



German  
OWASP  
Day 2025

***I have no idea how to make it safer:***  
Studying Security and Privacy Mindsets of  
Browser Extension Developers

***Shubham Agarwal, Rafael Mrowczynski, Maria Hellenthal, Ben Stock***

German OWASP Day 2025 - Düsseldorf, DE

whoami



German  
**OWASP**  
Day 2025



**Shubham Agarwal**

whoami



German  
**OWASP**  
Day 2025



## Shubham Agarwal

Security and Privacy Engineering (SPRING) Lab

Web Applications Security, Data Privacy, Large-scale Vulnerability Analysis, Browser Extensions, **Stakeholders Analysis**

whoami



German  
**OWASP**  
Day 2025

- Scientific Researcher at Max Planck Institute-Security and Privacy, Bochum, DE.



## Shubham Agarwal

Security and Privacy Engineering (SPRING) Lab

Web Applications Security, Data Privacy, Large-scale Vulnerability Analysis, Browser Extensions, **Stakeholders Analysis**

whoami



German  
**OWASP**  
Day 2025



- Scientific Researcher at Max Planck Institute-Security and Privacy, Bochum, DE.
- Likes to talk/hear on all things Applications Security & Data Privacy.

## **Shubham Agarwal**

Security and Privacy Engineering (SPRING) Lab

Web Applications Security, Data Privacy, Large-scale Vulnerability Analysis, Browser Extensions, **Stakeholders Analysis**



German  
**OWASP**  
Day 2025



- Scientific Researcher at Max Planck Institute-Security and Privacy, Bochum, DE.
- Likes to talk/hear on all things Applications Security & Data Privacy.
- Now (also) interested in Digital Harms & S&P Guidelines for Stakeholders.

## **Shubham Agarwal**

Security and Privacy Engineering (SPRING) Lab

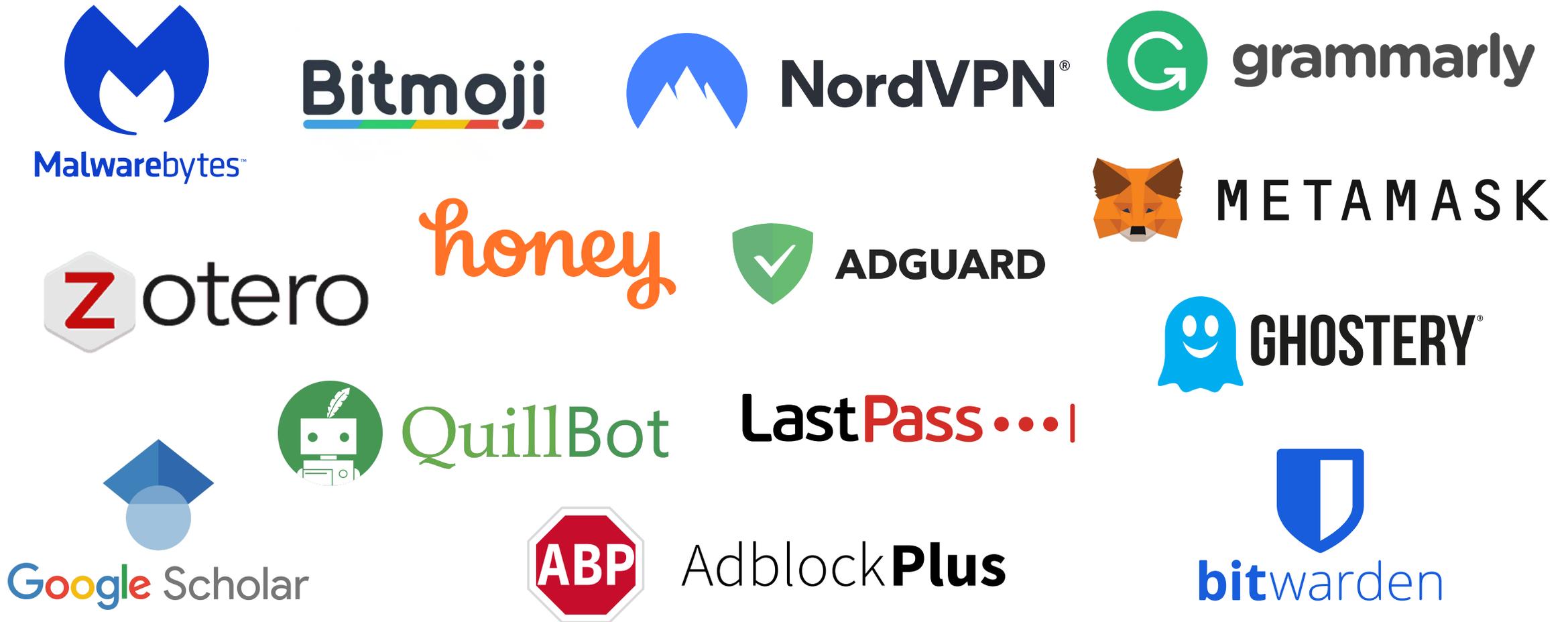
Web Applications Security, Data Privacy, Large-scale Vulnerability Analysis, Browser Extensions, **Stakeholders Analysis**

# Have you developed/used browser extensions?

---



# Have you developed/used browser extensions?



# Why do we care about extensions?

---



# Why do we care about extensions?

---



## Supply chain attack hits Chrome extensions, could expose millions

Threat actor exploited phishing and OAuth abuse to inject malicious code

## Compromised Browser Extensions - A Growing Threat Vector

Learn how threat actors leverage browser extensions as an attack vector, including examples for Cyberhaven and GraphQL Network Inspector.



**Pulsedive Threat Research**

25 Feb 2025 · 15 min read

## Universal Code Execution by Chaining Messages in Browser Extensions

Jul 7, 2024 · 2111 words · 10 minute read

# Why do we care about extensions?



## Supply chain attack hits Chrome extensions, could expose millions

Threat actor exploited phishing and OAuth abuse to inject malicious code



### Universal Code Execution by Chaining Messages in Browser Extensions

Jul 7, 2024 · 2111 words · 10 minute read

## Compromised Browser Extensions - A Growing Threat Vector

Learn how threat actors leverage browser extensions as an attack vector, including examples for Cyberhaven and GraphQL Network Inspector.



**Pulsedive Threat Research**

25 Feb 2025 · 15 min read

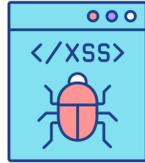
# What do we know, what have we learnt?

---



# What do we know, what have we learnt?

---



**Cross-Site Scripting**



**Information Leaks**



**Security Header Modifications**



**Extension Fingerprinting**

# What do we know, what have we learnt?



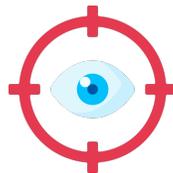
**Cross-Site Scripting**



**Information Leaks**



**Highlight: Malicious Benign-but-Buggy Extensions**



**Security Header Modifications**



**Extension Fingerprinting**



*What is the Security & Privacy Mindset of Browser Extension Developers?*



## *What is the Security & Privacy Mindset of Browser Extension Developers?*



Qualitative Interviews  
with **21 Browser  
Extension Developers**



## *What is the Security & Privacy Mindset of Browser Extension Developers?*



Qualitative Interviews  
with **21 Browser  
Extension Developers**



**Development Experience**



## *What is the Security & Privacy Mindset of Browser Extension Developers?*



Qualitative Interviews  
with **21 Browser  
Extension Developers**



**Development Experience**



**Coding Tasks (CTs)**



## *What is the Security & Privacy Mindset of Browser Extension Developers?*



Qualitative Interviews  
with **21 Browser  
Extension Developers**



**Development Experience**



**Coding Tasks (CTs)**



**S&P Mindset**

*~120 minutes of Semi-structured Interviews*

*Solution Strategy >>> Solution*

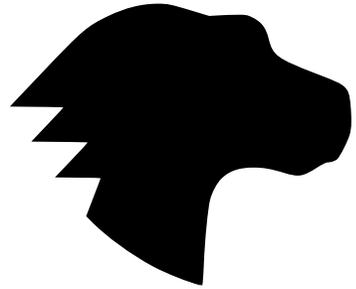
# Real-World Development Settings

---

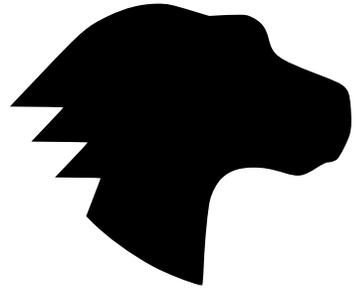


# Real-World Development Settings

---



# Real-World Development Settings



# Case I: Privacy Mindset of Developers

---



# Case I: Privacy Mindset of Developers

---



Task: *Allow Autofill of Address Data on Purchase Page.*

# Case I: Privacy Mindset of Developers



Task: *Allow Autofill of Address Data on Purchase Page.*

**Order Details**

Address Line 1

Address Line 2

City

State

Country

Pincode

Number of Copies

Language

English

Save Address for Later

Autofill Address Data

Continue to Payment

# Case I: Privacy Mindset of Developers



Task: *Allow Autofill of Address Data on Purchase Page.*

### Order Details

Language

```
async function recordAddressData(addressData) {  
  try {  
    console.log("Address Data:", addressData);  
    // do something here.  
  } catch (e) {  
    console.error(e);  
  }  
}  
  
async function retrieveAddress() {  
  try {  
    // do something here.  
  } catch (e) {  
    console.error(e);  
  }  
}
```

# Case I: Actions & Observations

---



# Case I: Actions & Observations



*“...we would collect I guess the address we would then store that through the **localStorage API**”*

Privacy-unfriendly data storage strategy:  
**Web Storage APIs >>> chrome.storage APIs**

Unclear privacy benefits of the dedicated **chrome.storage** API over **Web Storage** APIs.

*“...if I were to call **chrome.storage.sync API** here and save this address, would other extensions have access to this?”*

# Case I: Actions & Observations



*“...we would collect I guess the address we would then store that through the **localStorage API**”*

Privacy-unfriendly data storage strategy:  
**Web Storage APIs >>> chrome.storage APIs**

Unclear privacy benefits of the dedicated **chrome.storage** API over **Web Storage** APIs.

*“...if I were to call **chrome.storage.sync API** here and save this address, would other extensions have access to this?”*

***The S&P benefits of existing APIs and their actions are, at best, unclear to developers.***

# Case I: What do vendors say?

---



# Case I: What do vendors say?



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.

You can examine the stored data under the Extension Storage item in the [Storage Inspector](#) tab of the [developer toolbox](#), accessible from `about:debugging`.

ⓘ **Note:** The storage area is not encrypted and shouldn't be used for storing confidential user information.

<https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/storage>

# Case I: What do vendors say?



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.

You can examine the stored data under the Extension Storage item in the [Storage Inspector](#) tab of the [developer toolbox](#), accessible from `about:debugging`.

ⓘ **Note:** The storage area is not encrypted and shouldn't be used for storing confidential user information.

<https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/storage>

## Can extensions use web storage APIs?

While extensions can use the [Storage](#) interface (accessible from `window.localStorage`) in some contexts (popup and other HTML pages), we don't recommend it for the following reasons:

- Extension service workers can't use the Web Storage API.
- Content scripts share storage with the host page.
- Data saved using the Web Storage API is lost when the user clears their browsing history.

<https://developer.chrome.com/docs/extensions/reference/api/storage>

# Case I: What do vendors say?



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.

You can examine the stored data under the Extension Storage item in the [Storage Inspector](#) tab of the [developer toolbox](#), accessible from `about:debugging`.

**Note:** The storage area is not encrypted and shouldn't be used for storing confidential user information.

***Interestingly, official documents also lacks explicit S&P context! (Surprise?!)*** 🤪

<https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API/storage>

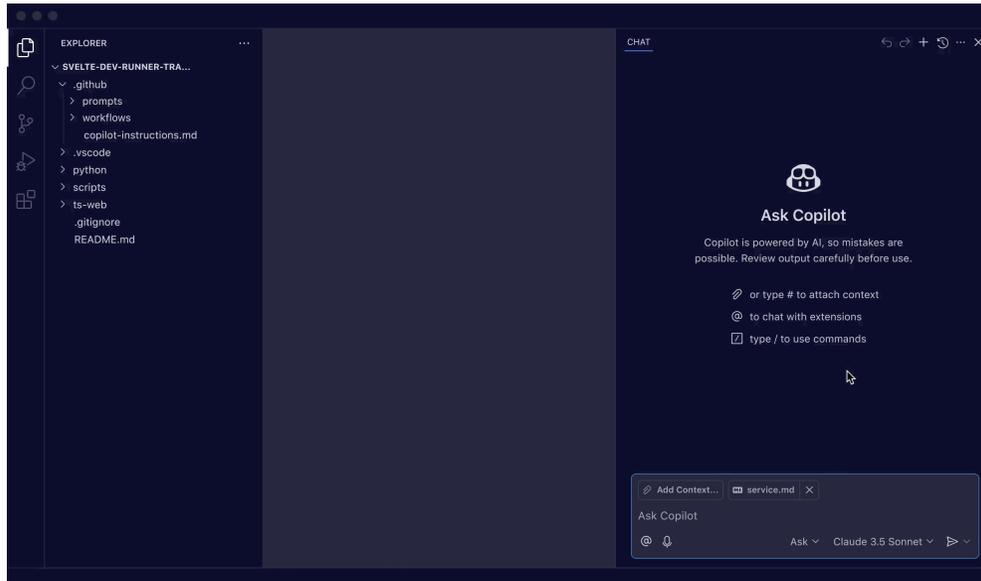
## Can extensions use web storage APIs?

While extensions can use the `Storage` interface (accessible from `window.localStorage`) in some contexts (popup and other HTML pages), we don't recommend it for the following reasons:

- Extension service workers can't use the Web Storage API.
- Content scripts share storage with the host page.
- Data saved using the Web Storage API is lost when the user clears their browsing history.

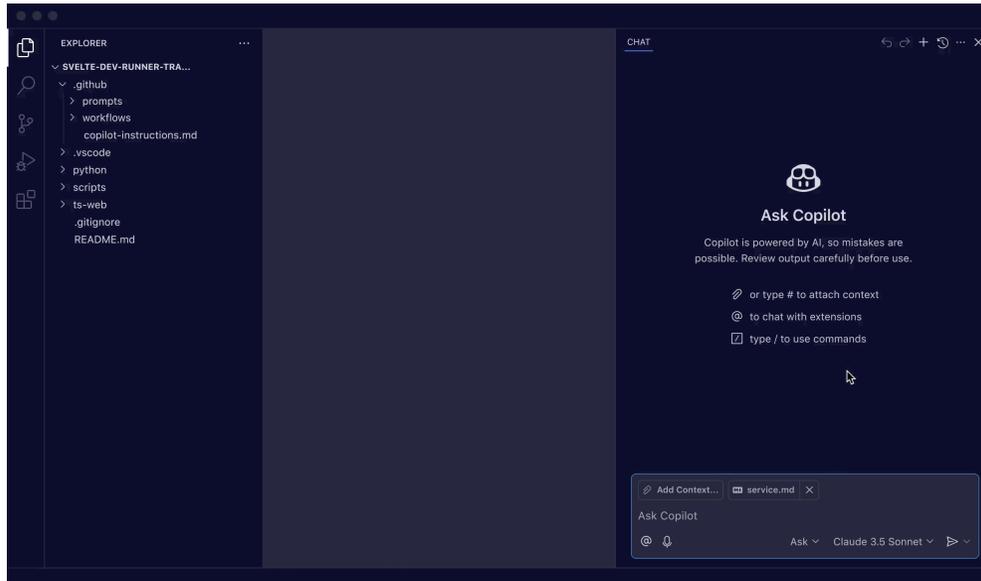
<https://developer.chrome.com/docs/extensions/reference/api/storage>

# Development Sources – AI Assistants



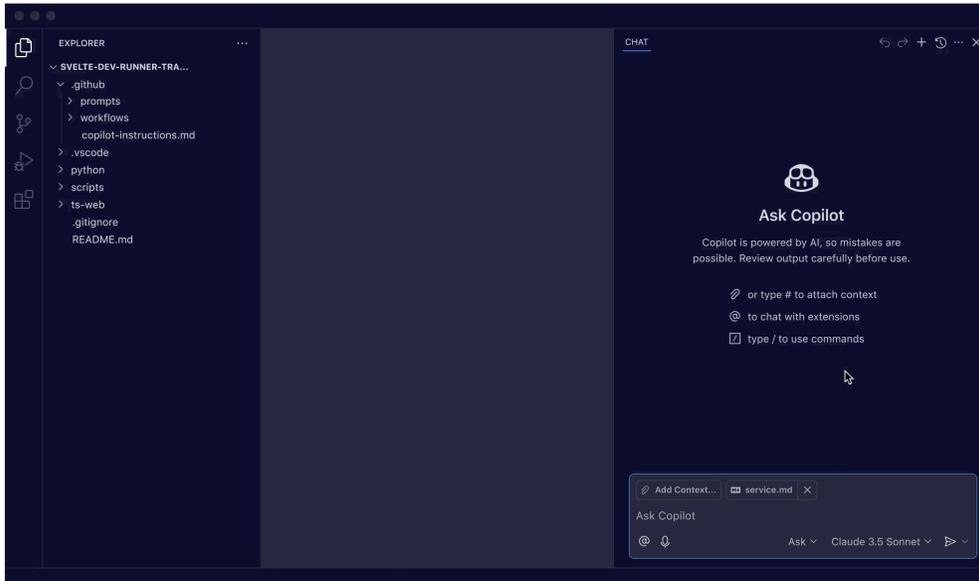
<https://github.com/features/copilot>

# Development Sources – AI Assistants



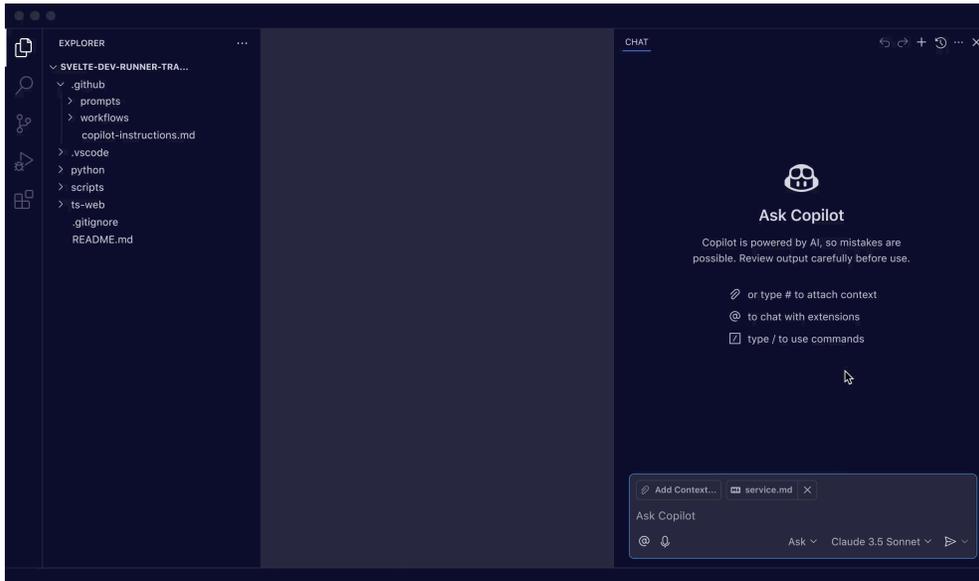
<https://github.com/features/copilot>

# Development Sources – AI Assistants



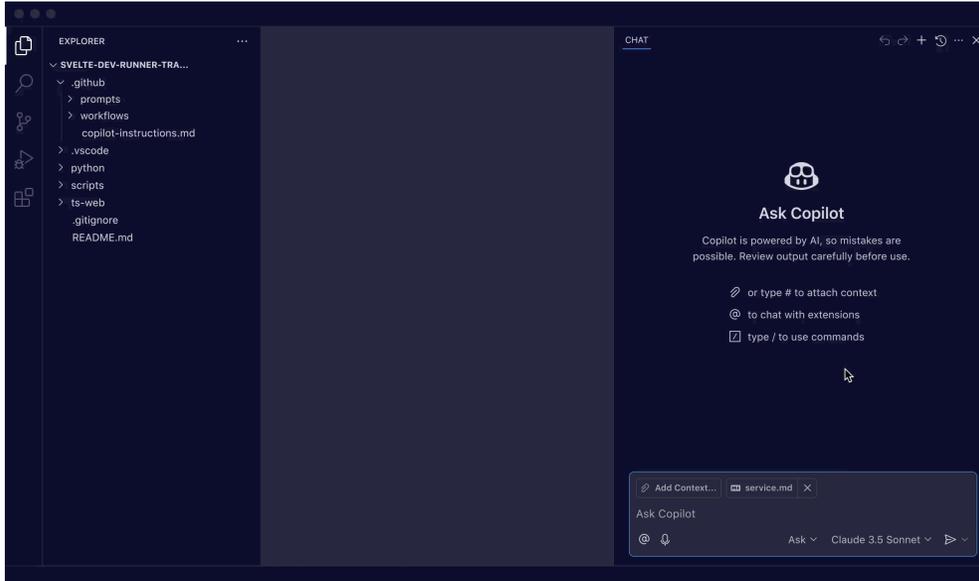
<https://cdn.supermaven.com/video/montage.mp4>

# Development Sources – AI Assistants



<https://cdn.supermaven.com/video/montage.mp4>

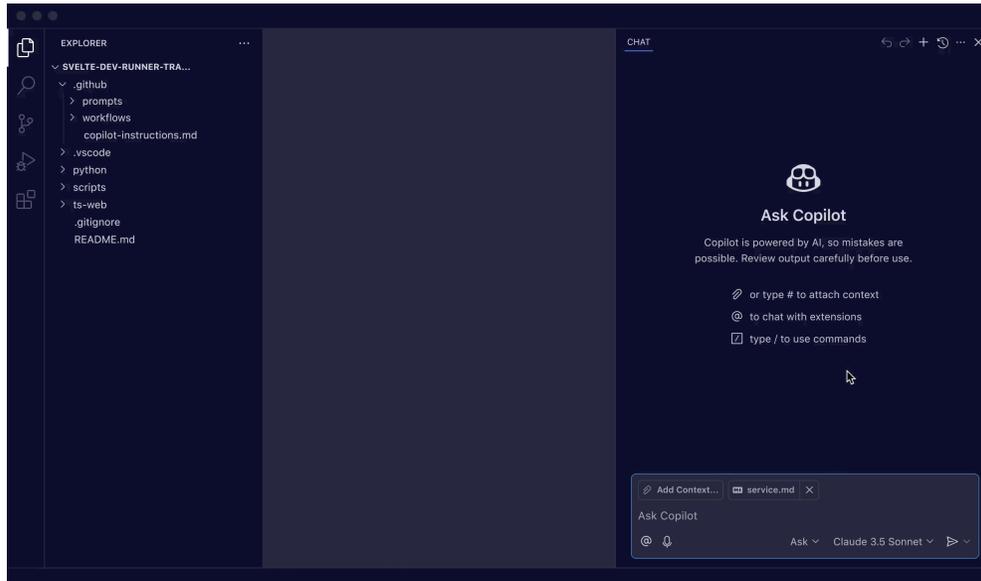
# Development Sources – AI Assistants



<https://cdn.supermaven.com/video/montage.mp4>



# Development Sources – AI Assistants



<https://cdn.supermaven.com/video/montage.mp4>

***AI Assistants may only be helpful to the extent of your knowledge!***



# How did the AI Assistants fare?

---



# How did the AI Assistants fare?



```
async function recordAddressData(addressData) {  
  try {  
    console.log("Address Data:", addressData);  
    // do something here.  
    localStorage.setItem('addressData', JSON.stringify(addressData));  
  } catch (e) {  
    console.error(e);  
  }  
}
```

# How did the AI Assistants fare?



```
async function recordAddressData(addressData) {  
  try {  
    console.log("Address Data:", addressData);  
    // do something here.  
    localStorage.setItem('addressData', JSON.stringify(addressData));  
  } catch (e) {  
    console.error(e);  
  }  
}
```

```
*/  
async function recordAddressData(addressData) : Promise<any> { Show usages  
  chrome.storage.local.set({ addressData });  
}  
  
async function retrieveAddress() : Promise<void> { Show usages  
  try {  
    // do something here.  
  } catch (e) {  
    console.error(e);  
  }  
}
```

# How did the AI Assistants fare?



```
async function recordAddressData(addressData) {
  try {
    console.log("Address Data:", addressData);
    // do something here.
    localStorage.setItem('addressData', JSON.stringify(addressData));
  } catch (e) {
    console.error(e);
  }
}

*/
async function recordAddressData(addressData) : Promise<any> { Show usages
  chrome.storage.local.set({ addressData });
}

async function retrieveAddress() : Promise<void> { Show usages
  try {
    // do something here.
  } catch (e) {
    console.error(e);
  }
}
```

***The developer “seed” could determine the solution!***

# Case II: Security Mindset of Developers

---



# Case II: Security Mindset of Developers

---



Task: *Embed amazon.de in an iframe to check for corresponding product price.*

# Case II: Security Mindset of Developers



Task: *Embed amazon.de in an iframe to check for corresponding product price.*

**Clean Code: A Handbook of Agile Software Craftsmanship**  
Robert C. Martin

~~\$48.99~~ \$9.49

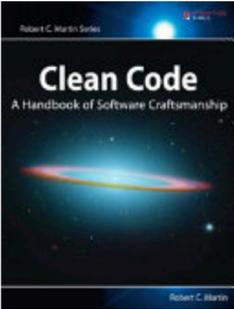
[Order Now](#)

[Check on Amazon](#)

# Case II: Security Mindset of Developers



Task: *Embed amazon.de in an iframe to check for corresponding product price.*



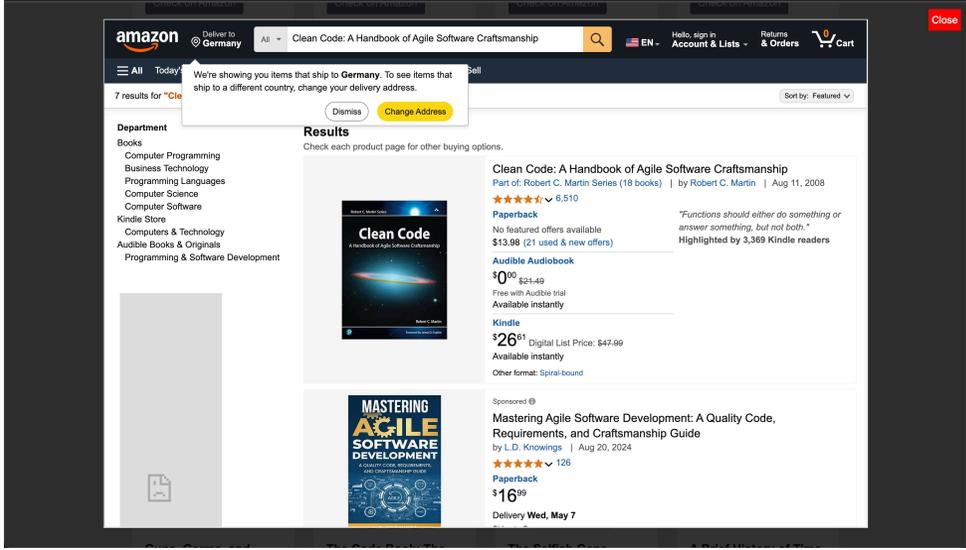
**Clean Code: A Handbook of Agile Software Craftmanship**

Robert C. Martin

~~\$48.99~~ \$9.49

[Order Now](#)

[Check on Amazon](#)



The screenshot shows the Amazon.de product page for 'Clean Code: A Handbook of Agile Software Craftmanship'. The page is in German and shows the product title, author (Robert C. Martin), and a price of €13.98. The page also displays a star rating of 4.5 out of 5 stars and a 'Paperback' format. The page is framed by a dark border, suggesting it is an iframe.

# Case II: Security Mindset of Developers



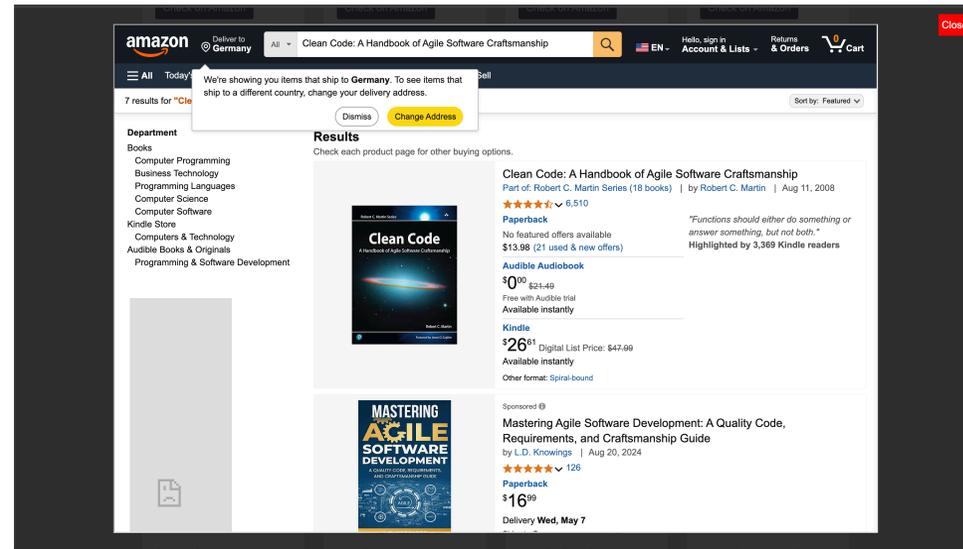
Task: *Embed amazon.de in an iframe to check for corresponding product price.*



**Clean Code: A Handbook of Agile Software Craftsmanship**  
Robert C. Martin

~~\$18.99~~ \$9.49

[Order Now](#)  
[Check on Amazon](#)



amazon Germany Clean Code: A Handbook of Agile Software Craftsmanship

We're showing you items that ship to Germany. To see items that ship to a different country, change your delivery address.

7 results for "Clean Code: A Handbook of Agile Software Craftsmanship"

Department: Books

Check each product page for other buying options.

**Clean Code: A Handbook of Agile Software Craftsmanship**  
Part of: Robert C. Martin Series (18 books) | by Robert C. Martin | Aug 11, 2008  
★★★★★ 6,510

**Paperback**  
No featured offers available  
\$13.98 (21 used & new offers)

"Functions should either do something or answer something, but not both."  
Highlighted by 3,369 Kindle readers

**Audible Audiobook**  
\$0.00 \$21.49  
Free with Audible trial  
Available instantly

**Kindle**  
\$26.99 Digital List Price: \$47.99  
Available instantly  
Other format: Spiral-bound

Sponsored @  
**Mastering Agile Software Development: A Quality Code, Requirements, and Craftsmanship Guide**  
by L.D. Knowings | Aug 20, 2024  
★★★★★ 126  
**Paperback**  
\$16.99  
Delivery Wed, May 7

```
X-Cache NotCacheable from child
X-Content-Type-Options nosniff
X-Frame-Options SAMEORIGIN
X-Ua-Compatible IE=edge
X-Xss-Protection 1;
```

# Case II: Actions & Observations

---



# Case II: Actions & Observations

---



*“...can't really think of another way to get around this!”*

# Case II: Actions & Observations

---



*“...can't really think of another way to get around this!”*

*“...it seemed a little bit, bit hacky.”*

# Case II: Actions & Observations



*"...can't really think of another way to get around this!"*

*"...it seemed a little bit, bit hacky."*

Preferred to explore potential alternatives (new tabs, popups, etc.)

# Case II: Actions & Observations



*“...can’t really think of another way to get around this!”*

*“...it seemed a little bit, bit hacky.”*

Preferred to explore potential alternatives (new tabs, popups, etc.)

***Developers may even be inherently mindful of overly security-critical actions and take responsive decisions even when not aware of its implications.***

# What did the LLMs think of the challenge?

---



# What did the LLMs think of the challenge?



Unfortunately, if a website sets the `X-Frame-Options` header to disallow framing (such as with values like `DENY` or `SAMEORIGIN`), it's a security feature implemented by the website itself, and there is no way to bypass it from the client side legally or ethically. This is designed to prevent clickjacking attacks and ensure that websites cannot be embedded in unauthorized third-party pages.

Here are a few legitimate alternatives:

1. **API or Data Embedding:** If you're trying to include content or data from another website, check if they provide an API. This can allow you to fetch and embed the data or content without using an iframe.
2. **Redirecting Users:** If you need to send users to that website, consider adding a link or a button that redirects them to the site in a new tab, rather than embedding it.
3. **Proxy Server (with Permission):** If you have control of the backend or have permission from the target site, you can set up a proxy on your server to retrieve the content and display it. However, this requires both technical permission and respect for the website's terms of service.
4. **Contacting the Website Owner:** Sometimes, website owners can whitelist specific domains to allow embedding. If this is important for your project, consider reaching out to them to see if they would permit it.



Attempting to bypass these protections without permission can lead to legal consequences and is against most websites' terms of use.



# What did the LLMs think of the challenge?



Unfortunately, if a website sets the `X-Frame-Options` header to disallow framing (such as with values like `DENY` or `SAMEORIGIN`), it's a security feature implemented by the website itself, and there is no way to bypass it from the client side legally or ethically. This is designed to prevent clickjacking attacks and ensure that websites cannot be embedded in unauthorized third-party pages.

Here are a few legitimate alternatives:

1. **API or Data Embedding:** If you're trying to include content or data from another website, check if they provide an API. This can allow you to fetch and embed the data or content without using an iframe.
2. **Redirecting Users:** If you need to send users to that website, consider adding a link or a button that redirects them to the site in a new tab, rather than embedding it.
3. **Proxy Server (with Permission):** If you have control of the backend or have permission from the target site, you can set up a proxy on your server to retrieve the content and display it. However, this requires both technical permission and respect for the website's terms of service.
4. **Contacting the Website Owner:** Sometimes, website owners can whitelist specific domains to allow embedding. If this is important for your project, consider reaching out to them to see if they would permit it.



Attempting to bypass these protections without permission can lead to legal consequences and is against most websites' terms of use.



# What did the LLMs think of the challenge?



Unfortunately, if a website sets the `X-Frame-Options` header to disallow framing (such as with values like `DENY` or `SAMEORIGIN`), it's a security feature implemented by the website itself, and there is no way to bypass it from the client side legally or ethically. This is designed to prevent clickjacking attacks and ensure that websites cannot be embedded in unauthorized third-party pages.

Here are a few legitimate alternatives:

1. **API or Data Embedding:** If you're trying to include content or data from another website, check if they provide an API. This can allow you to fetch and embed the data or content without using an iframe.
2. **Redirecting Users:** If you need to send users to that website, consider adding a link or a button that redirects them to the site in a new tab, rather than embedding it.
3. **Proxy Server (with Permission):** If you have control of the backend or have permission from the target site, you can set up a proxy on your server to retrieve the content and display it. However, this requires both technical permission and respect for the website's terms of service.
4. **Contacting the Website Owner:** Sometimes, website owners can whitelist specific domains to allow embedding. If this is important for your project, consider reaching out to them to see if they would permit it.

Attempting to bypass these protections without permission can lead to legal consequences and is against most websites' terms of use.



```
{
  "header": "X-Frame-Options",
  "operation": "remove"
}
],
"condition": {
  "urlFilter": "example.com",
  "resourceTypes": ["main_frame", "sub_frame"]
}
}
```

However, this will not work for security headers like `X-Frame-Options`, as Chrome restricts this modification.

## Alternative Approach:

- **Requestly Extension:** If you are testing or need a quick fix for development purposes, you can use an extension like Requestly, which allows you to modify response headers directly within the browser. This is a tool that can help you bypass `X-Frame-Options` in a non-production environment for testing purposes.

## Conclusion:

Chrome extensions, for security reasons, cannot use `declarativeNetRequest` to bypass `X-Frame-Options` directly. The best approach for bypassing this limitation is setting up a proxy server or using an alternative tool for local development and testing.

# What did the LLMs think of the challenge?



Unfortunately, if a website sets the `X-Frame-Options` header to disallow framing (such as with values like `DENY` or `SAMEORIGIN`), it's a security feature implemented by the website itself, and there is no way to bypass it from the client side legally or ethically. This is designed to prevent clickjacking attacks and ensure that websites cannot be embedded in unauthorized third-party pages.

Here are a few legitimate alternatives:

1. **API or Data Embedding:** If you're trying to include content or data from another website, check if they provide an API. This can allow you to fetch and embed the data or content without using an iframe.
2. **Redirecting Users:** If you need to send users to that website, consider adding a link or a button that redirects them to the site in a new tab, rather than embedding it.
3. **Proxy Server (with Permission):** If you have control of the backend or have permission from the target site, you can set up a proxy on your server to retrieve the content and display it. However, this requires both technical permission and respect for the website's terms of service.
4. **Contacting the Website Owner:** Sometimes, website owners can whitelist specific domains to allow embedding. If this is important for your project, consider reaching out to them to see if they would permit it.

Attempting to bypass these protections without permission can lead to legal consequences and is against most websites' terms of use.



```
{
  "header": "X-Frame-Options",
  "operation": "remove"
}
],
"condition": {
  "urlFilter": "example.com",
  "resourceTypes": ["main_frame", "sub_frame"]
}
}
```

However, this will not work for security headers like `X-Frame-Options`, as Chrome restricts this modification.

## Alternative Approach:

- **Requestly Extension:** If you are testing or need a quick fix for development purposes, you can use an extension like Requestly, which allows you to modify response headers directly within the browser. This is a tool that can help you bypass `X-Frame-Options` in a non-production environment for testing purposes.

## Conclusion:

Chrome extensions, for security reasons, cannot use `declarativeNetRequest` to bypass `X-Frame-Options` directly. The best approach for bypassing this limitation is setting up a proxy server or using an alternative tool for local development and testing.

# What did the LLMs think of the challenge?



Unfortunately, if a website sets the `X-Frame-Options` header to disallow framing (such as with values like `DENY` or `SAMEORIGIN`), it's a security feature implemented by the website itself, and there is no way to bypass it from the client side legally or ethically. This is designed to prevent clickjacking attacks and ensure that websites cannot be embedded in unauthorized third-party pages.

Here are a few legitimate alternatives:

1. **API or Data Embedding:** If you're trying to include content or data from another website, check if they provide an API. This can allow you to fetch and embed the data or content without using an iframe.
2. **Redirecting Users:** If you need to send users to that website, consider adding a link or a button that redirects them to the site in a new tab, rather than embedding it.
3. **Proxy Server (with Permission):** If you have control of the backend or have permission from the target site, you can set up a proxy on your server to retrieve the content and display it. However, this requires both technical permission and respect for the website's terms of service.
4. **Contacting the Website Owner:** Sometimes, website owners can whitelist specific domains to allow embedding. If this is important for your project, consider reaching out to them to see if they would permit it.

Attempting to bypass these protections without permission can lead to legal consequences and is against most websites' terms of use.



```
{
  "header": "X-Frame-Options",
  "operation": "remove"
}
],
"condition": {
  "uriFilter": "example.com",
  "resourceTypes": ["main_frame", "sub_frame"]
}
}
```

However, this will not work for security headers like `X-Frame-Options`, as Chrome restricts this modification.

## Alternative Approach:

- **Requestly Extension:** If you are testing or need a quick fix for development purposes, you can use an extension like Requestly, which allows you to modify response headers directly within the browser. This is a tool that can help you bypass `X-Frame-Options` in a non-production environment for testing purposes.

## Conclusion:

Chrome extensions, for security reasons, cannot use `declarativeNetRequest` to bypass `X-Frame-Options` directly. The best approach for bypassing this limitation is setting up a proxy server or using an alternative tool for local development and testing.

***“Security” & “Legal” concerns of LLMs often serve as strong indicator of overall correctness!***





## Trustworthy Chrome Extensions, by default

Monday, October 1, 2018

<https://blog.chromium.org/2018/10/trustworthy-chrome-extensions-by-default.html>



## Trustworthy Chrome Extensions, by default

Monday, October 1, 2018

<https://blog.chromium.org/2018/10/trustworthy-chrome-extensions-by-default.html>

### Deal with remote hosted code violations



Remotely hosted code, or RHC, is what the Chrome Web Store calls anything that is executed by the browser that is loaded from someplace other than the extension's own files. Things like JavaScript and WASM. It *does not* include data or things like JSON or CSS.

#### Why is RHC no longer allowed?

With Manifest V3 extensions now need to bundle *all* code they are using inside the extension itself. In the past, you could dynamically inject script tags from any URL on the web.

<https://developer.chrome.com/docs/extensions/develop/migrate/remote-hosted-code>





- Goal: Limit damage through untrusted and non-vetted code.

# Security, Privacy & Compliance

---



- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?



# Security, Privacy & Compliance

---



- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?
  - Allows customized JavaScript execution.





- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?
  - Allows customized JavaScript execution.
- Is it too bad?





- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?
  - Allows customized JavaScript execution.
- Is it too bad?
  - Well, extensions can inject iframes (with srcdoc) on any page.





- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?
  - Allows customized JavaScript execution.
- Is it too bad?
  - Well, extensions can inject iframes (with srcdoc) on any page.
  - `postMessage` helps, if you are still wondering!





- Goal: Limit damage through untrusted and non-vetted code.
- What about page-to-extension data flow and iframes then?
  - Allows customized JavaScript execution.
- Is it too bad?
  - Well, extensions can inject iframes (with srcdoc) on any page.
  - `postMessage` helps, if you are still wondering!



***Browser extension still compliant with store policies!***

# Threat Perception & Modeling

---



# Threat Perception & Modeling

---



- *“...Websites cannot attack extensions!”*
  - *“...only theoretically possible.”*



# Threat Perception & Modeling

---



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”

***No threats, no problem!***



# Threat Perception & Modeling

---



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.

***No threats, no problem!***



# Threat Perception & Modeling



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.

***No threats, no problem!***

***web accessible resources  
et al. are useless!***



# Threat Perception & Modeling



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.
- Inherent trust on platform-exposed APIs.
  - Trust & Safety >>> Security & Privacy.

***No threats, no problem!***

***web accessible resources  
et al. are useless!***



# Threat Perception & Modeling



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.
- Inherent trust on platform-exposed APIs.
  - Trust & Safety >>> Security & Privacy.

***No threats, no problem!***

***web accessible resources  
et al. are useless!***

***Future-proofing is priority!***



# Threat Perception & Modeling



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.
- Inherent trust on platform-exposed APIs.
  - Trust & Safety >>> Security & Privacy.
- All extension developers are not actually *developers*.

**No threats, no problem!**

**web accessible resources  
et al. are useless!**

**Future-proofing is priority!**



# Threat Perception & Modeling



- “...Websites cannot attack extensions!”
  - “...only theoretically possible.”
- Little to no idea of isolation boundaries.
- Inherent trust on platform-exposed APIs.
  - Trust & Safety >>> Security & Privacy.
- All extension developers are not actually *developers*.

**No threats, no problem!**

**web accessible resources  
et al. are useless!**

**Future-proofing is priority!**

**Vibe-coding will thrive...**



# Other S&P Remarks

---



# Other S&P Remarks

---



# Other S&P Remarks

---



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*



# Other S&P Remarks

---



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*



# Other S&P Remarks

---



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*
  - Third-party scripts make it tricky.



# Other S&P Remarks

---



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*
  - Third-party scripts make it tricky.
- *“...these companies just scrape a lot of information from the user. I think actually the stores don’t do so much against it”.*



# Other S&P Remarks

---



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*
  - Third-party scripts make it tricky.
- *“...these companies just scrape a lot of information from the user. I think actually the stores don’t do so much against it”.*
  - (Even pay around \$10K per extension, if you are wondering).



# Other S&P Remarks



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*
  - Third-party scripts make it tricky.
- *“...these companies just scrape a lot of information from the user. I think actually the stores don’t do so much against it”.*
  - (Even pay around \$10K per extension, if you are wondering).
- *“...users shouldn’t be held responsible!”*



# Other S&P Remarks



- *“Privacy Policies and Data Protection Laws are hard (to even understand)!”*
  - *“...trust platforms to take care of it.”*
  - Third-party scripts make it tricky.
- *“...these companies just scrape a lot of information from the user. I think actually the stores don’t do so much against it”.*
  - (Even pay around \$10K per extension, if you are wondering).
- *“...users shouldn’t be held responsible!”*
  - Then, who should act as the wise man here?



# Summary

---



# Summary

---



- The definition of security and privacy is highly subjective.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.
- (Un-)Surprisingly enough, assistance by LLMs are only as helpful as developers' understanding.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.
- (Un-)Surprisingly enough, assistance by LLMs are only as helpful as developers' understanding.
  - Right S&P outcome ≠ Right S&P processes.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.
- (Un-)Surprisingly enough, assistance by LLMs are only as helpful as developers' understanding.
  - Right S&P outcome ≠ Right S&P processes.
  - Mere availability of deprecated APIs make it worse.



# Summary

---



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.
- (Un-)Surprisingly enough, assistance by LLMs are only as helpful as developers' understanding.
  - Right S&P outcome ≠ Right S&P processes.
  - Mere availability of deprecated APIs make it worse.
- Dominance of browser vendors influences the implementation and their S&P outcomes.



# Summary



- The definition of security and privacy is highly subjective.
  - often confused with trust and safety.
  - Improper threat modeling → incorrect prioritization and measures.
- (Un-)Surprisingly enough, assistance by LLMs are only as helpful as developers' understanding.
  - Right S&P outcome ≠ Right S&P processes.
  - Mere availability of deprecated APIs make it worse.
- Dominance of browser vendors influences the implementation and their S&P outcomes.



Find more details about our study here.



**THANK YOU!**



@shubh401



@ap0ca1ypse.bsky.social