Extending a Hand to Attackers

Browser Privilege Escalation Attacks via Extensions

USENIX Security 2023

Young Min Kim, Byoungyoung Lee
TL;DR: What’s Possible?

Visit

UXSS (universal cross-site scripting)

Steal Coins

Steal Passwords
Multi-Process Architecture

- **Browser Process**
  - Sandbox
  - evil.com

- **Renderer Process**
Multi-Process Architecture

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions
Threat Model

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions
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/](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

Trustworthy Chrome Extensions, by default - Chromium Blog
2018. 10. 1. — ... Chrome Web Store, and nearly half of Chrome desktop users actively use extensions to customize Chrome and their experience on the web.

Here's what's going on in the world of extensions
2023. 1. 17. — About one-third of Firefox users have installed an add-on before — whether it's an extension to add powerful and customizable features or a ...
Browser Extension Architecture

Browser Process

Renderer Process

Renderer Process

extension://foo

Extension Page
Browser Extension Architecture

Renderer Process

Extension Page

manifest.json

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

Browser Process

Renderer Process
Browser Extension Architecture

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions

Renderer Process

Extension Page

Browser API (cookies)
Browser Extension Architecture

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions

Browser Process

Renderer Process

Content Script

Extension Page

Renderer Process

extension://foo

Browser API (cookies)
1 Extension Message Passing

Sender: evil.com
Give me the cookie of evil.com

Content Script

Renderer Process

Extension Page

Browser API (cookies)

Sender: evil.com
Give me the cookie of evil.com

Renderer Process
Extension Message Passing

Sender: evil.com
Give me the cookie of evil.com

Content Script

Renderer Process

Extension Page

Browser API (cookies)

Attacker RW

Sender Message: Give me the cookie of evil.com

Attacker RW

evil.com
① Extension Message Passing

Sender: evil.com
Give me the cookie of google.com

Browser Process

Renderer Process

Content Script

Renderer Process

Extension Page

evil.com
1 Extension Message Passing

Security Requirement 1. The extension page should authenticate and validate the extension message.

Sender: evil.com
Give me the cookie of google.com
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

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions
② Extension Storage

Browser Process

Renderer Process

Renderer Process

Extension Storage

Extension Page

Content Script

Attacker

evil.com
② Extension Storage

Browser Process

Renderer Process

Extension Storage

Extension Page

Renderer Process

Content Script

evil.com

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions
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:

![mdn web docs](image)

Although this API is similar to `window.localStorage`, it is recommended that you don't use `window.localStorage` in the extension code to store extension-related data. Firefox will clear data stored by extensions using the `localStorage` API in various scenarios where users clear their browsing history and data for privacy reasons, while data saved using the `storage.local` API will be correctly persisted in these scenarios.

![Chrome Developers](image)

Even though extensions can use the `Storage` interface (accessible from `window.localStorage`) in some contexts (popup and other HTML pages), it is not recommended for the following reasons:
③ Isolated World
Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions

③ Isolated World

Browser Process

Renderer Process

Content Script

Renderer Process

Extension Page

Attacker_{R/RW}

evil.com
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 sensitive 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** sensitive 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

ID: [ ]
PW: [ ]

Content Script

Give me the credentials

Extension Storage
Encryption Key

Background Script
Privilege Escalation Attacks

• Failure to authenticate the extension message
  • Read sensitive data
  • *Modify* sensitive 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 for debugging, but as a long-time solution a scriptlet rule should be used.

Syntax

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

Extension Storage

User filters:
```
example.com#$
showAd=false
```

UXSS

```
showAd=false
```

Extension Storage

User filters:
```
example.com#$
alert(1)
```

```
alert(1)
```
Privilege Escalation Attacks

- Failure to authenticate the extension *message*
  - Read sensitive data
  - Modify sensitive 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

Approve the transaction

Background Script

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions
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

Extending a Hand to Attackers: Browser Privilege Escalation Attacks via Extensions

DOMPROXY

• DOM operations
• Events

Renderer Process

Extension Page

Content Script

• Garbage collection
• Batch processing
Evaluation

![Graph showing runtime comparison between Baseline and FistBump]
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