Modern web experience
Modern web experience
Modern web experience

Web apps

- NYTimes
- ...
- Chase

Extensions

- AdBlock
- Evernote

Core browser
Web app security

- **Trust model:** malicious code

- Apps are isolated according to same-origin policy

- Apps are constrained to Web APIs (e.g., DOM)
  - They cannot access arbitrary files, devices, etc.
Extension security?

• Extensions need direct access to app DOMs
  ➤ They modify app style, content, behavior, ...

• Extensions need privileged APIs
  ➤ To fetch/store cross-origin content, to read/modify history and bookmarks, to create new tabs, etc.
Chrome extension security model

- **Trust model**: extensions are benign-but-buggy
  - Protects vulnerable extension from malicious apps

- Privilege separate extension: core and content
  - Protects vulnerable extension from malicious apps

- Run extensions with least privilege
  - Limits damage due to exploits
Least privilege via permission system

• Extensions declare necessary permissions

```json
{
    "name": "AdBlock Plus",
    "version": "2.1.10",
    ...
    "permissions": [
        "http://*/*", "https://*/*", "contextMenus"
    ],
    ...
}
```

• Users must grant permissions at install time
What does mean?

• Can read and modify data on any site, regardless of what site you are visiting

• AdBlock must be a special case, right?

 ➤ 71.6% of top 500 extensions need this privilege!
What does mean?

• Can read and modify data on any site, regardless of what site you are visiting

• AdBlock must be a special case, right?
  ➢ 71.6% of top 500 extensions need this privilege!
It gets worse with popularity

![Graph showing the fraction of users who can read and change top n extensions against the number of users (few days later). The graph illustrates the decline in access and changes as the popularity of the extensions increases.]
It gets worse with popularity

% of n that can read and change all your data...

Top n extensions

Fraction that can read and change ...

Number of users (few days later)
It gets worse with popularity

The diagram shows the fraction of users who can read and change all your data, compared to the number of users (few days later). The x-axis represents the top n extensions, while the y-axis shows the fraction and number of users. The graph indicates that as the popularity of the extensions increases, the percentage of users who can access your data also increases significantly.
It gets worse with popularity

Removed from Chrome Web Store

Fraction that can read and change... % of n that can read and change all your data...

Top n extensions

# of users

Number of users (few days later)
Problem with Chrome’s model

• Permission requests are meaningless
  ➤ Descriptions are broad and context-independent

• Model encourages principle of most privilege
  ➤ Extensions don’t auto-update if they need more privs

• Threat model is not realistic
  ➤ Chrome Web Store listed many malicious extensions
  ➤ Roughly 5% of Google users run malicious extensions
Problem with Chrome’s model

• Permission requests are meaningless
  ➤ Descriptions are broad and context-independent

• Model encourages principle of most privilege
  ➤ Extensions don’t auto-update if they need more privs

• Threat model is not realistic
  ➤ Chrome Web Store listed many malicious extensions
  ➤ Roughly 5% of Google users run malicious extensions
Problem with Chrome’s model

- Permission requests are meaningless
  - Descriptions are broad and context-independent
- Model encourages principle of most privilege
  - Extensions don’t auto-update if they need more privs
- Threat model is not realistic
  - Chrome Web Store listed many malicious extensions
  - Roughly 5% of Google users run malicious extensions
New extension-system goals

• Meaningful permission system
  ➤ Safe behavior should not require permission
  ➤ Permissions requests should be content-specific

• Model should encourage least privilege
  ➤ Permissions should be fine-grained
  ➤ Incentivize safe extensions

• Threat model: extensions may be malicious
  ➤ Need to also protect user app data from extensions
New extension-system goals

• Meaningful permission system
  ➢ Safe behavior should not require permission
  ➢ Permissions requests should be content-specific

• Model should encourage least privilege
  ➢ Permissions should be fine-grained
  ➢ Incentivize safe extensions

• Threat model: extensions may be malicious
  ➢ Need to also protect user app data from extensions
New extension-system goals

• Meaningful permission system
  ➤ Safe behavior should not require permission
  ➤ Permissions requests should be content-specific

• Model should encourage least privilege
  ➤ Permissions should be fine-grained
  ➤ Incentivize safe extensions

• Threat model: extensions may be malicious
  ➤ Need to also protect user app data from extensions
How can we do this?

**Insight:** it is safe for extension to read user data if it can’t arbitrarily disseminate it

- E.g., Google Mail Checker
- Taint extensions according to what it reads
- Confine code to protect user’s privacy
How can we do this?

**Insight:** it is safe for extension to read user data if it can’t arbitrarily disseminate it

- E.g., Google Mail Checker
  ![Google Mail Checker](image)

- Taint extensions according to what it reads

- Confine code to protect user’s privacy
How can we do this?

Insight: it is safe for extension to read user data if it can’t arbitrarily disseminate it

➤ E.g., Google Mail Checker

➤ Taint extensions according to what it reads

➤ Confine code to protect user’s privacy
How can we do this?

**Insight:** it is safe for extension to read user data if it can’t arbitrarily disseminate it

- E.g., Google Mail Checker
- Taint extensions according to what it reads
- Confine code to protect user’s privacy
How can we do this?

**Insight:** it is safe for extension to read user data if it can’t arbitrarily disseminate it

- E.g., Google Mail Checker

- Taint extensions according to what it reads

- Confine code to protect user’s privacy
How can we do this?

**Insight:** It is safe for extension to read user data if it can’t arbitrarily disseminate it

- E.g., Google Mail Checker
- Taint extensions according to what it reads
- Confine code to protect user’s privacy
Safely read and modify pages?

Add “Adblock Plus”?
It can:
- Read and change all your data on the websites you visit

[Cancel] [Add]
Safely read and modify pages?

Add “Adblock Plus”?
It can:
- Read and change all your data in the websites you visit

[X] Cancel [ ] Add
Safely read and modify pages?

- **Idea:** tie extension script with app page
  - Impose at least same-origin policy on extension

- **Challenge:** read data from page and leak it by injecting content into page’s DOM

- **Solution:** taint extension, write to isolated DOM
  - Loads due to extension restricted: confined!
Safely read and modify pages?

- **Idea:** tie extension script with app page
  - Impose at least same-origin policy on extension

- **Challenge:** read data from page and leak it by injecting content into page’s DOM

- **Solution:** taint extension, write to isolated DOM
  - Loads due to extension restricted: confined!
Confinement: safe, too restricting

- Challenge: extensions need to “leak” data
  - E.g., Evernote is used to save URL, page, etc.
  - Reading DOM taints extension:

- Solution: declassification via sharing menu API
Confinement: safe, too restricting

• Challenge: extensions need to “leak” data
  ➢ E.g., Evernote is used to save URL, page, etc.
  ➢ Reading DOM taints extension:

• Solution: declassification via sharing menu API
Confinement: safe, too restricting

- Challenge: extensions need to “leak” data
  - E.g., Evernote is used to save URL, page, etc.
  - Reading DOM taints extension:

- Solution: declassification via sharing menu API
Confinement: safe, too restricting

• Challenge: extensions need to “leak” data
  ➢ E.g., Evernote is used to save URL, page, etc.
  ➢ Reading DOM taints extension:

• Solution: declassification via sharing menu API
Confinement: safe, too restricting

- Challenge: extensions need to “leak” data
  - E.g., Evernote is used to save URL, page, etc.
  - Reading DOM taints extension:
- Solution: declassification via sharing menu API

![Diagram of NYTimes and Evernote icons with a sharing menu API]

evernote.com
Confinement: safe, too restricting

- Challenge: extensions need to “leak” data
  - E.g., Evernote is used to save URL, page, etc.
  - Reading DOM taints extension:
- Solution: declassification via sharing menu API
Usable confinement via APIs

• Crypto API
  ➤ Convert tainted values to encrypted blobs (LastPass)

• Declarative CSS API
  ➤ Taint-oblivious styling changes

• Network filtering API
  ➤ Allow/deny network requests given regex (AdBlock)

• ...
How can permissions be more meaningful?

• Many extensions can be safe by default
  ➤ Confinement protects user privacy
  ➤ Incentivize developers by making warnings rare

• To capture remaining models: need permissions
  ➤ Use declassification as guide for informing messages: what data is being “leaked”?  
    - E.g., URLs, page location, whole page, etc.
How can permissions be more meaningful?

- Many extensions can be safe by default
  - Confinement protects user privacy
  - Incentivize developers by making warnings rare

- To capture remaining models: need permissions
  - Use declassification as guide for informing messages: what data is being “leaked”?
    - E.g., URLs, page location, whole page, etc.
Summary

• Extensions: most dangerous code in the browser
  ➤ Third-party, unaudited, highly-privileged JavaScript

• Rethink extension security systems
  ➤ Need to protect user privacy from extensions
  ➤ Make user permissions requests rare and clear

• One direction: confinement + new APIs
  ➤ Captures many extensions as “safe”, makes permission requests rare