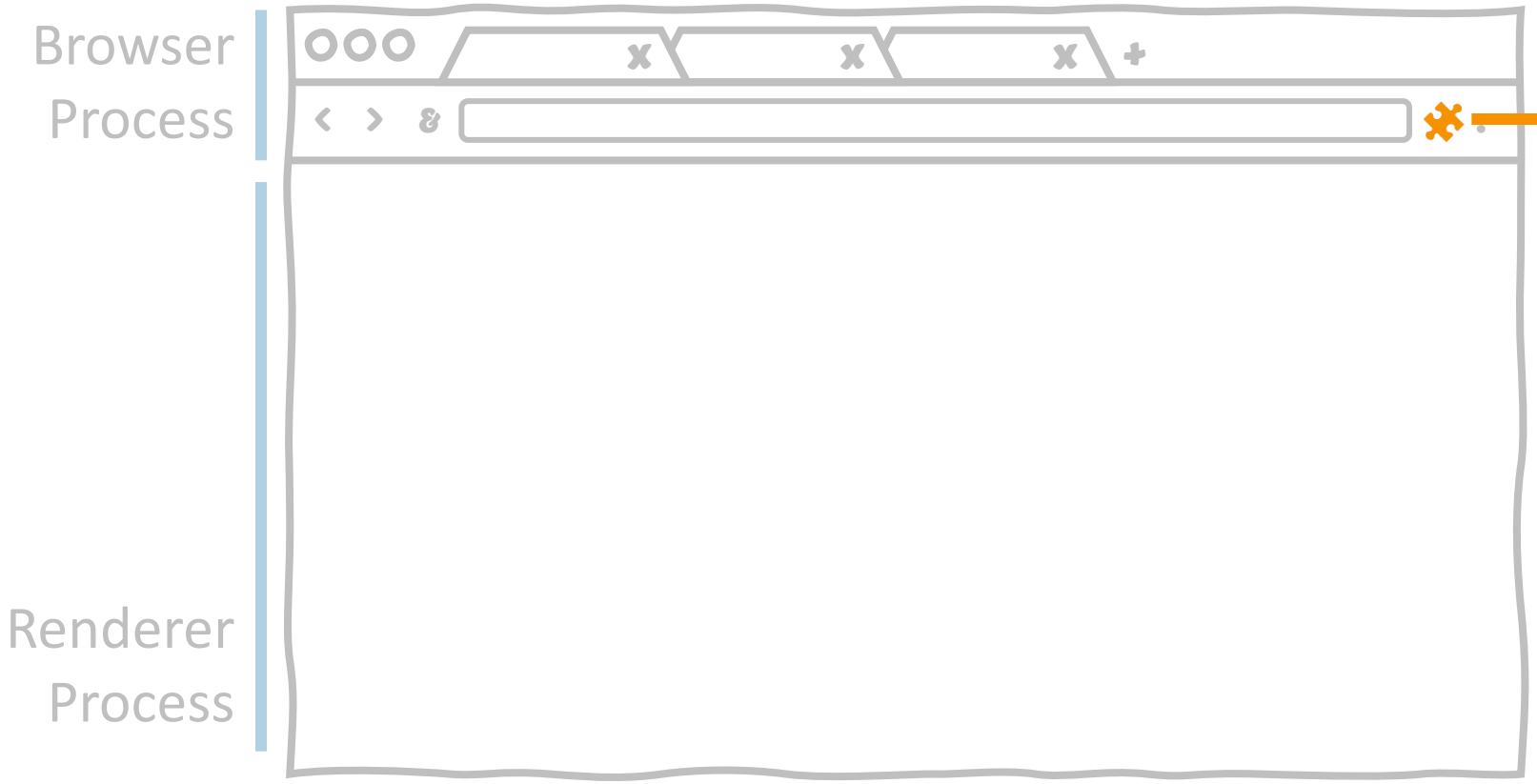
Browser Extension



Browser Extension Architecture



Browser Extension Architecture



Renderer Process

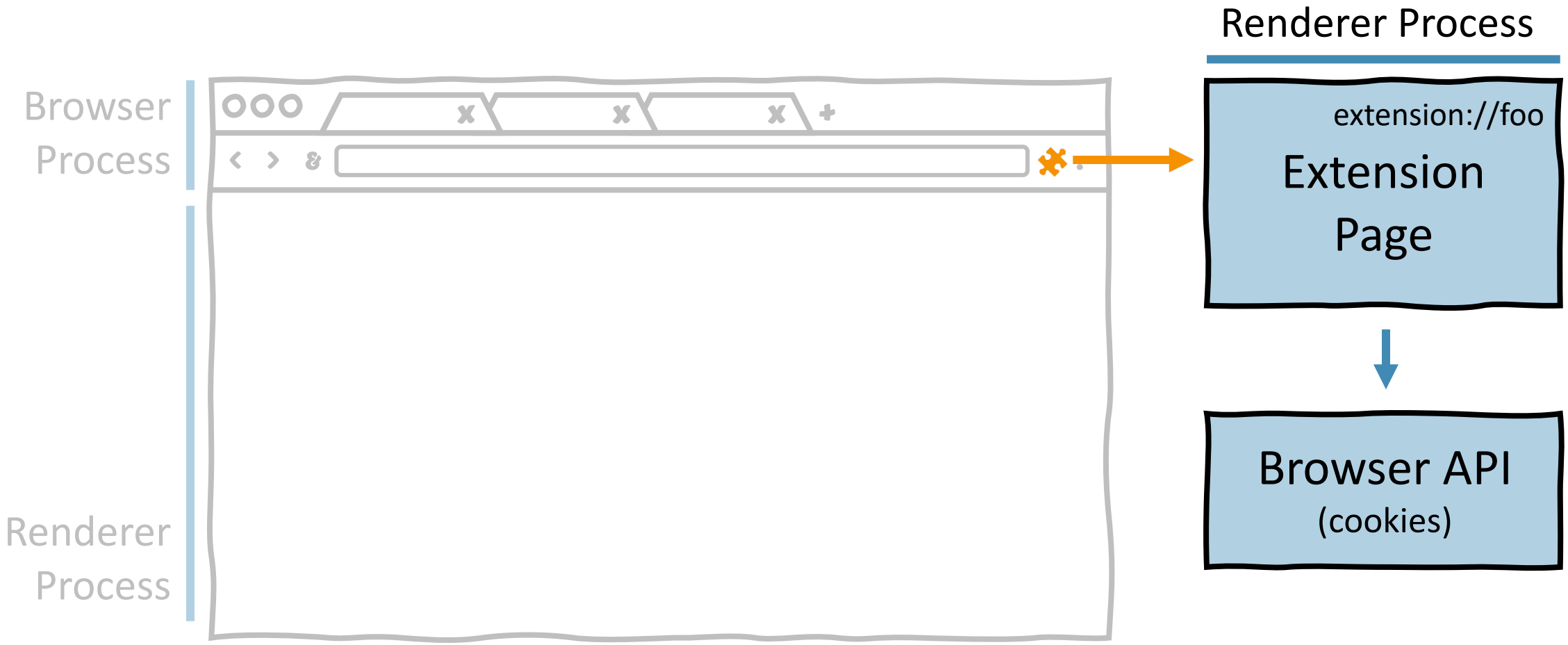
extension://foo

Extension Page

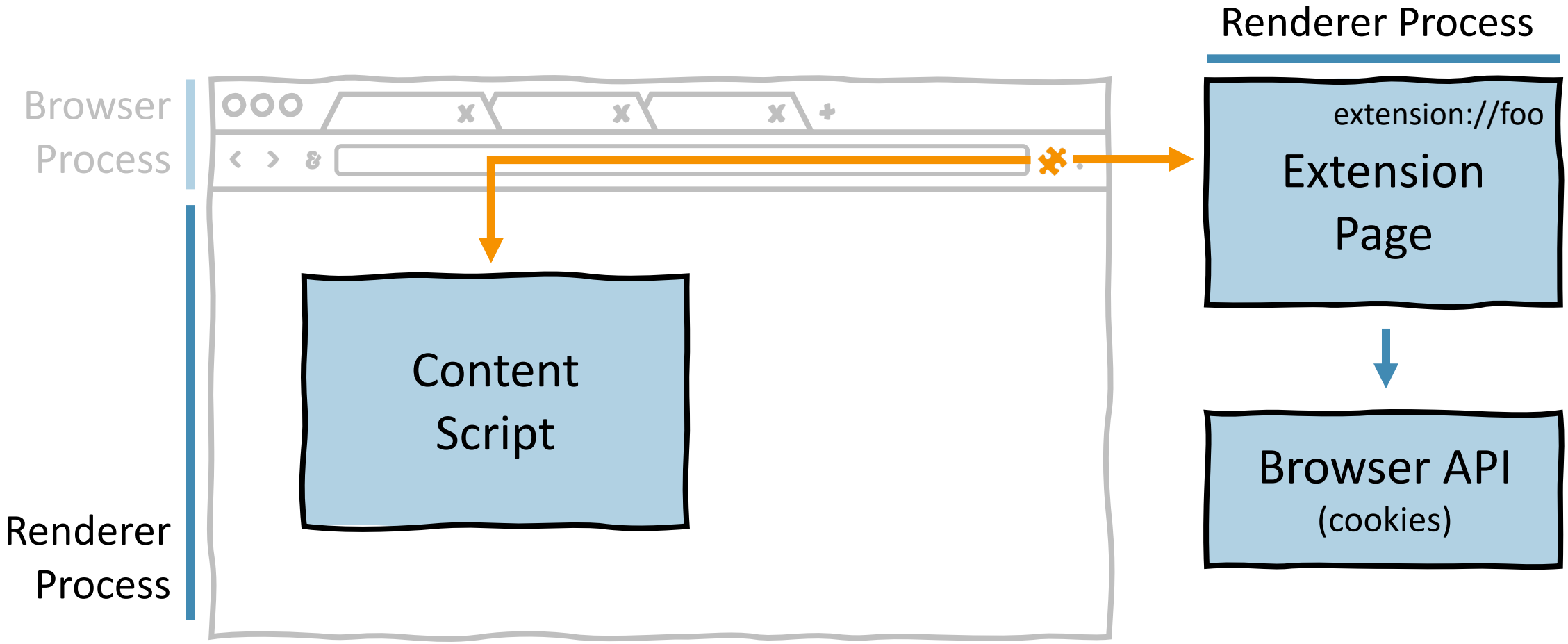
manifest.json

```
{
  "permissions": [
    "cookies",
    "<all_urls>"
  ]
}
```

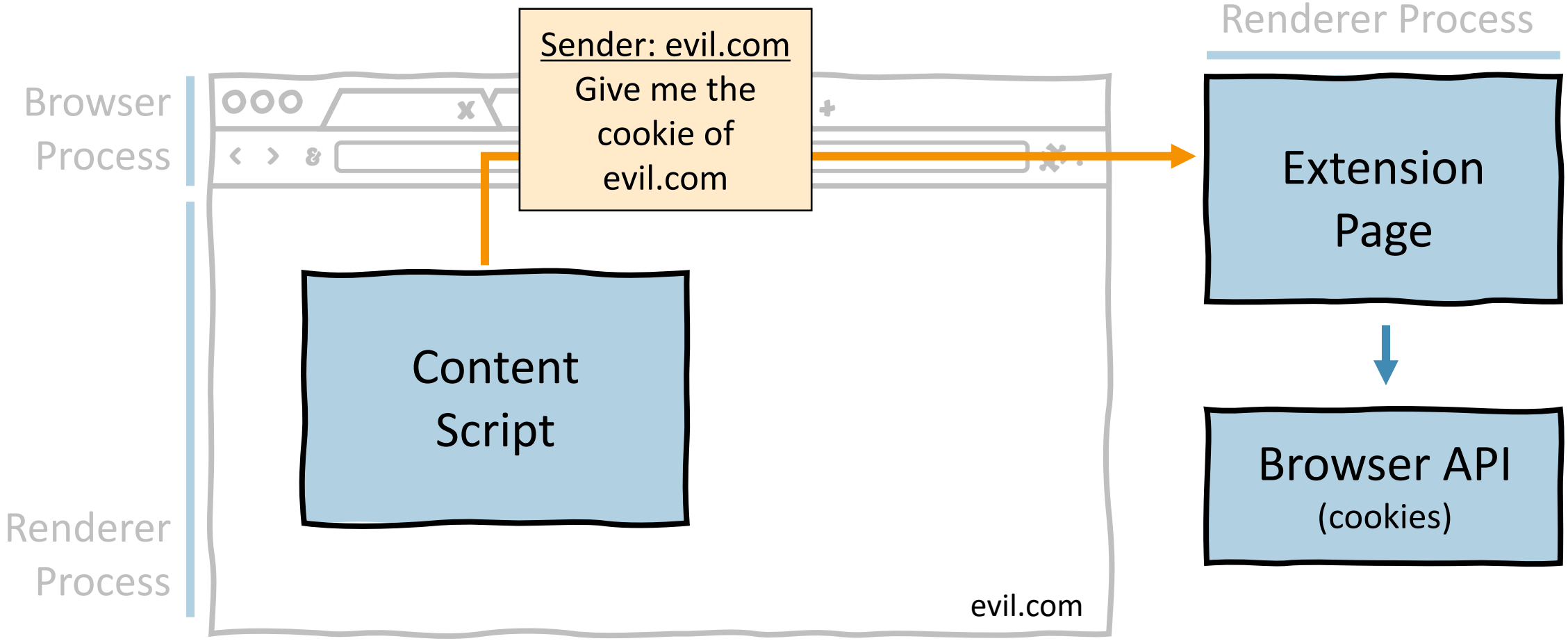
Browser Extension Architecture



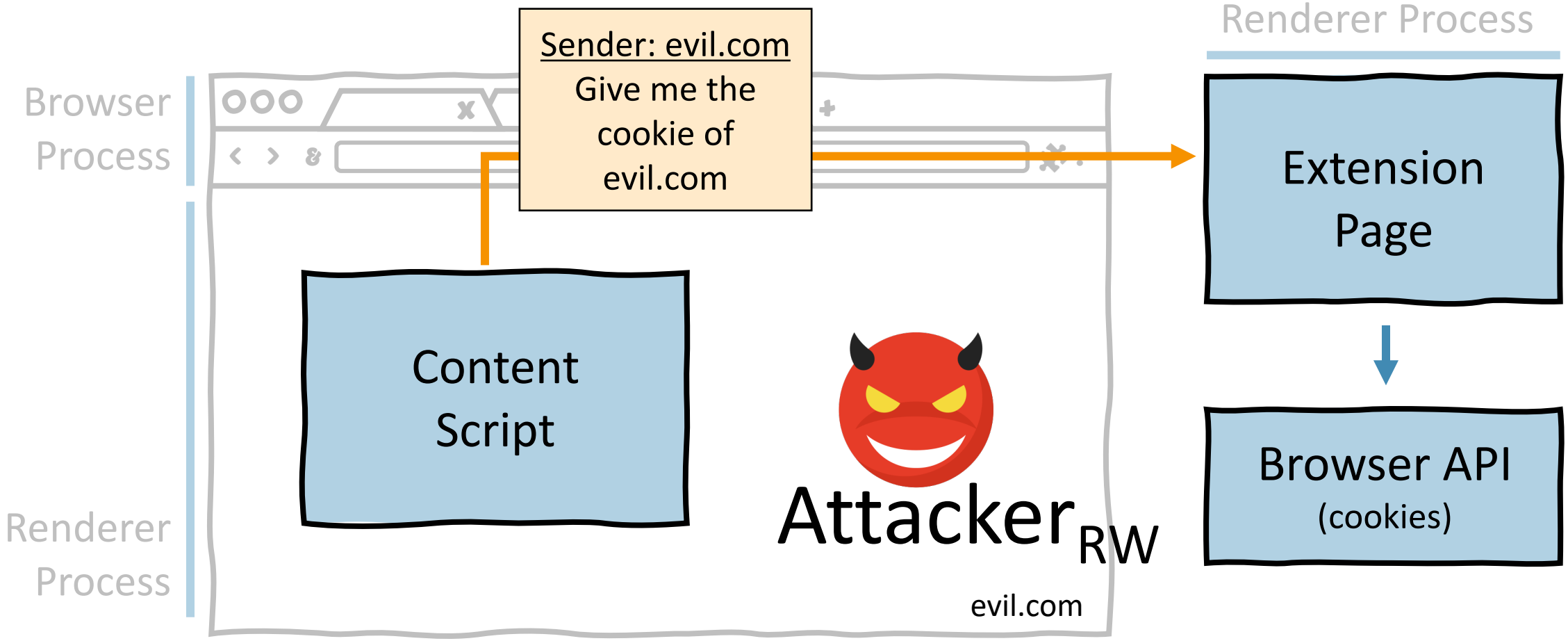
Browser Extension Architecture



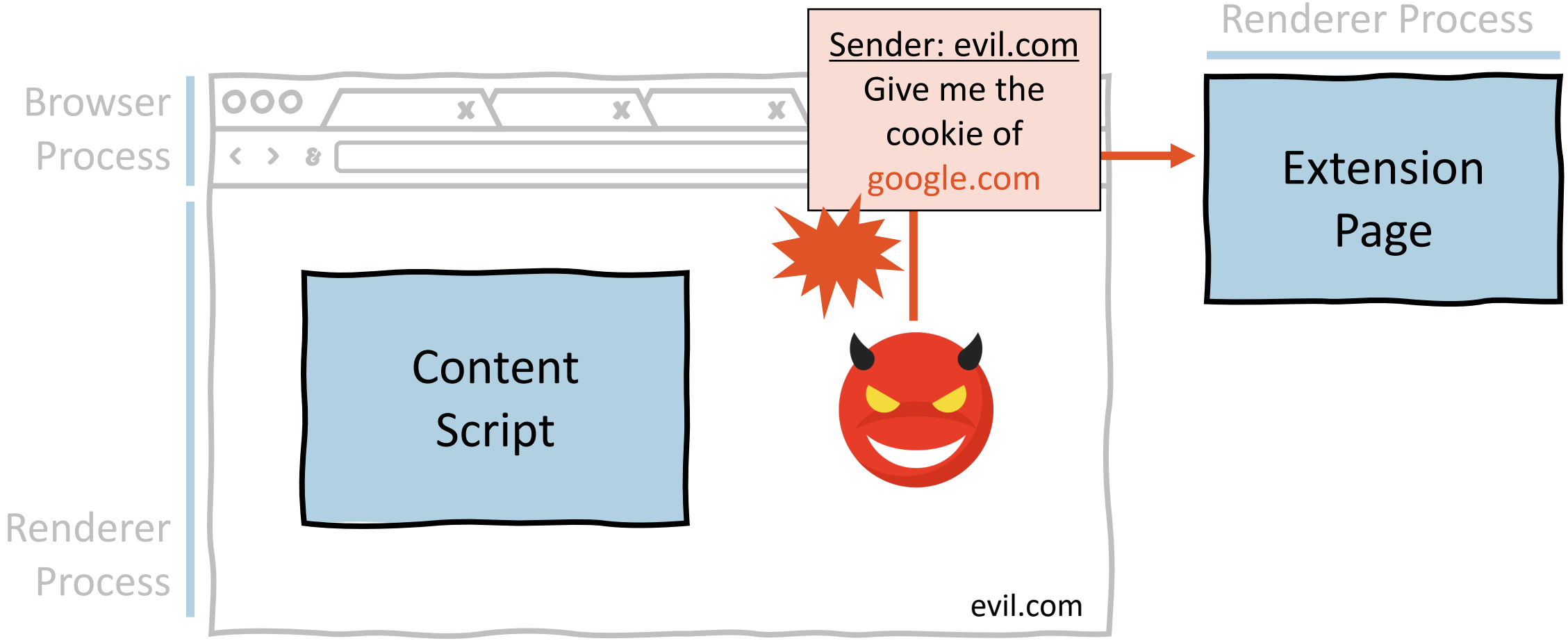
① Extension Message Passing



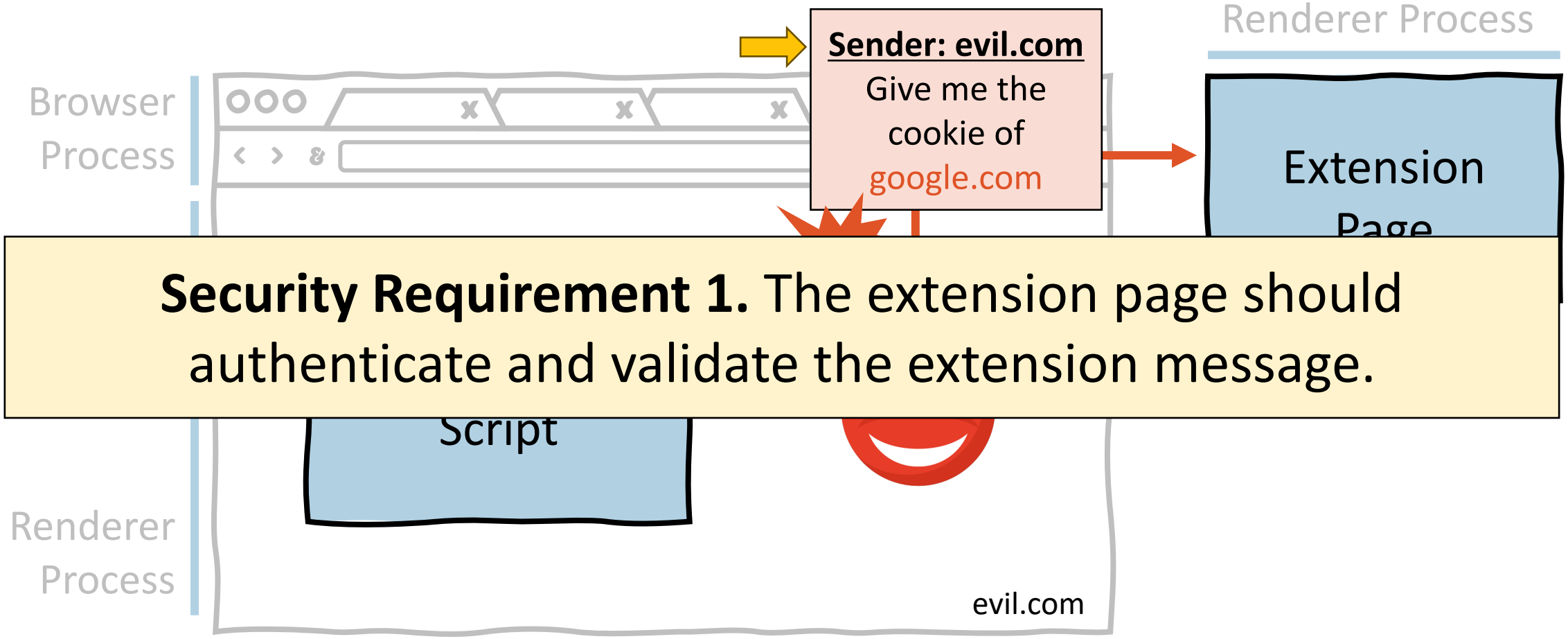
① Extension Message Passing



① Extension Message Passing



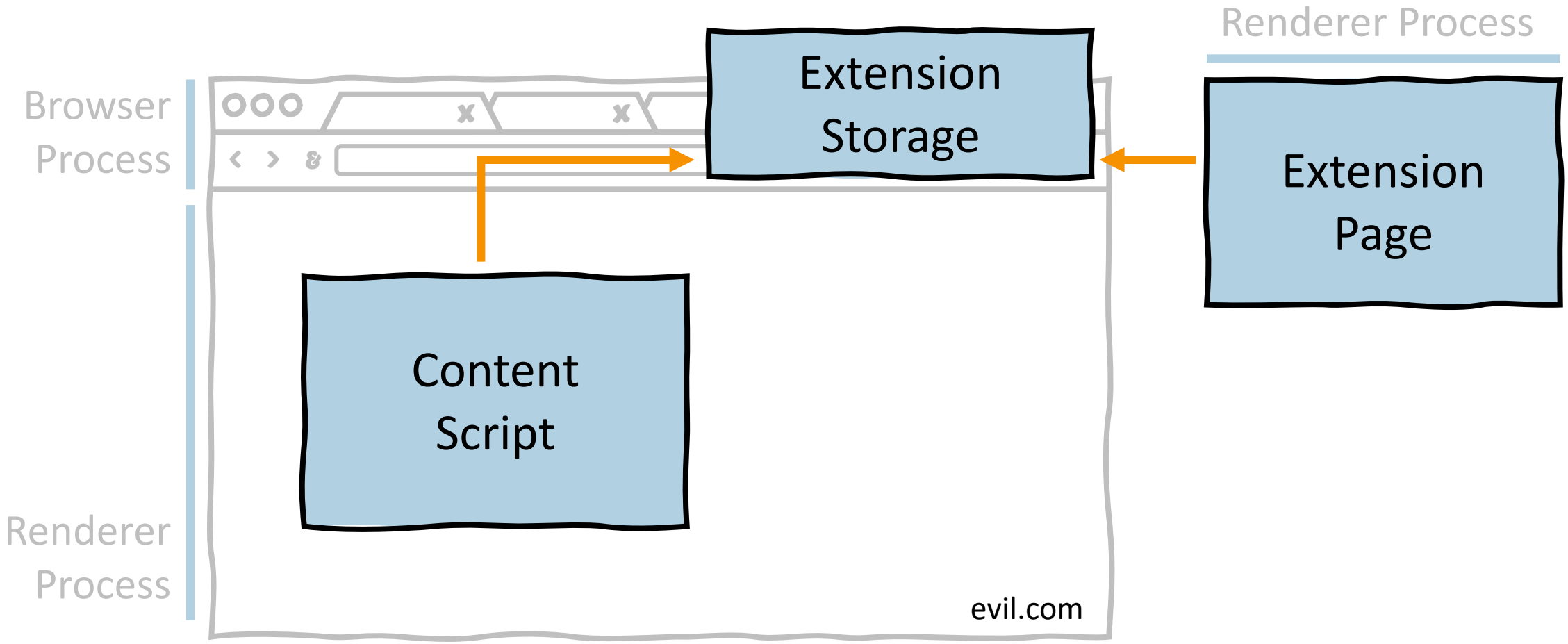
① Extension Message Passing



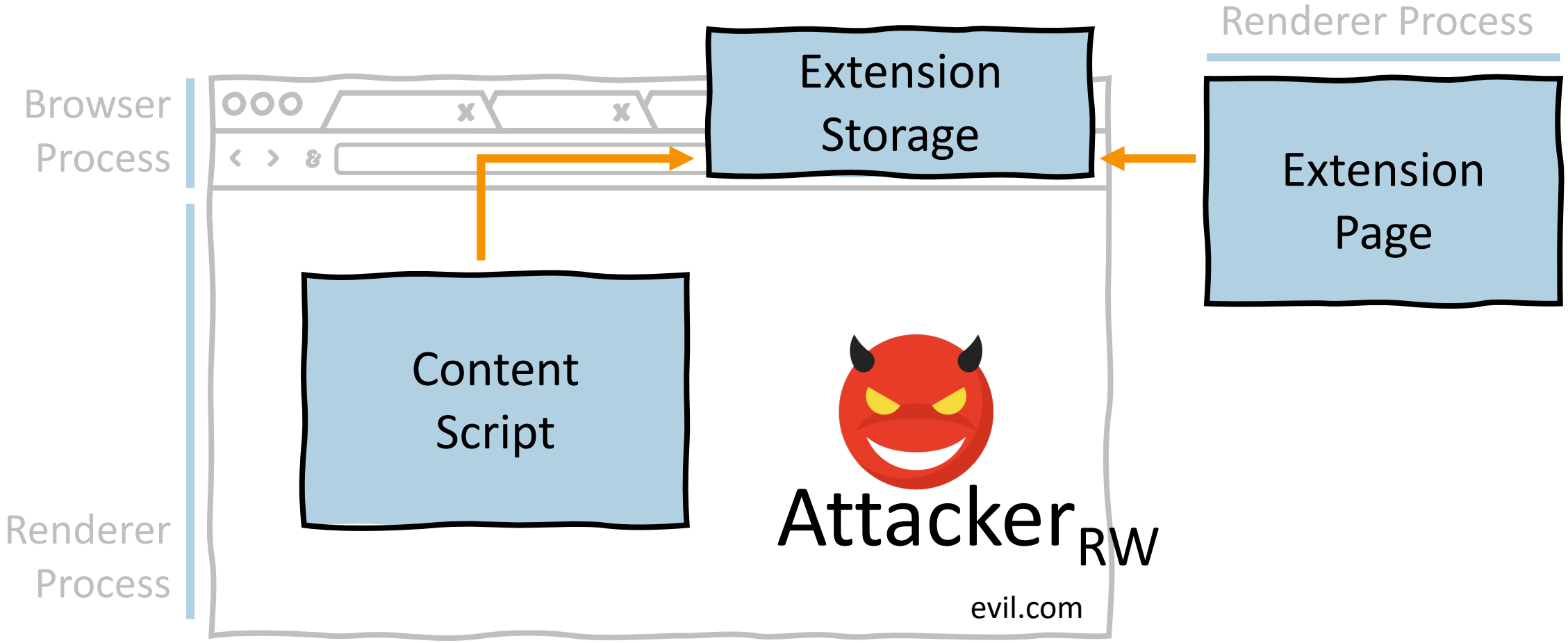
Browser Issues & Vulnerabilities I

- [~~Chrome~~/Firefox/~~Safari~~] Compromised renderer can spoof `MessageSender.id` of an extension that has not injected a content script
- [~~Chrome~~/Firefox/~~Safari~~] Compromised renderer can spoof `MessageSender.url`
- [Firefox] `MessageSender.origin` is not available
- Sender information may be incorrect or unavailable
- Compromised renderer can send message to disallowed targets
- No API or documentation on how to verify the message

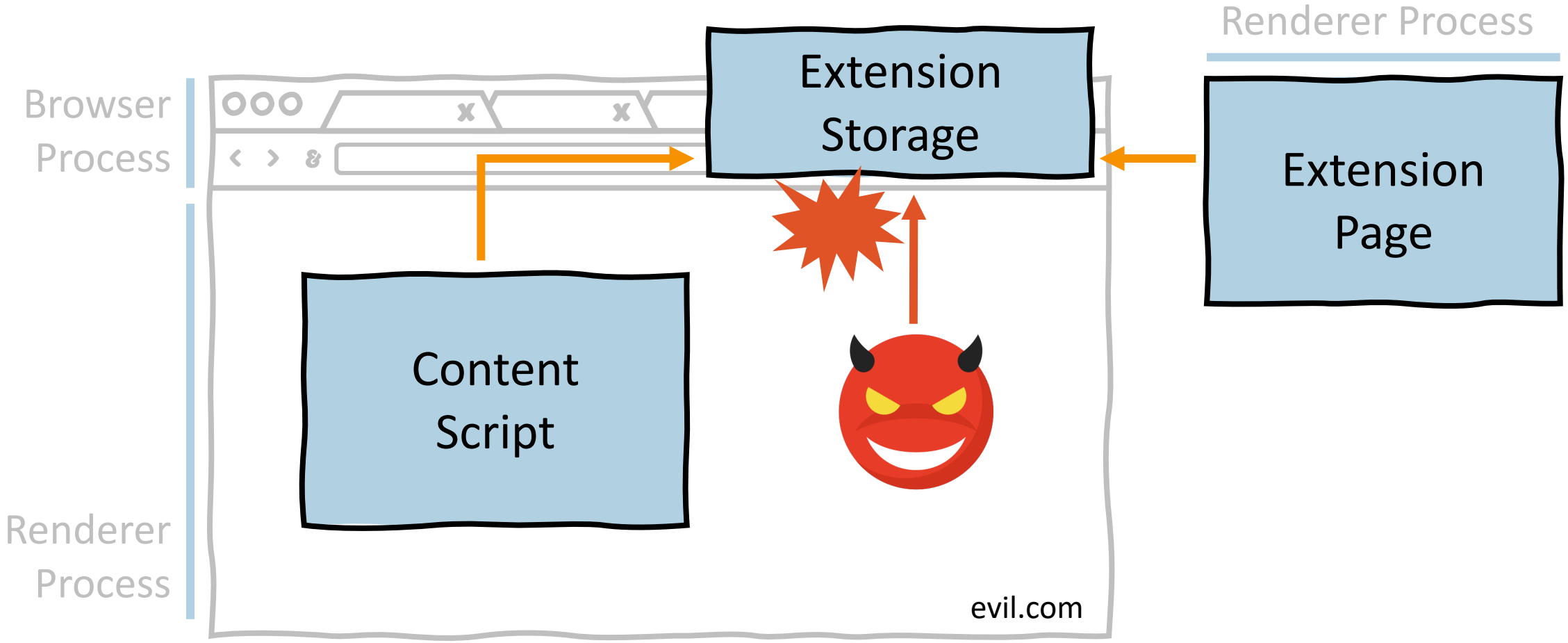
② Extension Storage



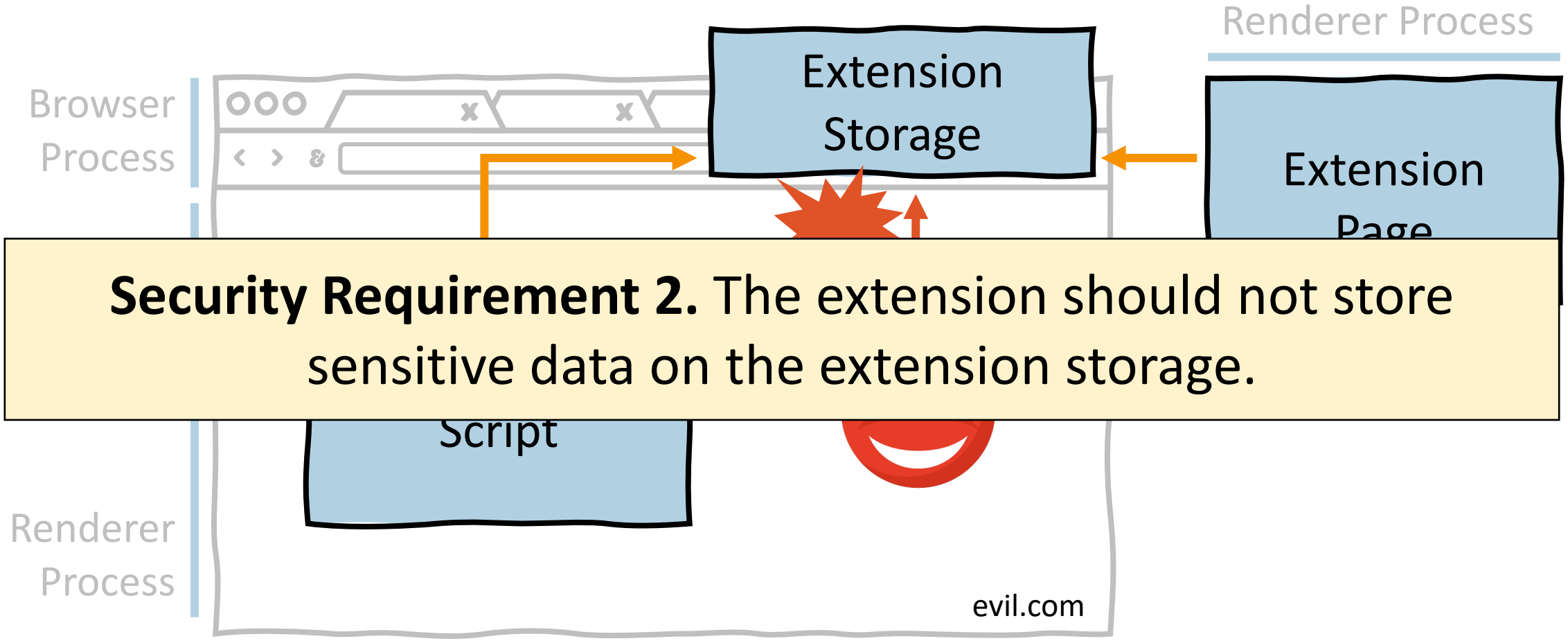
② Extension Storage



② Extension Storage



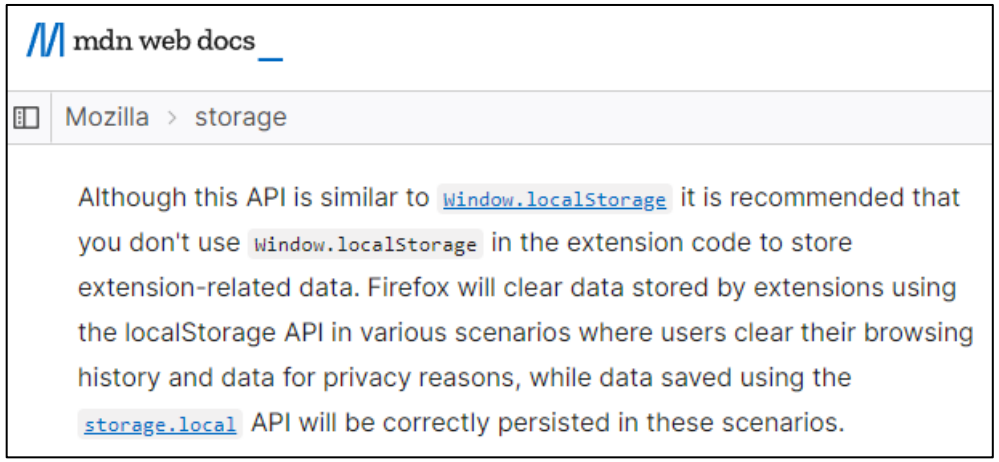
② Extension Storage

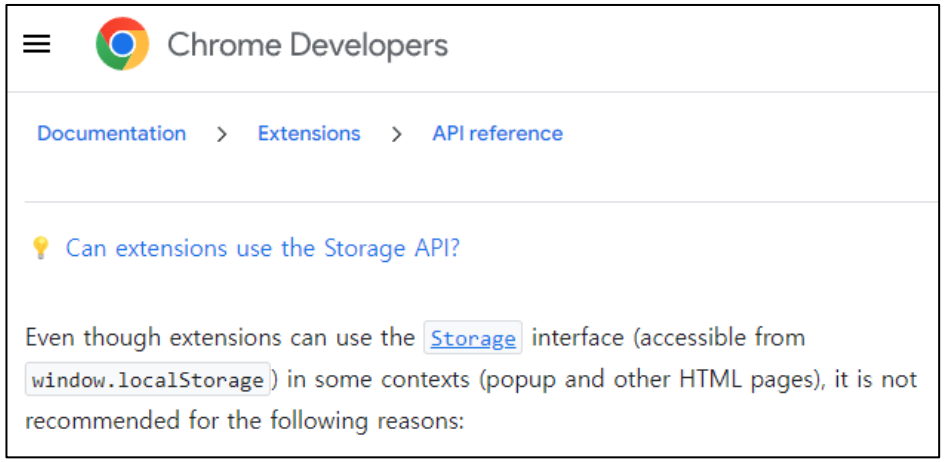


Security Requirement 2. The extension should not store sensitive data on the extension storage.

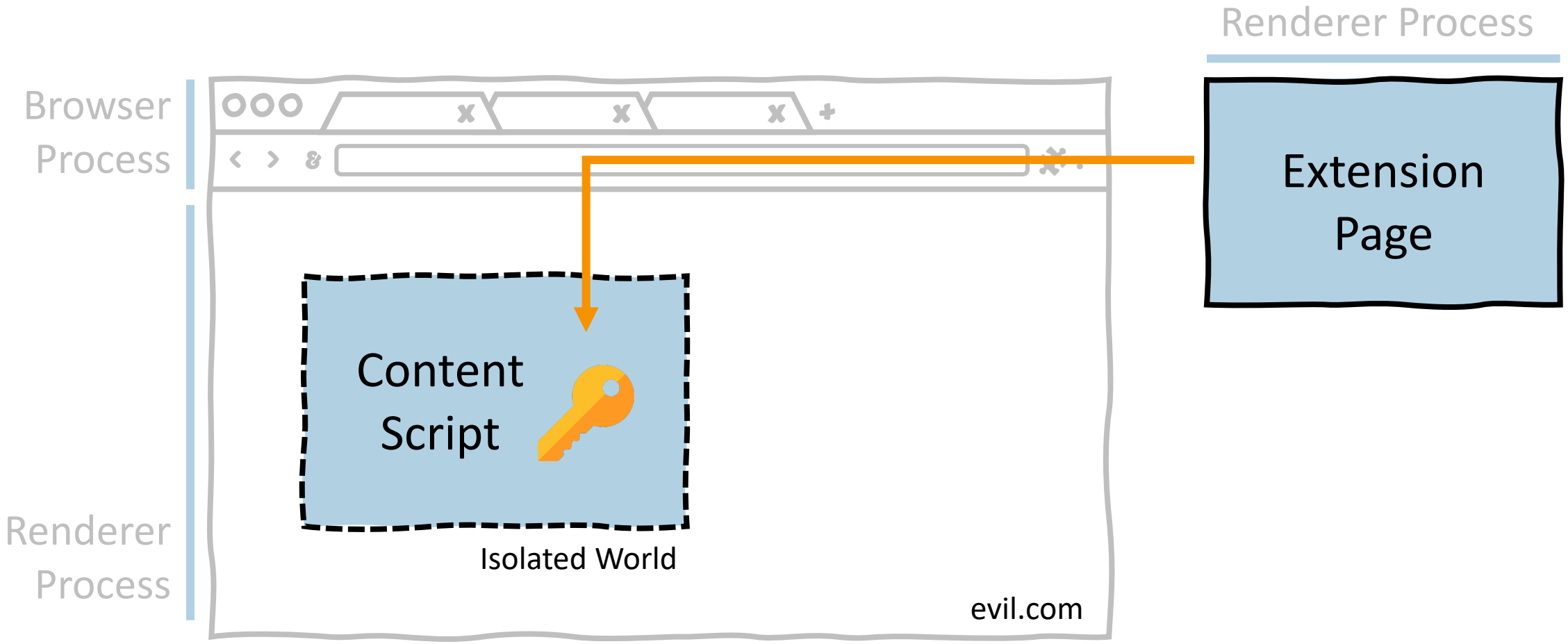
Browser Issues & Vulnerabilities II

- [~~Chrome~~/Firefox/~~Safari~~] Compromised renderer can access the storage of an extension that has not injected a content script
- Origin-specific (non-shared) storages are discouraged in the documentation:

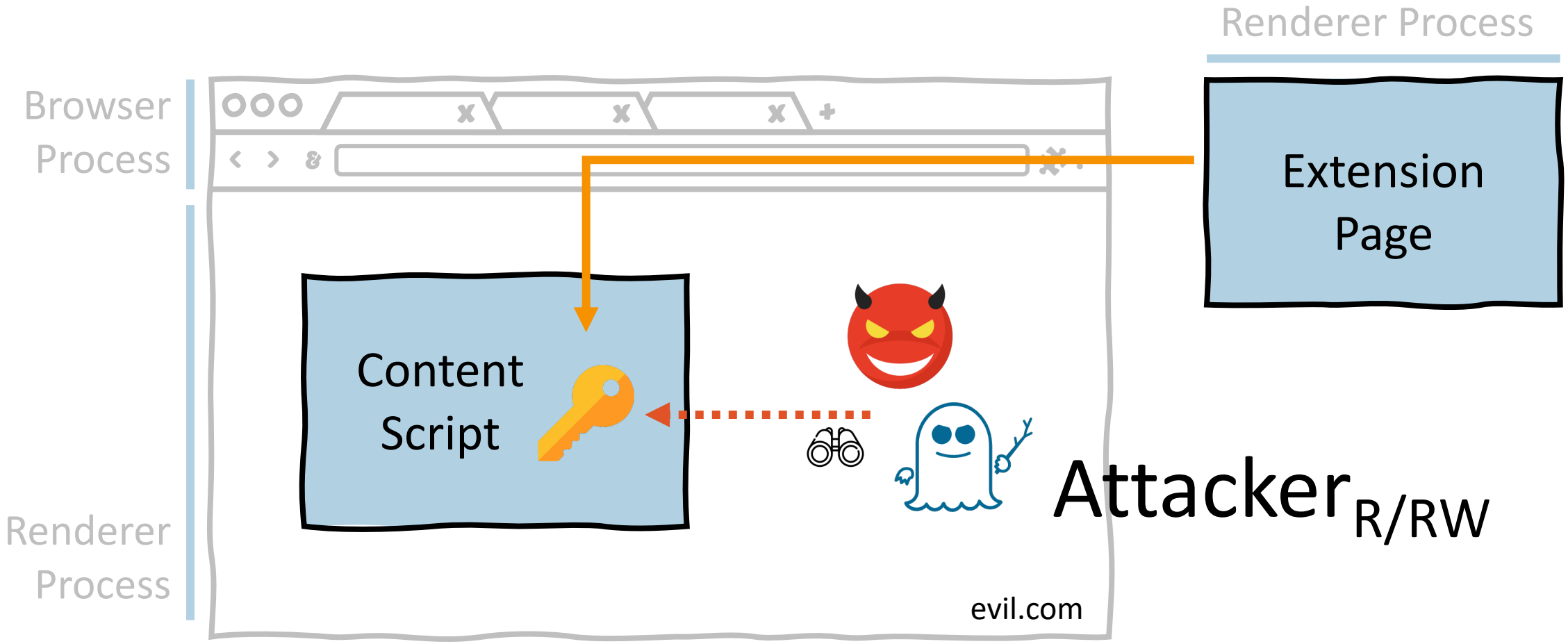




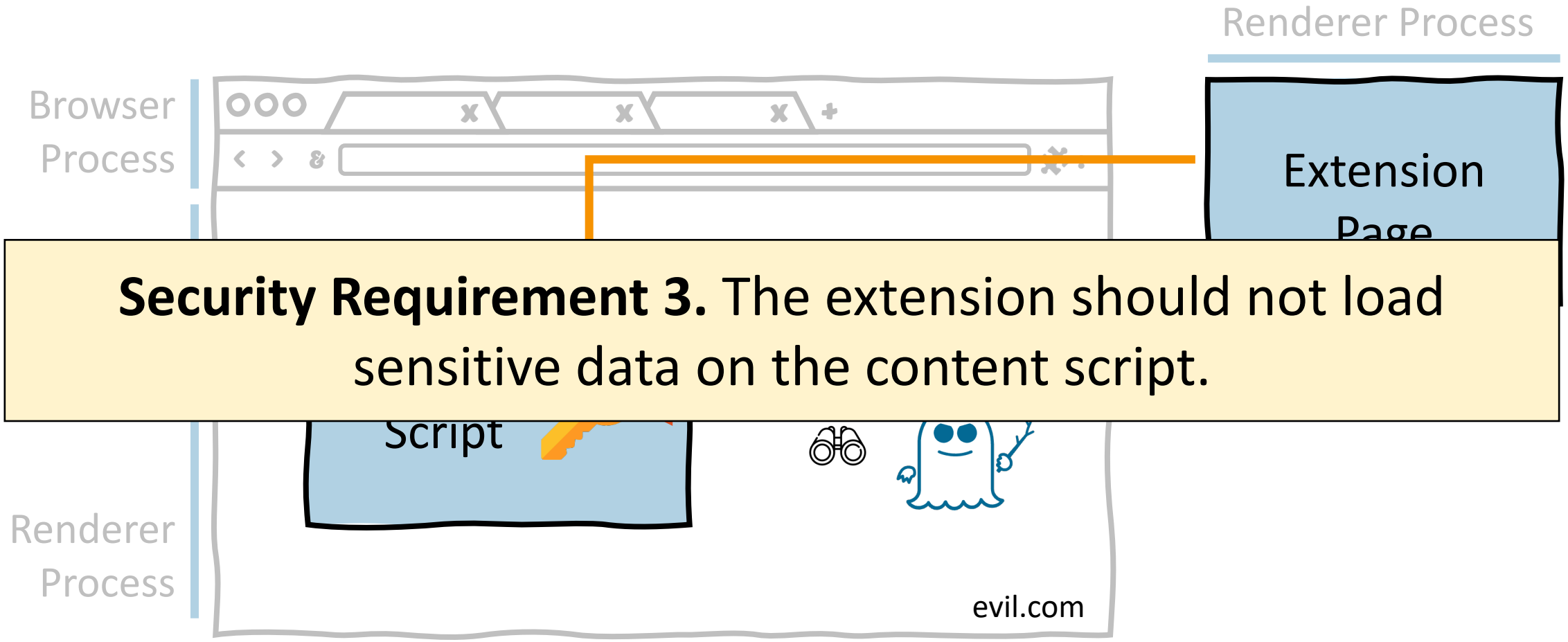
③ Isolated World



③ Isolated World



③ Isolated World



Security Requirement 3. The extension should not load sensitive data on the content script.

How well do extension developers meet these requirements?

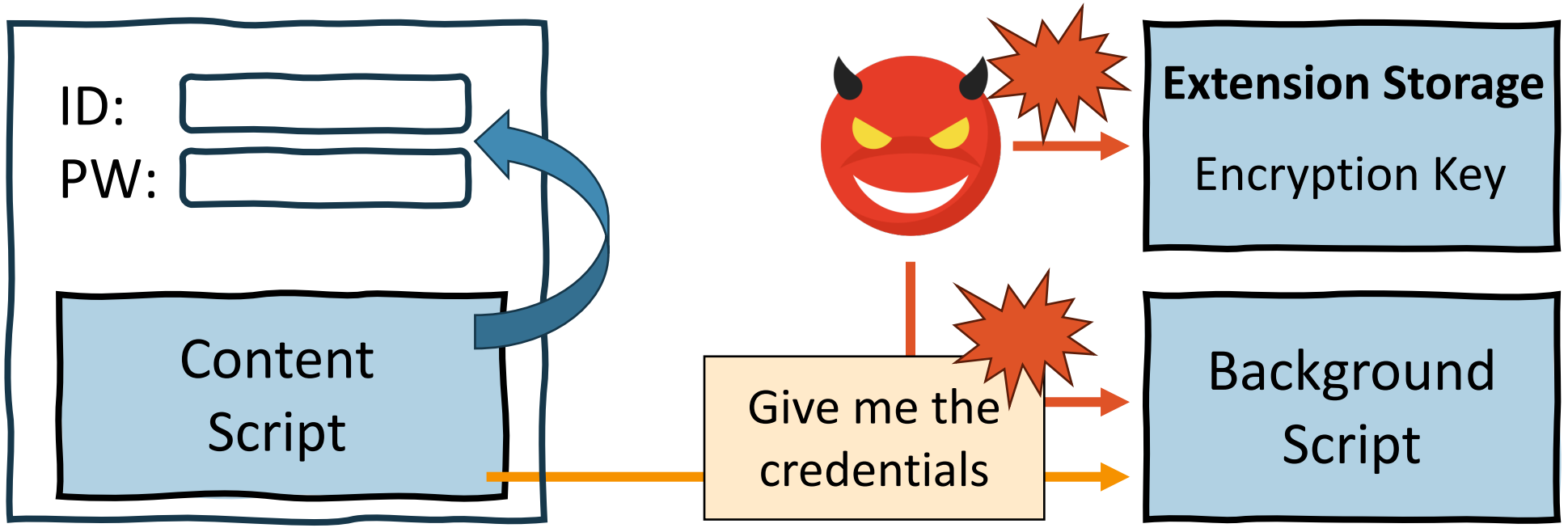
Privilege Escalation Attacks

- Failure to authenticate the extension **message**
  - **Read** sensitive data
  - **Modify** senstive data
  - **Execute** privileged APIs
- Sensitive data in the extension **storage**
  - **Read** sensitive data
  - **Modify** sensitive data
- Sensitive data in the **content script**
  - **Read** sensitive data

Privilege Escalation Attacks

- Failure to authenticate the extension **message**
  - **Read** sensitive data
  - **Modify** senstive data
  - **Execute** privileged APIs
- Sensitive data in the extension **storage**
  - **Read** sensitive data
  - **Modify** sensitive data
- Sensitive data in the **content script**
  - **Read** sensitive data

Password Managers



Privilege Escalation Attacks

- Failure to authenticate the extension **message**
  - **Read** sensitive data
  - **Modify** senstive data
  - **Execute** privileged APIs
- Sensitive data in the extension **storage**
  - **Read** sensitive data
  - **Modify** sensitive data
- Sensitive data in the **content script**
  - **Read** sensitive data

Ad Blockers, Userscript Managers

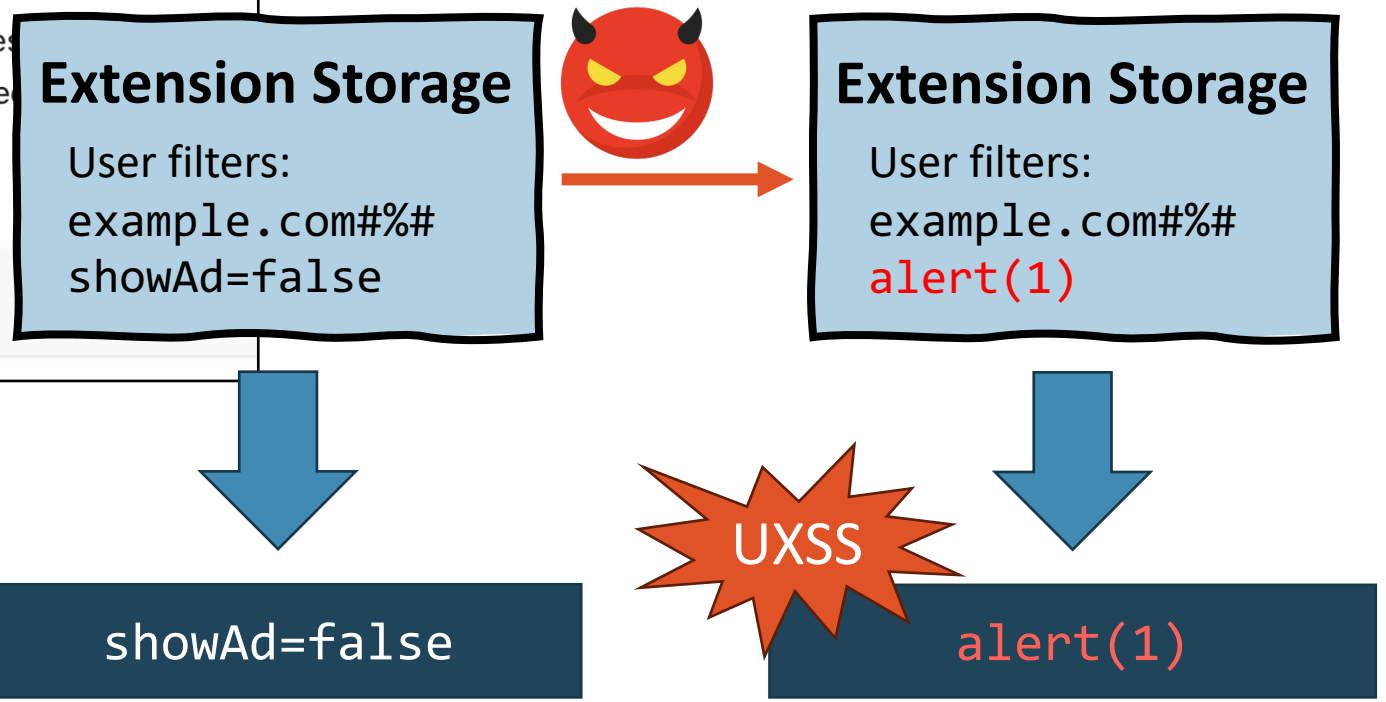
JavaScript rules

AdGuard supports a special type of rules that allows you to inject any javascript code

We **strongly recommend** using scriptlets instead of JavaScript rules
debugging, but as a long-time solution a scriptlet rule should be use

Syntax

```
rule = [domains]  "#%#" script
```

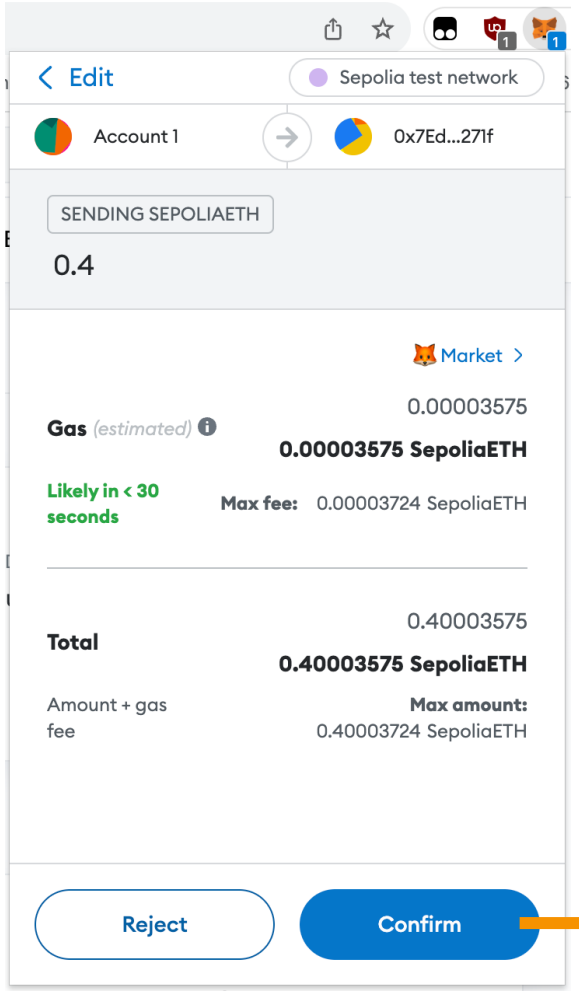


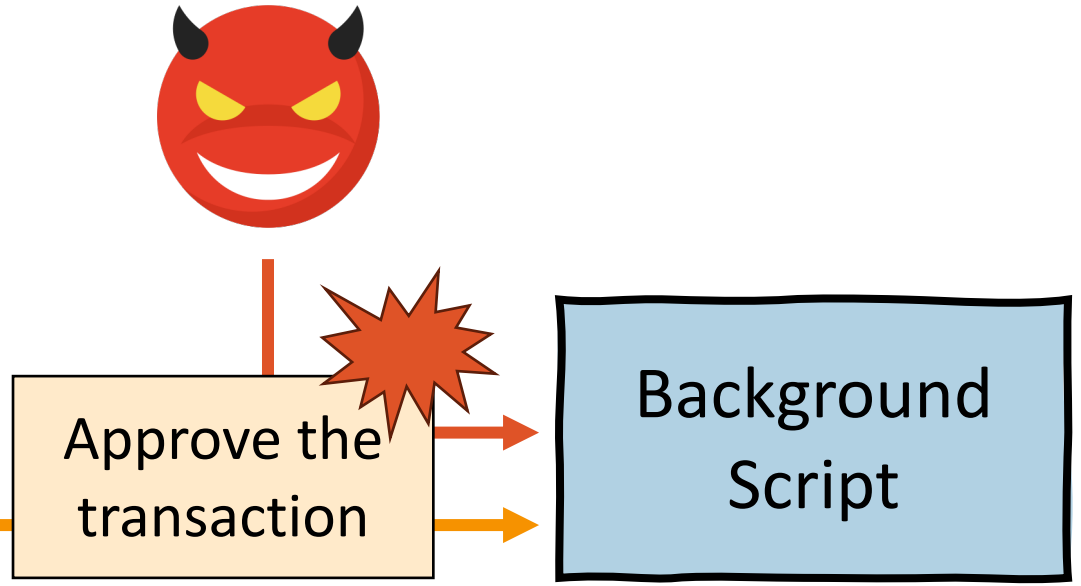
Privilege Escalation Attacks

- Failure to authenticate the extension **message**
  - **Read** sensitive data
  - **Modify** senstive data
  - **Execute** privileged APIs
- Sensitive data in the extension **storage**
  - **Read** sensitive data
  - **Modify** sensitive data
- Sensitive data in the **content script**
  - **Read** sensitive data

Cryptocurrency Wallets

Popup



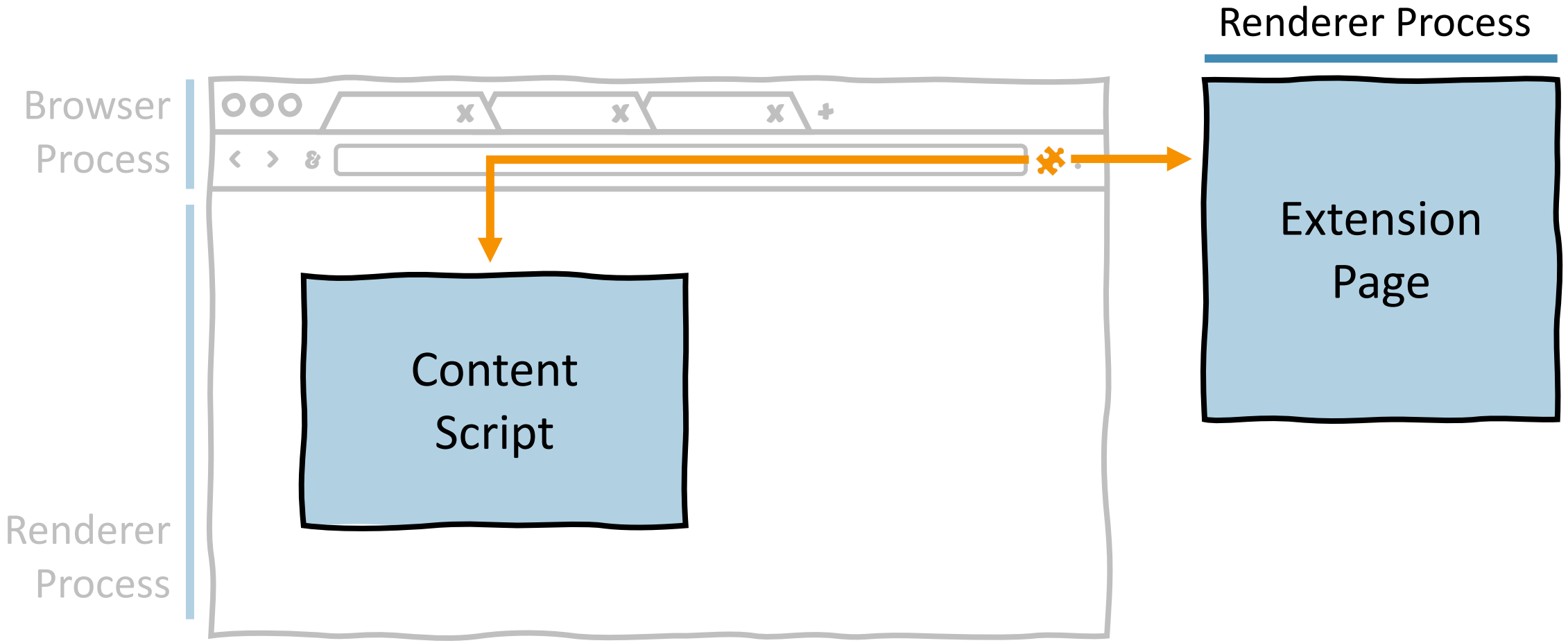


Vulnerabilities found in…

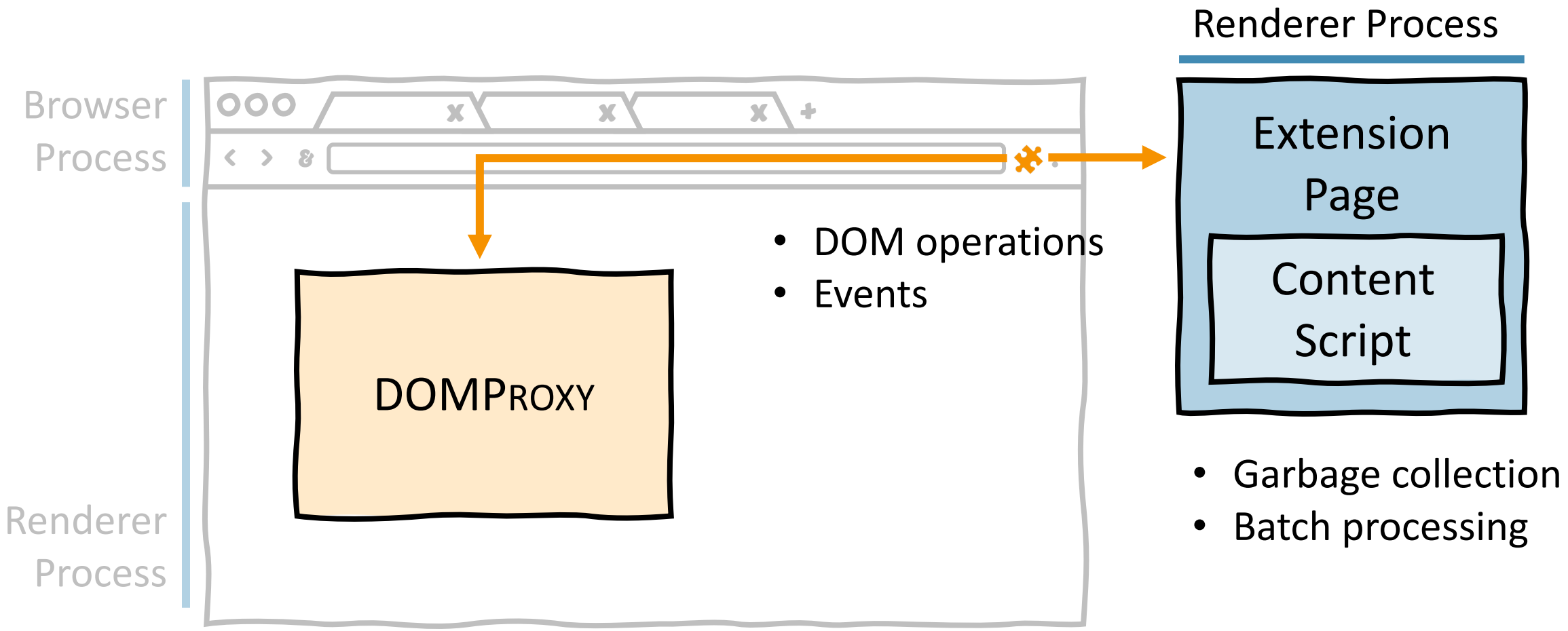
- Google Translate
- AdGuard AdBlocker
- Honey
- Adblock Plus
- uBlock Origin
- Adblock for YouTube
- Tampermonkey
- Kami
- Adobe Acrobat
- AdBlock
- LastPass
- Read&Write
- Remote Desktop
- ClassLink OneClick
- Cisco Webex
- Grammarly
- Screencastify
- MetaMask
- Clever
- Teleparty
- Windows Account

Problem: Security requirements imposed on extension developers

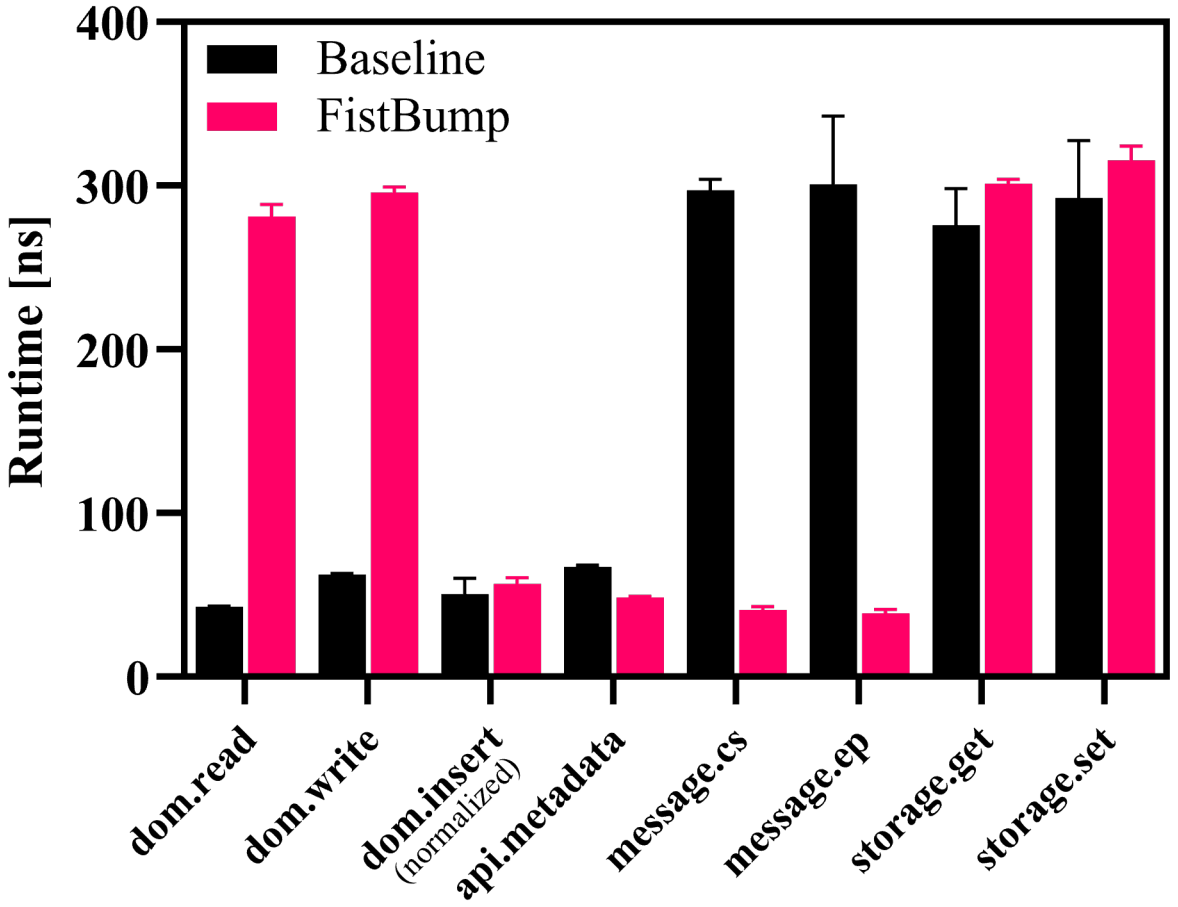
Our Solution: FISTBUMP



Our Solution: FISTBUMP



Evaluation


400
300
200
100
0
Runtime [ns]
Baseline
FistBump
dom.read
dom.write
dom.insert
(normalized)
api.metadata
message.cs
message.ep
storage.get
storage.set

Summary

- Current extension architecture imposes difficult **security requirements** on extension developers
- These lead to **privilege escalation attacks**, rendering the site isolation useless
- We propose FISTBUMP, a new extension architecture that **moves the content script to the extension process**


ym.kim@snu.ac.kr