

On the Security and Privacy Risks of Browser Extensions

Dr.-Ing. Aurore Fass

Tenured Researcher — Inria Centre at Université Côte d'Azur (since Dec 1, 2025)

Dr.-Ing. Aurore (/ɔʁɔʁ/) FASS

🇫🇷 2017: Graduated from [TELECOM Nancy](#) (FR)



🇩🇪 2017–21: PhD Student + Postdoc at [CISPA](#) (DE)

🇺🇸 2021–23: Visiting Assistant Professor at [Stanford](#) (US)



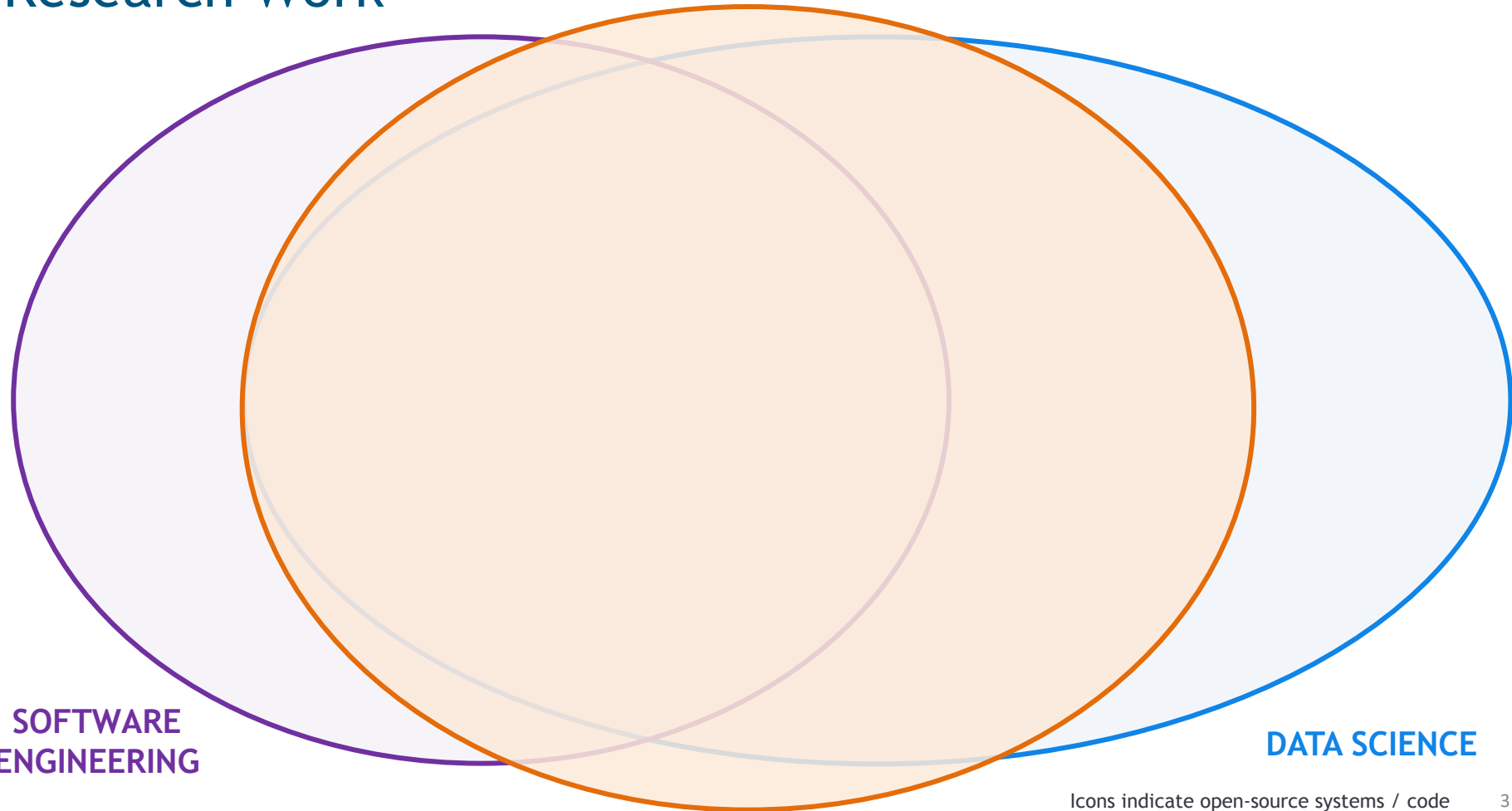
🇩🇪 2023–25: Tenure-Track Faculty W2 at [CISPA](#) (DE)

🇫🇷 2025–: Tenured Researcher at [Inria](#) (FR)



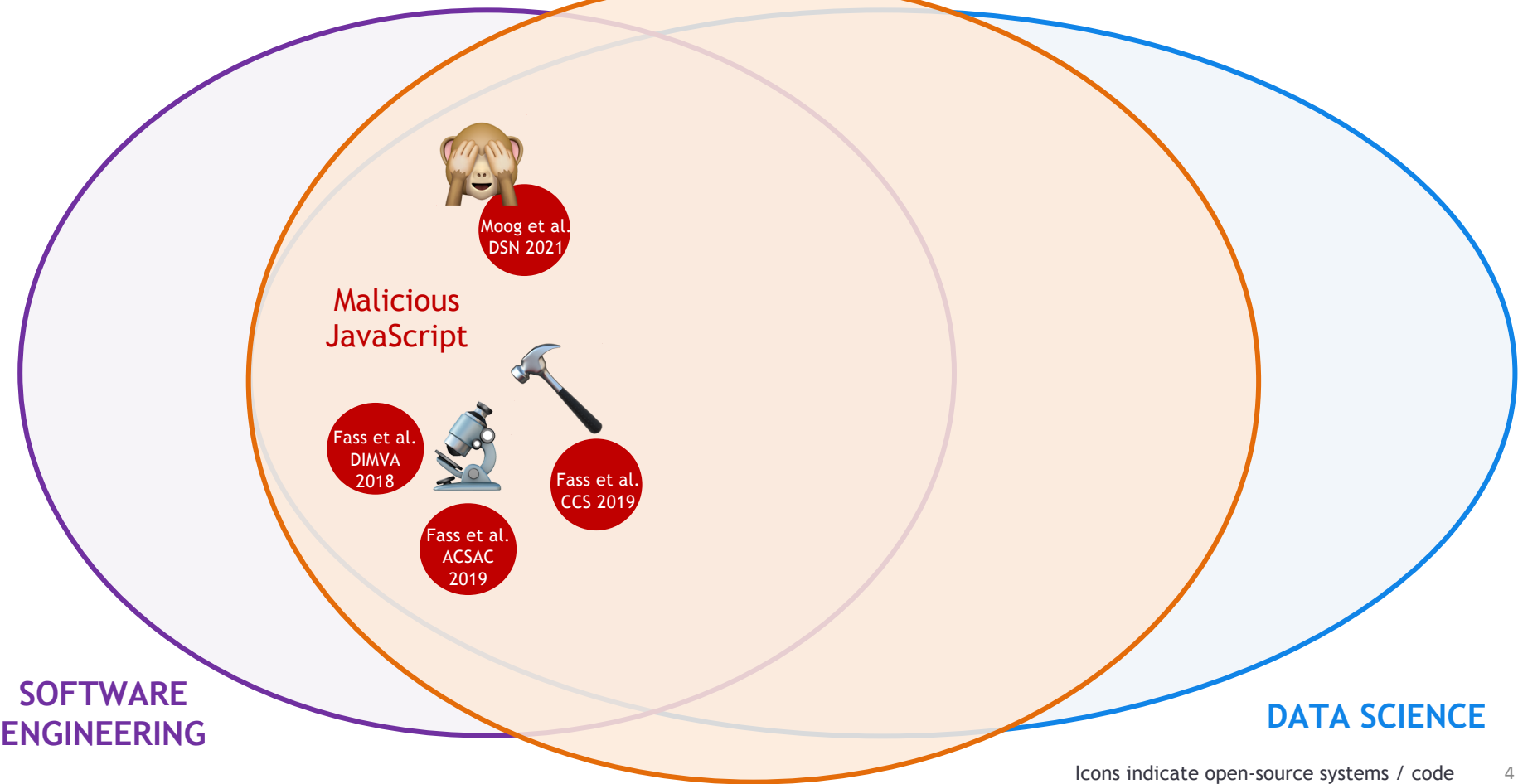
Research Work

WEB SECURITY & PRIVACY



SOFTWARE
ENGINEERING

DATA SCIENCE



Malicious JavaScript

Browser Extensions



Moog et al.
DSN 2021



Agarwal et al.
CCS 2024



Rosenzweig et al.
TWEB 2026

Fass et al.
DIMVA 2018



Fass et al.
CCS 2019

Fass et al.
ACSAC 2019

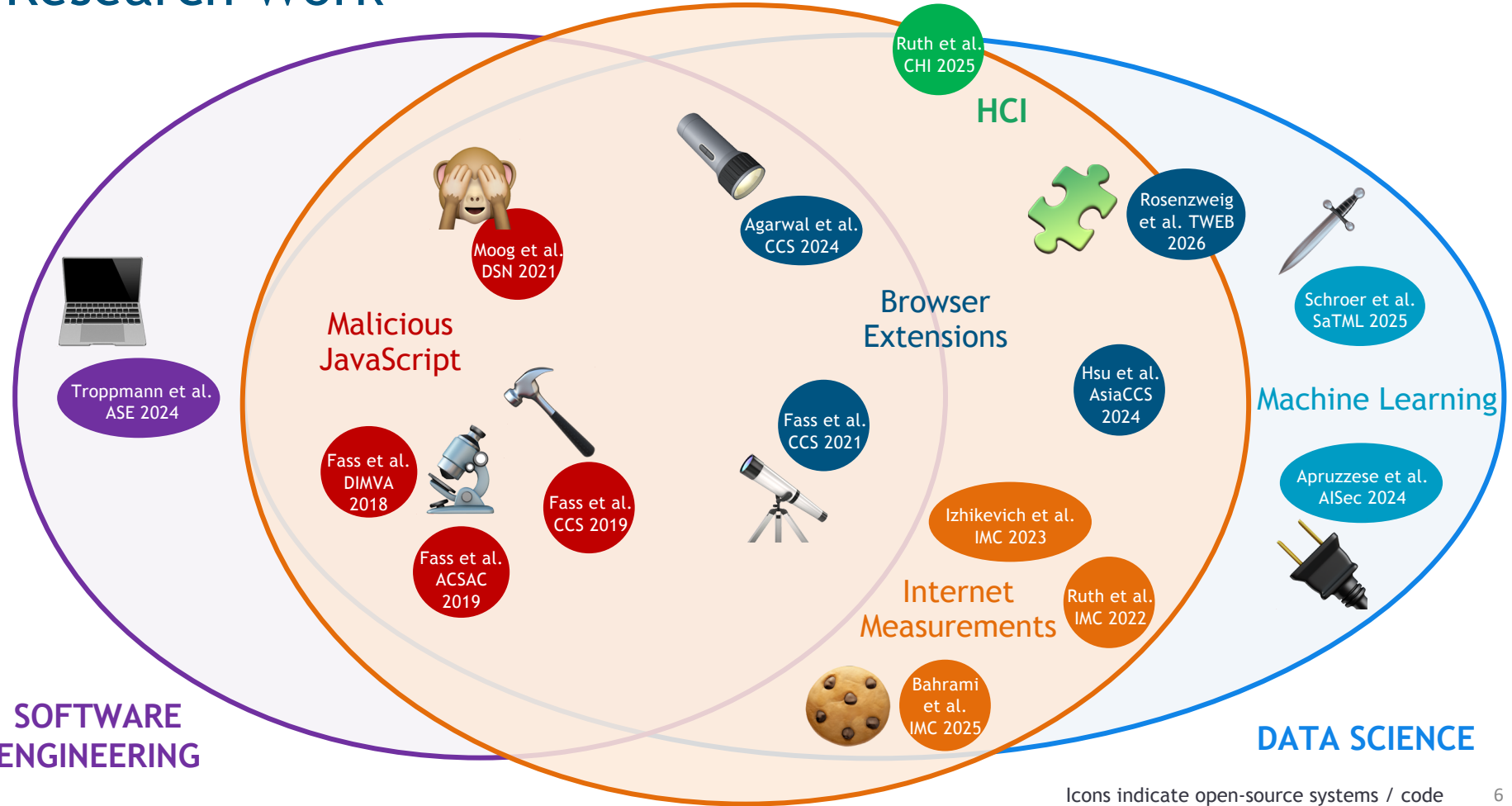
Fass et al.
CCS 2021



Hsu et al.
AsiaCCS 2024

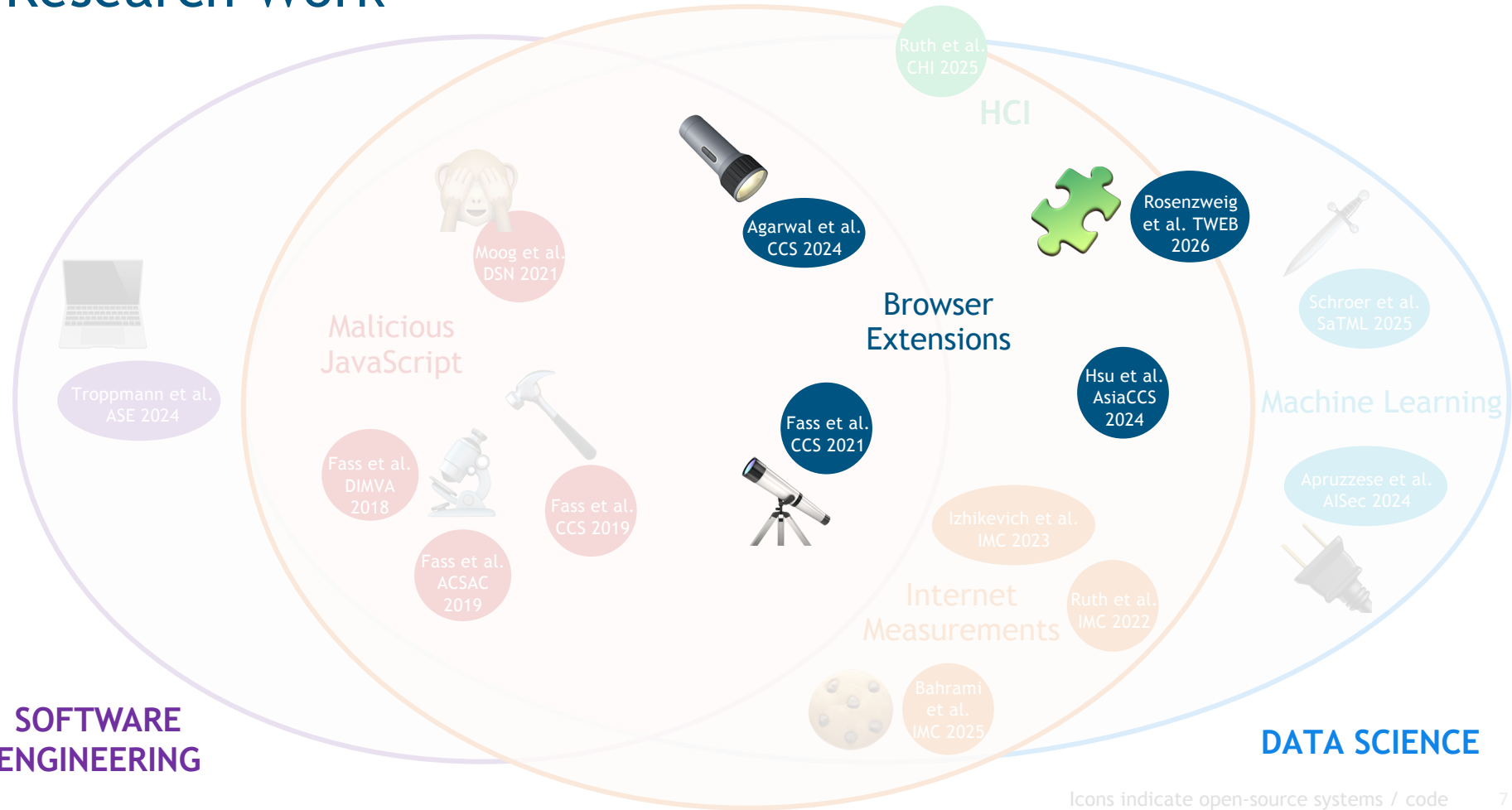
Research Work

WEB SECURITY & PRIVACY



Research Work

WEB SECURITY & PRIVACY



SOFTWARE ENGINEERING

DATA SCIENCE

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- Detecting Vulnerable Extensions
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- Detecting Malicious Extensions
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations



Hsu et al.
AsiaCCS
2024

Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- Detecting Vulnerable Extensions
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- Detecting Malicious Extensions
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations



Hsu et al.
AsiaCCS
2024

Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

What are Browser Extensions?

- Third-party programs to **improve user browsing experience**



Adblock Plus - free ad blocker

Offered by: adblockplus.org



Skype

Offered by: www.skype.com



Adobe Acrobat

Offered by: Adobe Inc.



Grammarly for Chrome

Offered by: grammarly.com



Honey

Offered by: <https://www.joinhoney.com>



Google Translate

Offered by: translate.google.com



LastPass: Free Password Manager

Offered by: LastPass



Cisco Webex Extension

Offered by: webex.com

- **~250k Chrome extensions** totaling almost **2B active users**

➤ Necessitates a thorough **security and privacy assessment**

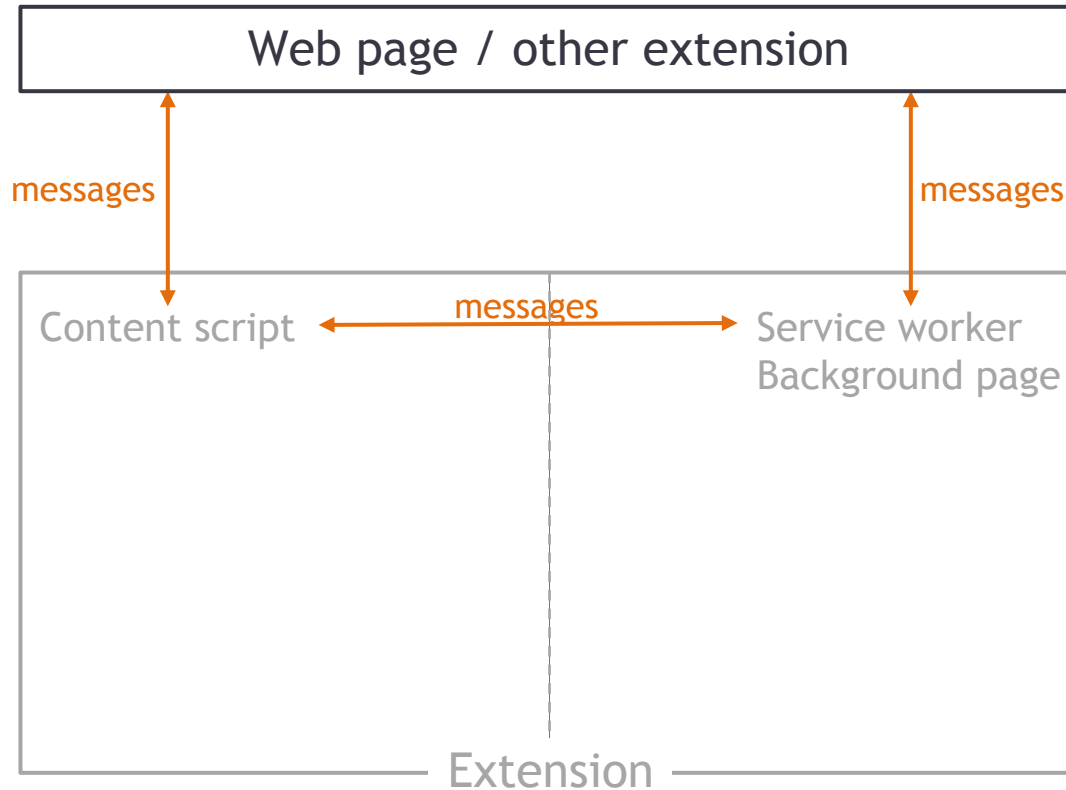
Background – manifest.json

- Every extension needs a manifest written in JSON, called `manifest.json`, which gives essential information, e.g.,
 - Extension's name, version, and manifest's version
 - Main components of an extension (CS, BP/SW, ...)
 - Permissions of an extension (downloads, history, ...)
 - ...

Background – Extension Architecture

- Service worker (SW in MV3) / Background page (BP in old MV2):
 - Core logic of an extension
 - Executed independently of the lifetime of a tab / window
 - Privileged part of an extension
- Content scripts (CS):
 - Injected by an extension into (a) web page(s)
 - Can use standard DOM APIs to read / modify a web page
 - Similar to scripts directly loaded by a web page + some more privileges
 - Restricted access to extension APIs

Background – Extension Architecture & Messages



Background – Authorized APIs & Permissions

- Extensions only have access to:
 - APIs explicitly declared in the `manifest.json`, e.g.,
 - `storage` - store/access data from the *extension storage*
 - `downloads` - download files
 - `history` - access to a user's browsing history
 - `bookmarks`, `cookies`, `topSites`, ...
 - `host` declared in the `manifest.json` = web pages an extension can access (read/write), e.g., to do some *cross-origin* requests

- https://developer.chrome.com/docs/extensions/mv3/declare_permissions/

- <https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions>

Background – manifest.json -- example

```
{
  "name": "My Extension",
  "version": "versionString",
  "description": "A plain text description",
  "manifest_version": 3
  "permissions": ["downloads", "history"],
  "host_permissions": ["https://example.com/*"],
  "background": {
    "service_worker": ["service_worker.js"],
  },
  "content_scripts": [{
    "matches": ["<all_urls>"],
    "js": ["content_script.js"]
  }],
}
```

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- Detecting Vulnerable Extensions
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- Detecting Malicious Extensions
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations



Hsu et al.
AsiaCCS
2024

Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

How Secure are Browser Extensions?

- Browser extensions provide **additional functionality**...
- ... so browser extensions need **additional & elevated privileges** compared to web pages
 - e.g., ad-blockers need to modify web page content or intercept network requests
 - e.g., extensions can download arbitrary files and access cross-domain data; while web pages cannot (blocked by the Same-Origin Policy)
- **Browser extensions are an attractive target for attackers** 🤩

Dangerous Browser Extensions

→ Extensions can put their users' security & privacy at risk:

- **Contain malware:** designed by malicious actors to harm victims
 - E.g., propagate malware, steal users' credentials, track users [1, 3]
- **Contain vulnerabilities:** designed by well-intentioned developers... but buggy
 - E.g., can lead to user-sensitive data exfiltration [1, 2]
- **Violate the Chrome Web Store policies**
 - E.g., deceive users, promote unlawful activities, lack a privacy policy [1]
- **Be fingerprintable:** can be recognized and uniquely identified
 - E.g., can lead to user tracking or inferring of user personal information [4]

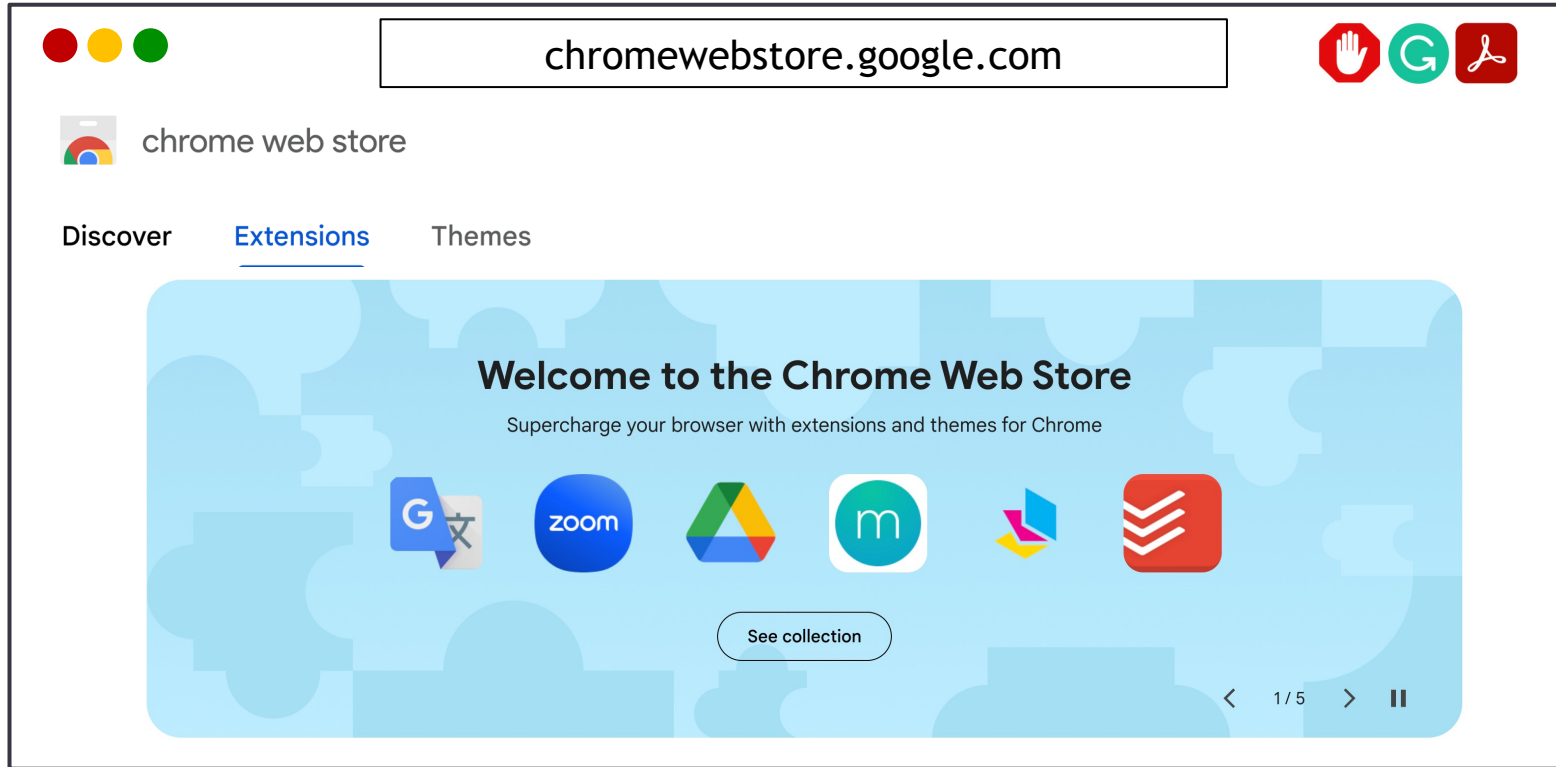
Dangerous Browser Extensions

→ Extensions can put their users' security & privacy at risk:


- **Contain malware:** designed by malicious actors to harm victims
 - E.g., propagate malware, steal users' credentials, track users [1, 3]
- **Contain vulnerabilities:** designed by well-intentioned developers... but buggy
 - E.g., can lead to user-sensitive data exfiltration [1, 2]
- **Violate the Chrome Web Store policies**
 - E.g., deceive users, promote unlawful activities, lack a privacy policy [1]
- **Be fingerprintable:** can be recognized and uniquely identified
 - E.g., can lead to user tracking or inferring of user personal information [4]

➤ Security-Noteworthy Extensions (SNE) [1]

How are Security-Noteworthy Extensions (SNE) Installed?



How are Security-Noteworthy Extensions (SNE) Installed?



The image shows a browser window at `chromewebstore.google.com`. The page title is "chrome web store" and the navigation menu includes "Discover", "Extensions" (which is underlined), and "Themes". A large blue banner is displayed with the following text:

Welcome to the Chrome Web Store
Subj...
>26k SNE
(in just 3 years, 2020–23)

Below the text is a "See collection" button. At the bottom right of the banner, there are navigation controls: a left arrow, "1/5", a right arrow, and a pause icon.

Main Findings on the Chrome Web Store

- **350M users** installed **Security-Noteworthy Extensions** in 2020–2023
- These **dangerous extensions** stay in the Chrome Web Store *for years*
- **60%** of extensions have **never received a single update**



> What is in the Chrome Web Store?

In *ACM AsiaCCS 2024*. Sheryl Hsu, Manda Tran, and Aurore Fass



Browser Extension Collection: Chrome-Stats

The screenshot shows the chrome-stats.com website. At the top, the URL 'chrome-stats.com' is displayed in a white box. Below this is a navigation sidebar on the left with a dark blue background, containing a 'Chrome-Stats' header and several menu items: 'Explore', 'Advanced Search', 'Data & API', 'Marketplace', 'App Pass', 'Extension Booster', and 'Pricing'. The main content area has a light blue header with the text 'Analyze your extensions & mobile apps' and a sub-header 'The data you need to grow — historical stats, downloads, keyword research, and competitor tracking.' Below this is a search bar with the placeholder text 'Search extensions / apps' and a magnifying glass icon. The main content area also features a section titled 'Extensive extension & app data' with the text 'We provide the most comprehensive database for browser extensions and mobile apps'. This section contains five cards, each representing a platform or browser: Chrome (259,562 Extensions, 71,446 Themes), Android (2,600,911 Apps, 298,222 Games), Apple (1,729,425 Apps, 12,592 Games), Edge (30,056 Add-ons, 729 Themes), and Firefox (65,244 Add-ons, 79,949 Themes).

chrome-stats.com

Chrome-Stats

- Explore
- Advanced Search
- Data & API
- Marketplace
- App Pass
- Extension Booster
- Pricing

Analyze your extensions & mobile apps

The data you need to grow — historical stats, downloads, keyword research, and competitor tracking.

Search extensions / apps

Extensive extension & app data

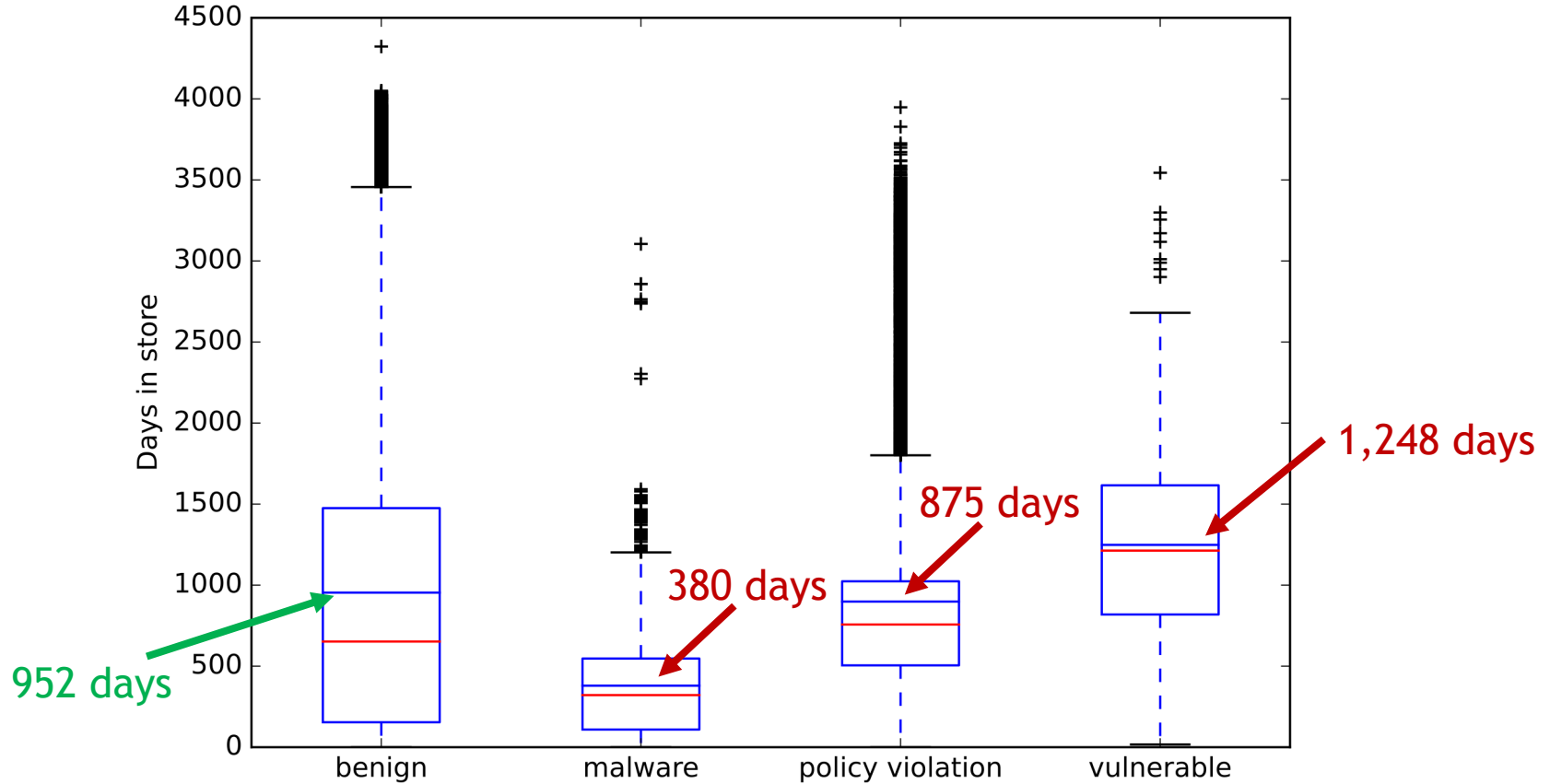
We provide the most comprehensive database for browser extensions and mobile apps

 Chrome 259 562 Extensions 71 446 Themes	 Android 2 600 911 Apps 298 222 Games	 Apple 1 729 425 Apps 12 592 Games
 Edge 30 056 Add-ons 729 Themes	 Firefox 65 244 Add-ons 79 949 Themes	

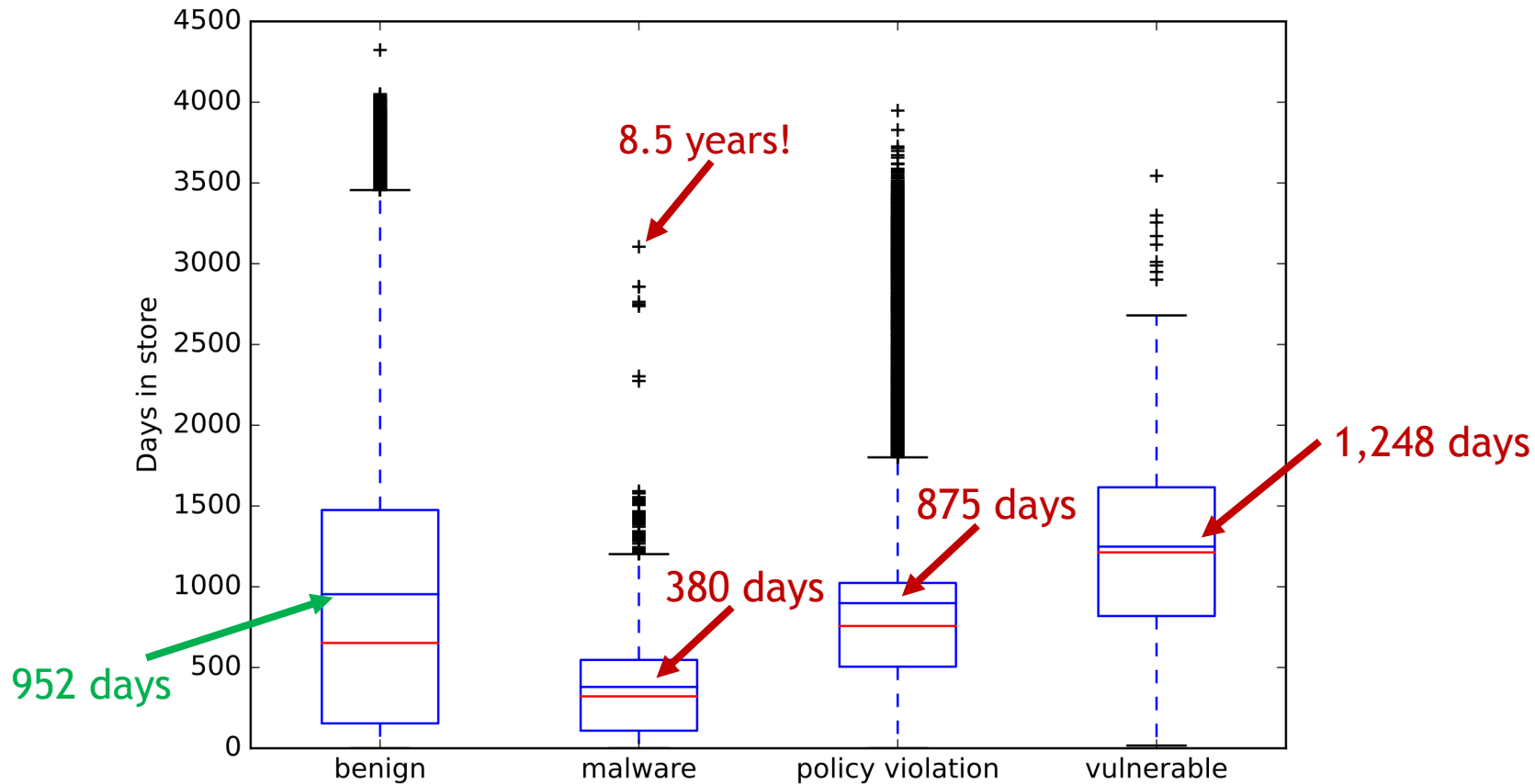
Browser Extension Collection: Chrome-Stats

Category	#Extensions Metadata collected	#Extensions Code collected	When collected
SNE	26,014	16,377	Before May 1, 2023
- Malware-containing	10,426	6,587	Before May 1, 2023
- Policy-violating	15,404	9,638	Before May 1, 2023
- Vulnerable [2]	184	152	March 16, 2021
“Benign” extensions	226,762	92,482	Before May 1, 2023

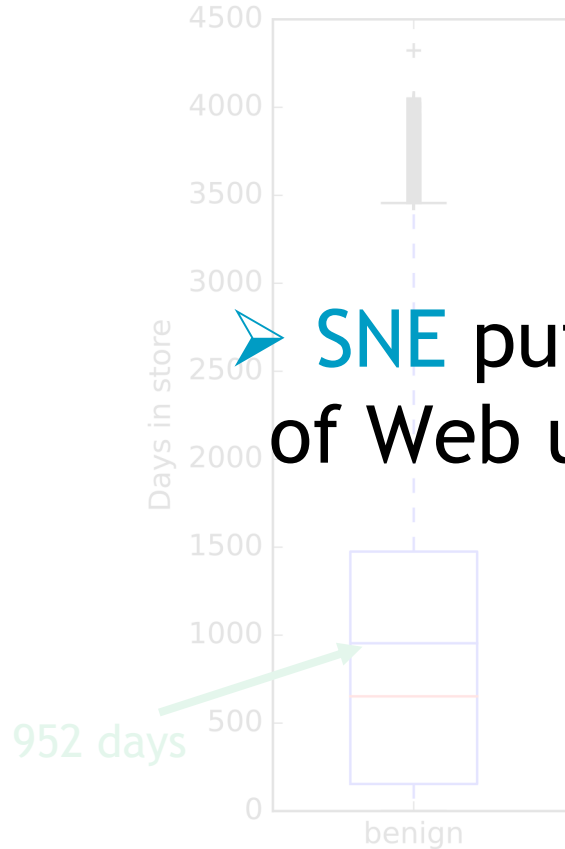
Number of Days in the CWS



Number of Days in the CWS



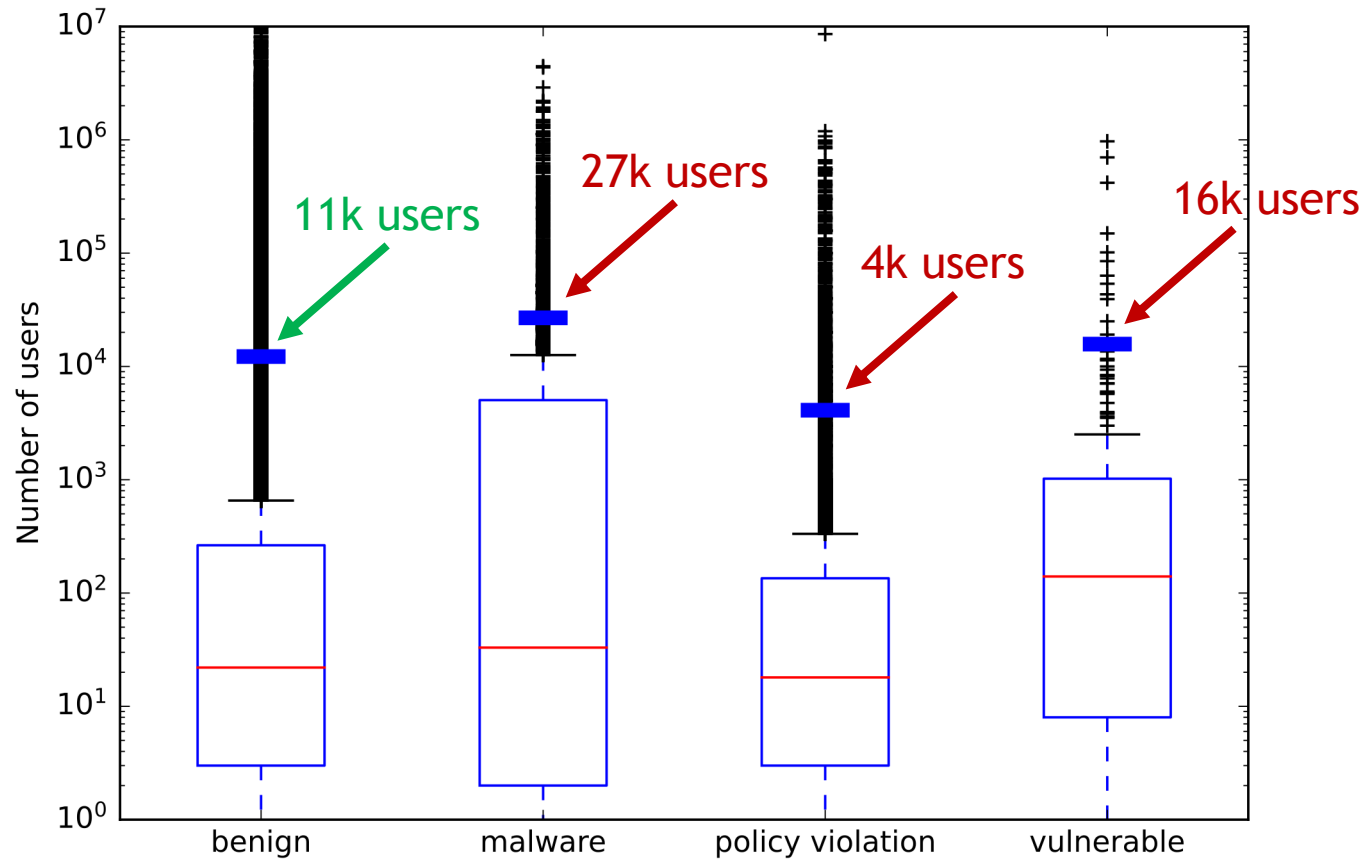
Number of Days in the CWS



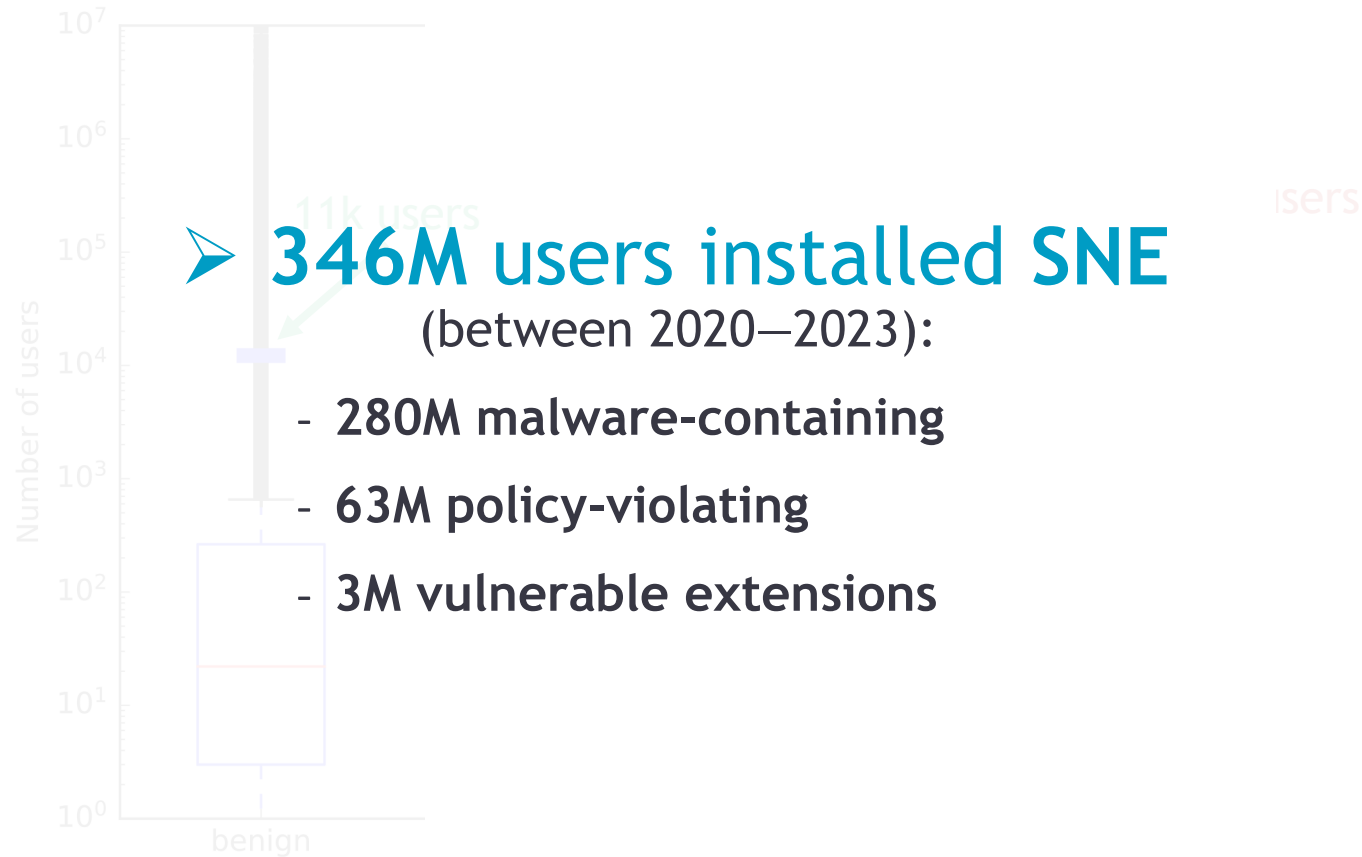
➤ SNE put the security & privacy of Web users *at risk for years*

,248 days

Number of Users



Number of Users



Media Coverage

Forbes

FORBES > INNOVATION > CYBERSECURITY

280 Million Google Chrome Users Installed Dangerous Extensions, Study Says

Davey Winder Senior Contributor @
Davey Winder is a veteran cybersecurity writer, hacker and analyst.

[Follow](#)

Jun 24, 2024, 06:57am EDT



How safe are Google Chrome extensions? SOPA IMAGES/LIGHTROCKET VIA GETTY IMAGES

The Register

Risk of installing dodgy extensions from Chrome store way worse than Google's letting on, study suggests

All depends on how you count it – Chocolate Factory claims 1% fail rate

[Thomas Claburn](#)

Sun 23 Jun 2024 // 10:36 UTC

ADGUARD

Subscribe to news  Search blog 

AdGuard > Blog > Google is failing miserably at weeding out bad extensions, new research indicates

Google is failing miserably at weeding out bad extensions, new research indicates

July 5, 2024 · 7 min read

TECHSPOT


TRENDING FEATURES REVIEWS THE BEST DOWNLOADS PRODUCT FINDER FORUMS

SECURITY THE WEB MALWARE CHROME

Researchers say 280 million people have installed malware-infected Chrome extensions in the last 3 years

Google claims less than 1% of all installs include malware

By Rob Thubron June 24, 2024 at 11:39 AM



Media Coverage

Risk of installing dodgy extensions from Chrome store way worse than Google's letting on, study suggests

This review process weeds out the overwhelming majority of bad extensions before they even get published. In 2024, less than 1% of all installs from the Chrome Web Store were found to include malware. We're proud of this record and yet some bad extensions still get through, which is why we also monitor published extensions.

<https://security.googleblog.com/2024/06/staying-safe-with-chrome-extensions.html>

Google is failing miserably at weeding out bad extensions, new research indicates

July 6, 2024 - 7 min read



Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- **Detecting Vulnerable Extensions**
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- Detecting Malicious Extensions
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Hsu et al.
AsiaCCS
2024



Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



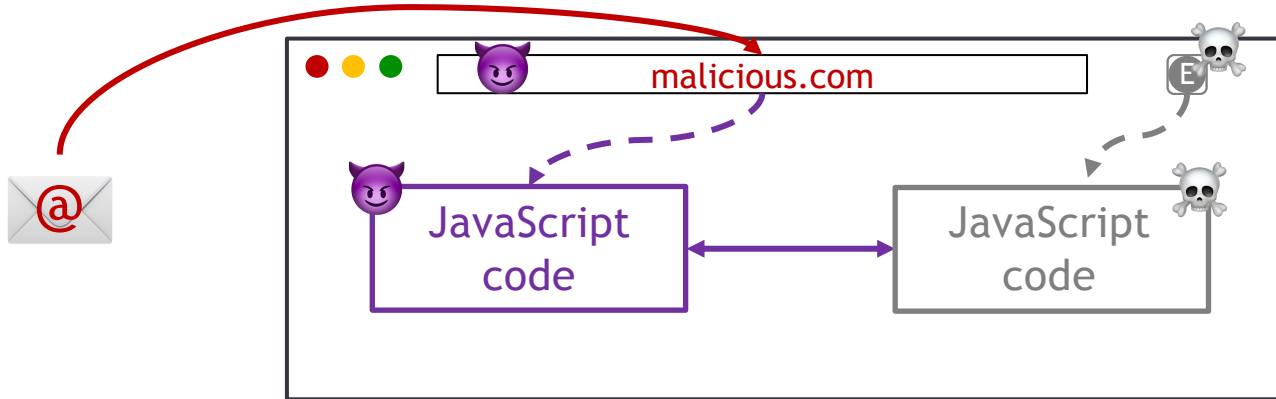
Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



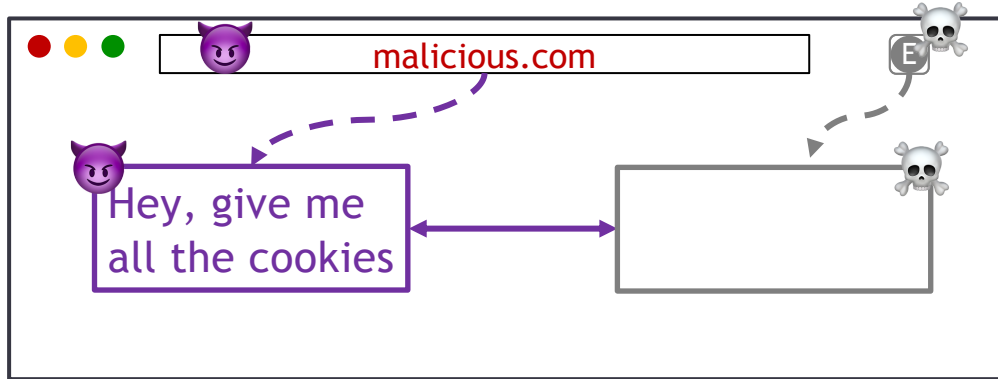
Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



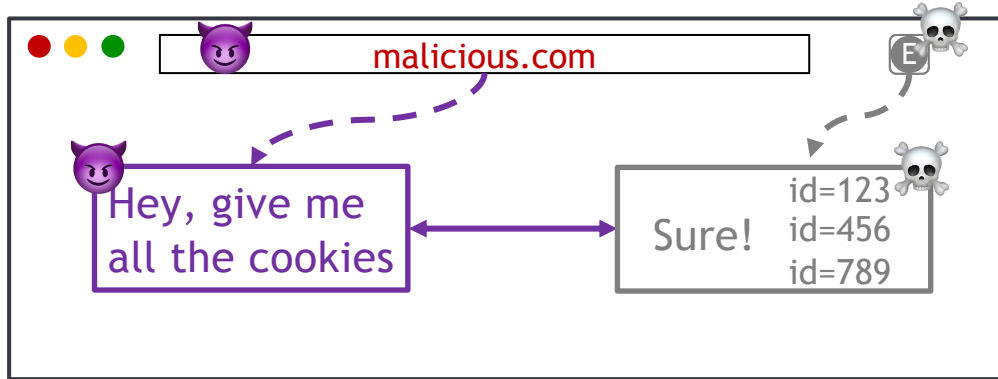
Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



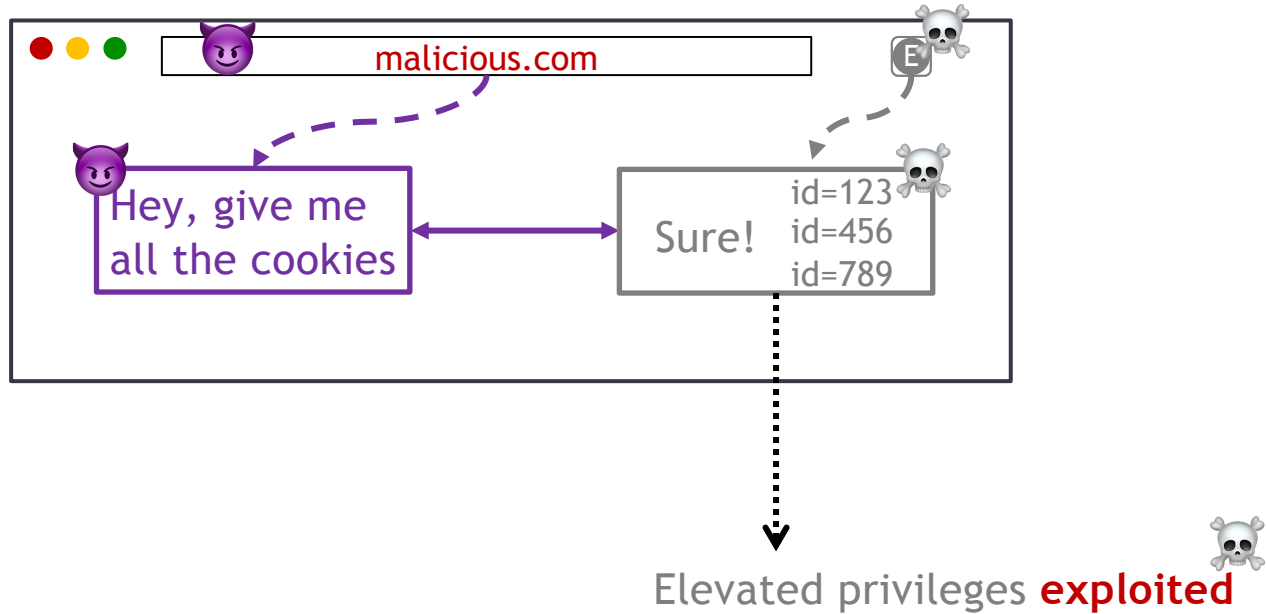
Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



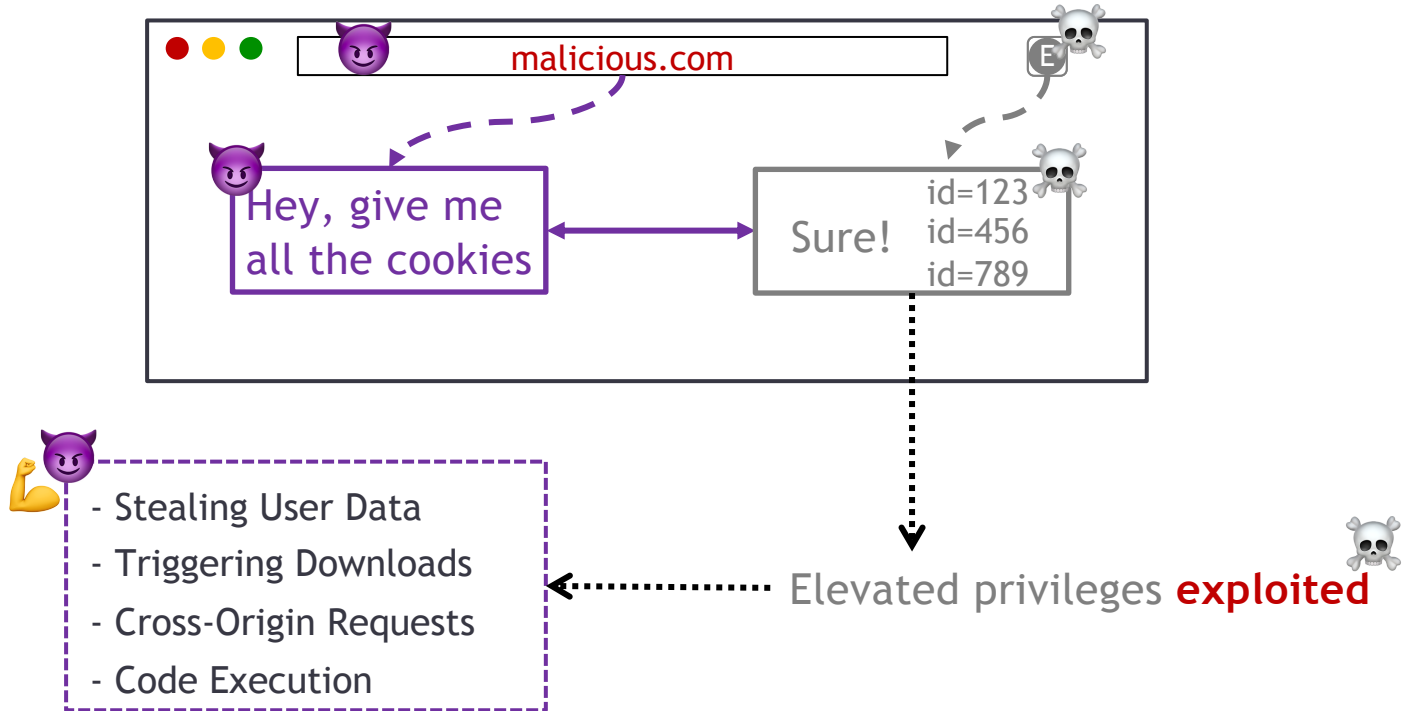
Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



Analysis of Vulnerable Extensions: Threat Model

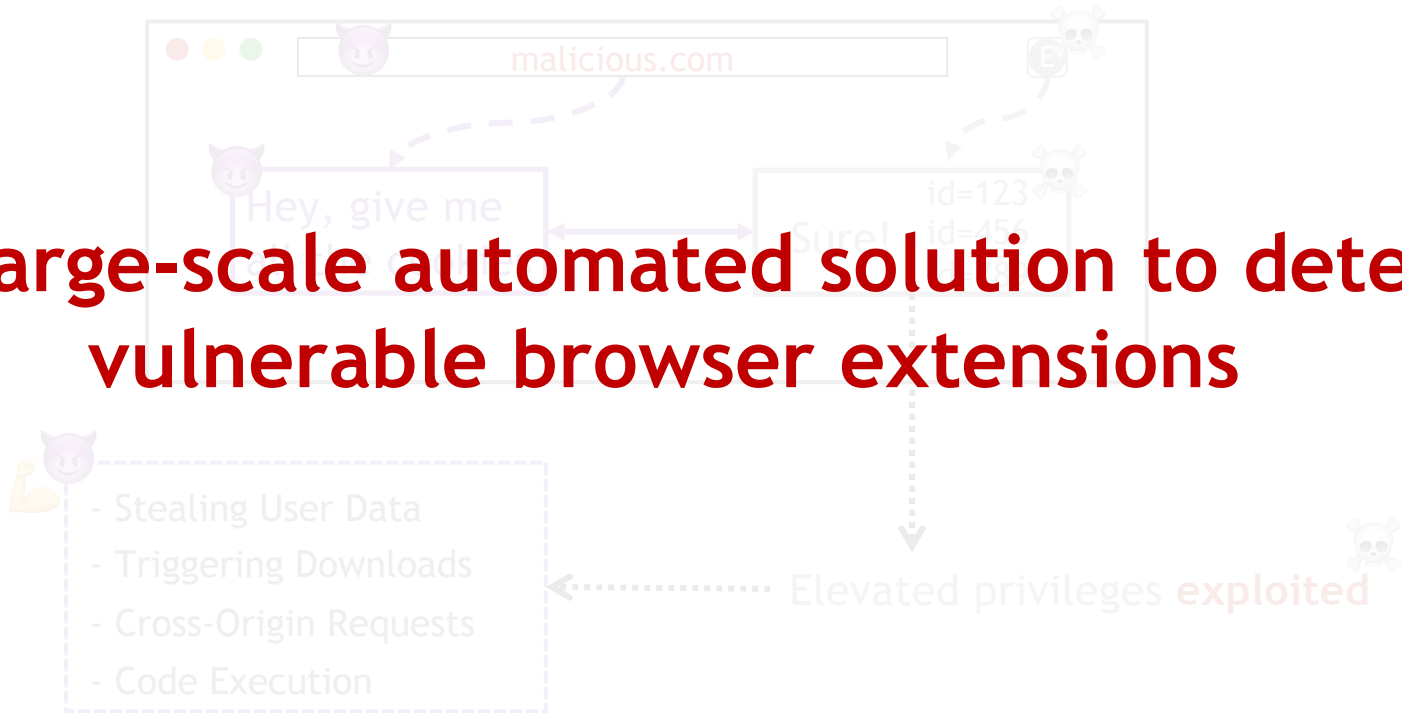
Challenging to detect due to their inherently benign intent (*benign-but-buggy*)



Analysis of Vulnerable Extensions: Threat Model

Challenging to detect due to their inherently benign intent (*benign-but-buggy*)

No large-scale automated solution to detect vulnerable browser extensions



Detecting Vulnerable Extensions



> **DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions**
In ACM CCS 2021. Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
2021, ACM CCS

Abstract
Browser extensions are a popular way to enhance browser functionality. However, they are often vulnerable to attacks, such as data leakage or unauthorized access to sensitive information. This paper presents DOUBLEX, a static analysis tool designed to detect vulnerable data flows in browser extensions. DOUBLEX analyzes the code of browser extensions to identify potential security vulnerabilities, such as insecure data storage, insecure network communication, and unauthorized access to sensitive information. DOUBLEX is designed to be scalable and efficient, allowing it to analyze a large number of browser extensions. DOUBLEX is evaluated on a large dataset of browser extensions, showing that it is able to detect a wide range of vulnerabilities. DOUBLEX is a practical tool for identifying and fixing security vulnerabilities in browser extensions.

CCS Concepts
Security and privacy → Web applications security; Browser security

Keywords
Browser extensions; Static analysis; Security vulnerabilities; Data flows

Detecting Vulnerable Extensions



DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale
Aurore Fass, Dolie Francis Somé, Michael Backes, and Ben Stock
1 Introduction

Abstract
Browser extensions are a popular way to enhance the functionality of web browsers. However, they are also a common source of security vulnerabilities. In this paper, we present DOUBLEX, a static analysis tool that detects vulnerable data flows in browser extensions. DOUBLEX is designed to be scalable and accurate, and is able to detect a wide range of vulnerabilities, including data leaks, data tampering, and data manipulation. We evaluate DOUBLEX on a large dataset of browser extensions and show that it is able to detect a high number of vulnerabilities, including several that were not detected by other tools. We also show that DOUBLEX is able to detect vulnerabilities in extensions that are not covered by other tools.

CCS keywords
Software security, Browser extensions, Static analysis, Vulnerability detection

> **DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions**
In ACM CCS 2021. Aurore Fass, Dolie Francis Somé, Michael Backes, and Ben Stock



Detecting Vulnerable Extensions



DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CCS 2021

Abstract
Browser extensions are a popular way to enhance the functionality of web browsers. However, they are also a common source of security vulnerabilities. In this paper, we present DOUBLEX, a static analysis tool that detects vulnerable data flows in browser extensions. DOUBLEX is designed to be scalable and accurate, and is able to detect a wide range of vulnerabilities, including data leaks, data tampering, and data manipulation. We evaluate DOUBLEX on a large dataset of browser extensions and show that it is able to detect a high number of vulnerabilities. We also discuss the implications of our findings and provide recommendations for browser extension developers.

1 Introduction
Browser extensions are a popular way to enhance the functionality of web browsers. However, they are also a common source of security vulnerabilities. In this paper, we present DOUBLEX, a static analysis tool that detects vulnerable data flows in browser extensions. DOUBLEX is designed to be scalable and accurate, and is able to detect a wide range of vulnerabilities, including data leaks, data tampering, and data manipulation. We evaluate DOUBLEX on a large dataset of browser extensions and show that it is able to detect a high number of vulnerabilities. We also discuss the implications of our findings and provide recommendations for browser extension developers.

CCS keywords
Browser extensions, static analysis, data flows, security vulnerabilities.

Keywords
Browser extensions, static analysis, data flows, security vulnerabilities.

> **DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions**

In ACM CCS 2021. Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

 Malicious web page



Detecting Vulnerable Extensions



DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CCS 2021

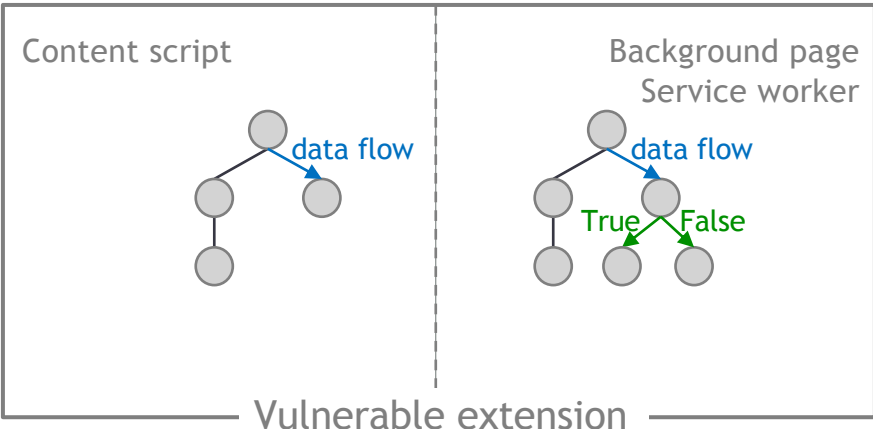
Abstract
Browser extensions are a popular way to enhance the functionality of web browsers. However, they are also a common source of security vulnerabilities. In this paper, we present DOUBLEX, a static analysis tool that detects vulnerable data flows in browser extensions. DOUBLEX is based on a novel abstraction of JavaScript code that allows for the analysis of data flows across different components of an extension. We evaluate DOUBLEX on a large dataset of browser extensions and show that it can detect a wide range of vulnerabilities, including data leaks and unauthorized access to sensitive information. Our results show that DOUBLEX is able to detect vulnerabilities in a significant number of extensions, and that it is able to detect vulnerabilities that were not detected by existing tools. We conclude that DOUBLEX is a valuable tool for detecting vulnerabilities in browser extensions and that it can help to improve the security of browser extensions.

1 Introduction
Browser extensions are a popular way to enhance the functionality of web browsers. However, they are also a common source of security vulnerabilities. In this paper, we present DOUBLEX, a static analysis tool that detects vulnerable data flows in browser extensions. DOUBLEX is based on a novel abstraction of JavaScript code that allows for the analysis of data flows across different components of an extension. We evaluate DOUBLEX on a large dataset of browser extensions and show that it can detect a wide range of vulnerabilities, including data leaks and unauthorized access to sensitive information. Our results show that DOUBLEX is able to detect vulnerabilities in a significant number of extensions, and that it is able to detect vulnerabilities that were not detected by existing tools. We conclude that DOUBLEX is a valuable tool for detecting vulnerabilities in browser extensions and that it can help to improve the security of browser extensions.

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

In ACM CCS 2021. Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

 Malicious web page



Per-component JavaScript code abstraction

- AST (Abstract Syntax Tree)
- Control flow
- Data flow
- Pointer analysis

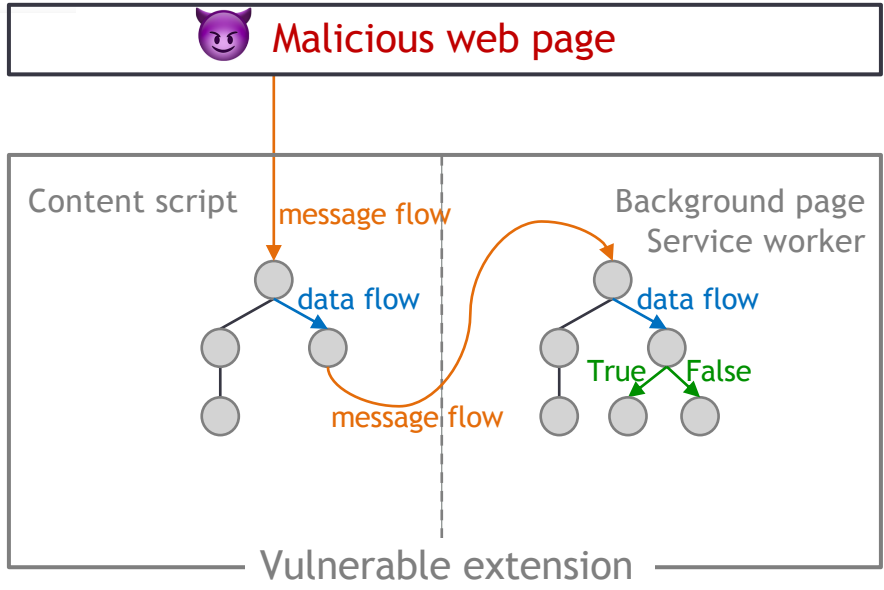
Detecting Vulnerable Extensions



DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CCS 2021

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

In ACM CCS 2021. Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock



Per-component JavaScript code abstraction

- AST (Abstract Syntax Tree)
- Control flow
- Data flow
- Pointer analysis

Extension Dependence Graph (EDG)

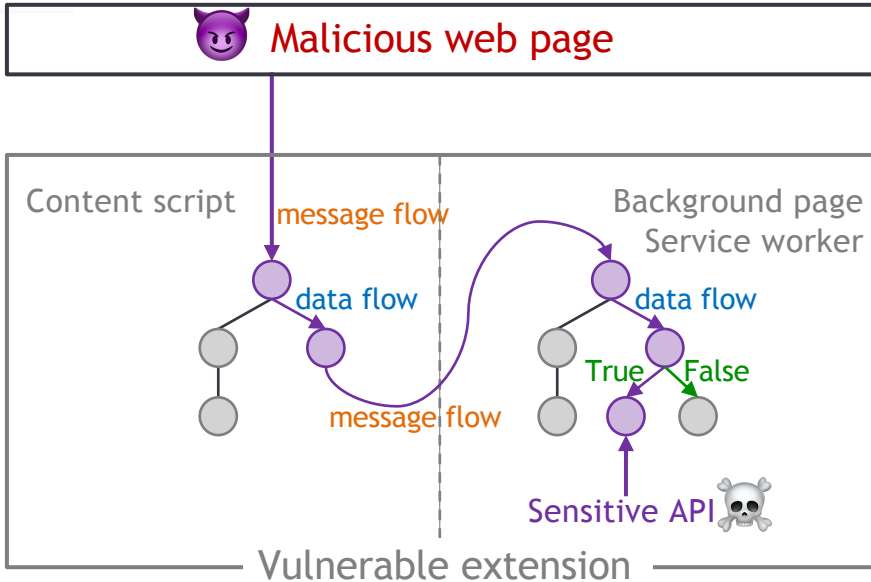
- > Message interactions

Detecting Vulnerable Extensions



> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

In ACM CCS 2021. Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock



Per-component JavaScript code abstraction

- AST (Abstract Syntax Tree)
- Control flow
- Data flow
- Pointer analysis

Extension Dependence Graph (EDG)

- > Message interactions

Suspicious data flow tracking

- > Detects any path between an attacker & sensitive APIs

Simplified Example of a Vulnerability

```
1 // Content script code = from the extension
2 window.addEventListener("message", function(event) {
3
4
5
6 })
```

Simplified Example of a Vulnerability

```
1 // Content script code = from the extension
2 window.addEventListener("message", function(event) {
3
4
5
6 })
```

message received



Simplified Example of a Vulnerability

```
1 // Content script code = from the extension
2 window.addEventListener("message", function(event) {
3
4     eval(event.data);
5
6 })
```

message received



Simplified Example of a Vulnerability

```
1 // Content script code = from the extension
2 window.addEventListener("message", function(event) {
3
4     eval(event.data);
5
6 })
```

message received



```
// DOUBLEX report
{"direct-danger1": "eval",
"value": "eval(event.data)",
"line": "4 - 4",
"dataflow": true,
"param1": {
"received": "event",
"line": "2 - 2"}}}
```

Simplified Example of a Vulnerability

```
// Content script code = from the extension
window.addEventListener("message", function(event) {

    eval(event.data);

})
```

```
// Attacker code = from the targeted web page
postMessage("alert(1)", "*")
```

(very) malicious payload

Simplified Example of a Vulnerability

```
// Content script code = from the extension  
window.addEventListener("message", function(event) {  
  
    eval(event.data);  
  
})
```

developer.chrome.com indique

1

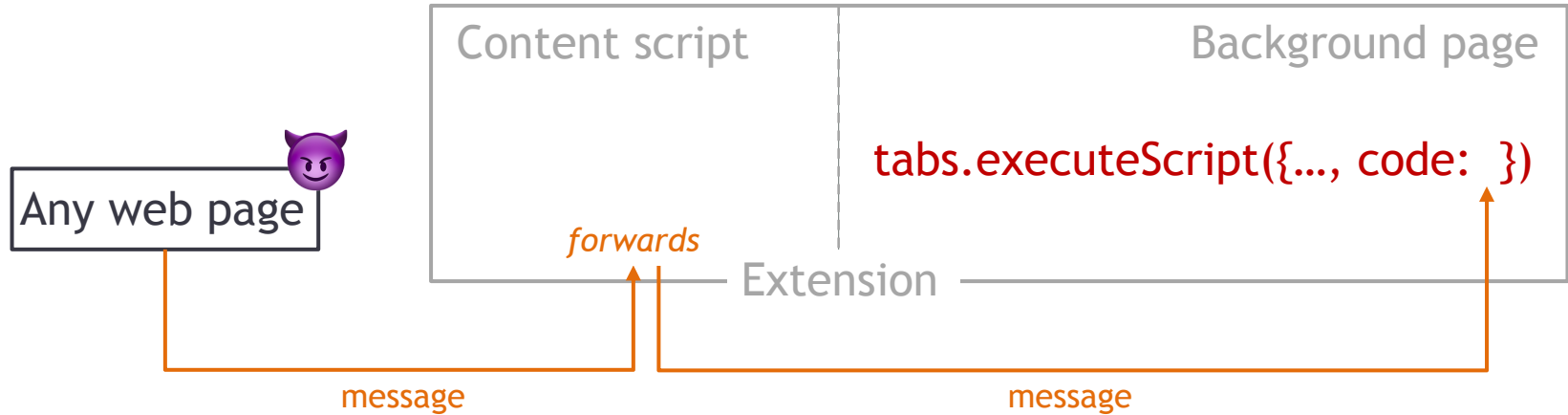
OK

```
// Attacker code = from the targeted web page  
postMessage("alert(1)", "*")
```

(very) malicious payload

Case Study of Vulnerable Chrome Extensions

Arbitrary code execution (*cdi...*, 4k+ users):



Detecting Vulnerable Extensions with DOUBLEX


Analyzed 155k Chrome extensions from 2021 with DOUBLEX (takes 2–3s per extension)

- **184 vulnerable Chrome extensions**
- **Impacting 3M users*** (* based on Google's definition)
- **Precision: 89%** of the flagged extensions are vulnerable
- **Recall: 93%** of known vulnerabilities [Somé, S&P 2019] are detected

- **Open source**, for developers and Web users

(even in other fields, e.g., mini apps [Wang et al., ICSE 2023])

 Fork 11   Star 82 

 Aurore54F/DoubleX

Defenses & Perspectives

- (Migrate an extension to Manifest V3)
- Know that communication with external actors may be dangerous
- Only allow communication with specified extensions or web pages
- Limit code execution by sanitizing messages
- DOUBLEX could provide a feedback channel for developers

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- Detecting Vulnerable Extensions
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- **Detecting Malicious Extensions**
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Hsu et al.
AsiaCCS
2024



Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

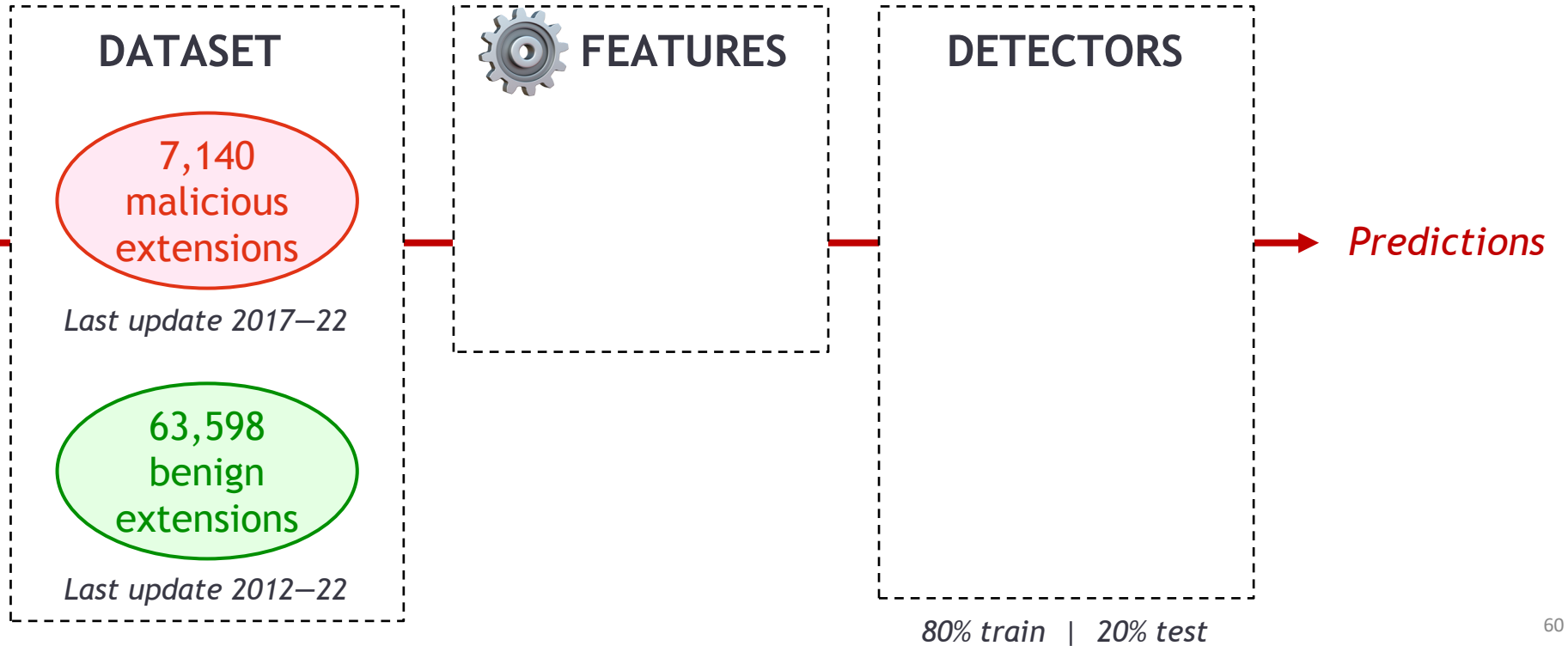
Detecting Malicious Extensions – Lab Setting



It's not Easy: Applying Supervised Machine Learning to Detect Malicious Extensions in the Chrome Web Store
BEN ROSENZWEIG, *University of Cambridge*, *University of Luxembourg*
VALENTINO DALLA VALLE, *University of Luxembourg*
GIOVANNI APRUZZESE, *University of Luxembourg*, *University of York*
AURORE FASS, *University of Luxembourg*

> It's not Easy: Applying Supervised ML to Detect Malicious Extensions in the CWS

In ACM TWEB 2026. Ben Rosenzweig, Valentino Dalla Valle, Giovanni Apruzzese, and Aurore Fass



Detecting Malicious Extensions – Lab Setting



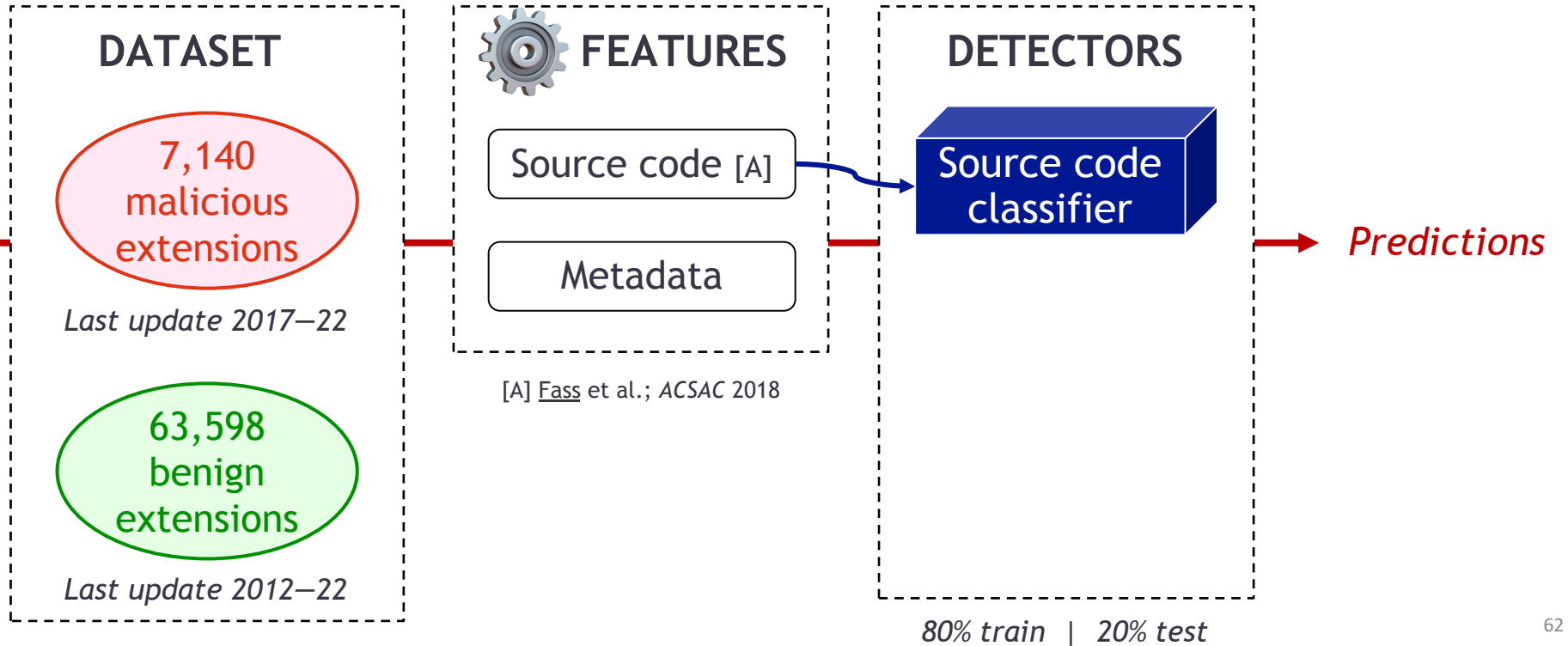
Rosenzweig
et al. TWEB
2026



> It's not Easy: Applying Supervised ML to Detect Malicious Extensions in the CWS

In ACM TWEB 2026. Ben Rosenzweig, Valentino Dalla Valle, Giovanni Apruzzese, and Aurore Fass

It's not Easy: Applying Supervised Machine Learning to Detect Malicious Extensions in the Chrome Web Store
BEN ROSENZWEIG, Lockheed Martin, Internet
VALENTINO DALLA VALLE, CNRS, Inria, Université de Bordeaux, Université de Lille
GIORGIO APRUZZESE, University of Luxembourg, Luxembourg and Yorkstadt University, Ireland
AURORE FASS, CNRS, Inria, Université de Bordeaux, France
Google Chrome is the most popular Web browser. Users can customize it with extensions that enhance their browsing experience. The market will have approximately 1.7 billion Chrome Web Store (CWS) Extensions by 2026. Unfortunately, this market is also a source of malware. In this paper, we study the problem of detecting malicious extensions in the CWS. We propose a supervised machine learning approach to detect malicious extensions in the CWS. We evaluate our approach on a dataset of 71,140 malicious extensions and 63,598 benign extensions. We compare our approach to existing state-of-the-art methods. We show that our approach outperforms existing methods. We also discuss the challenges of this task and the need for more research in this area.



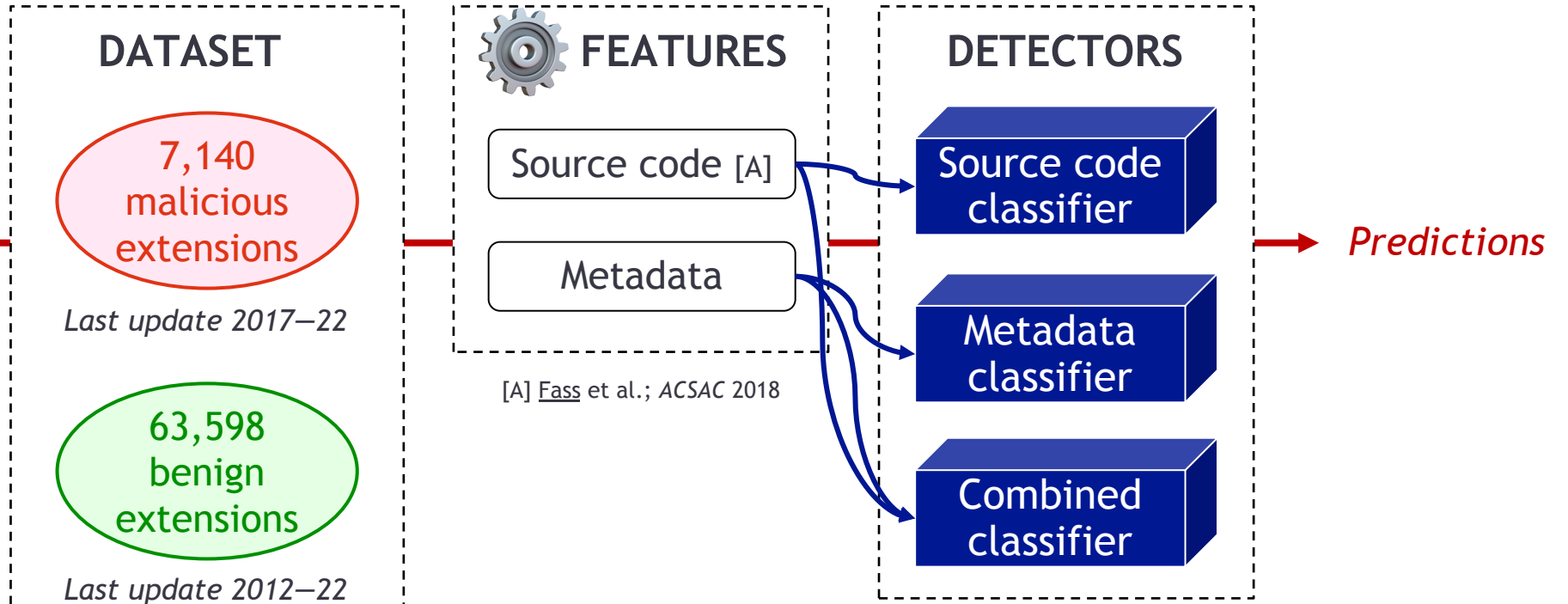
Detecting Malicious Extensions – Lab Setting



It's not Easy: Applying Supervised Machine Learning to Detect Malicious Extensions in the Chrome Web Store
BEN ROSENZWEIG, Lockheed Martin, Amazon
VALENTINO DALLA VALLE, CNR, University of Padua, University of Ferrara
GIUANNI APRUZZESE, University of L'Aquila, University of Padua, University of Ferrara
AUREO FASS, CNR, University of Padua, University of Ferrara

> It's not Easy: Applying Supervised ML to Detect Malicious Extensions in the CWS

In ACM TWEB 2026. Ben Rosenzweig, Valentino Dalla Valle, Giovanni Apruzzese, and Aurore Fass



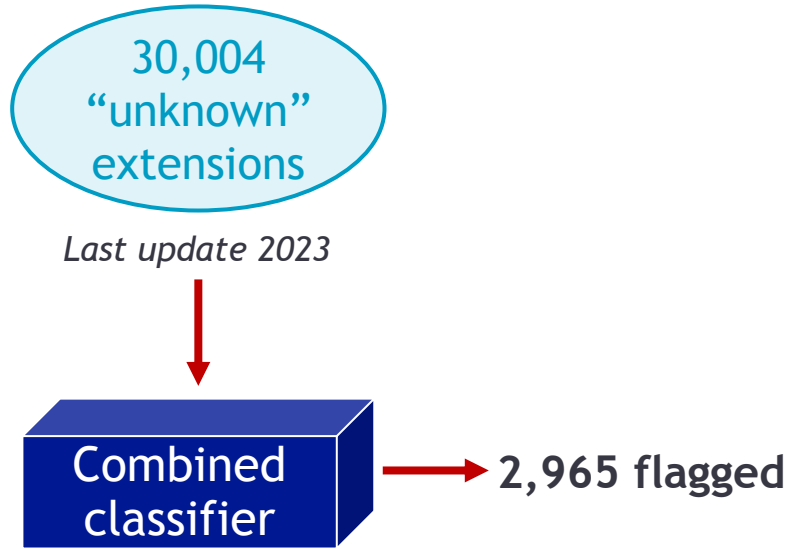
Detecting Malicious Extensions – Real World

30,004
“unknown”
extensions

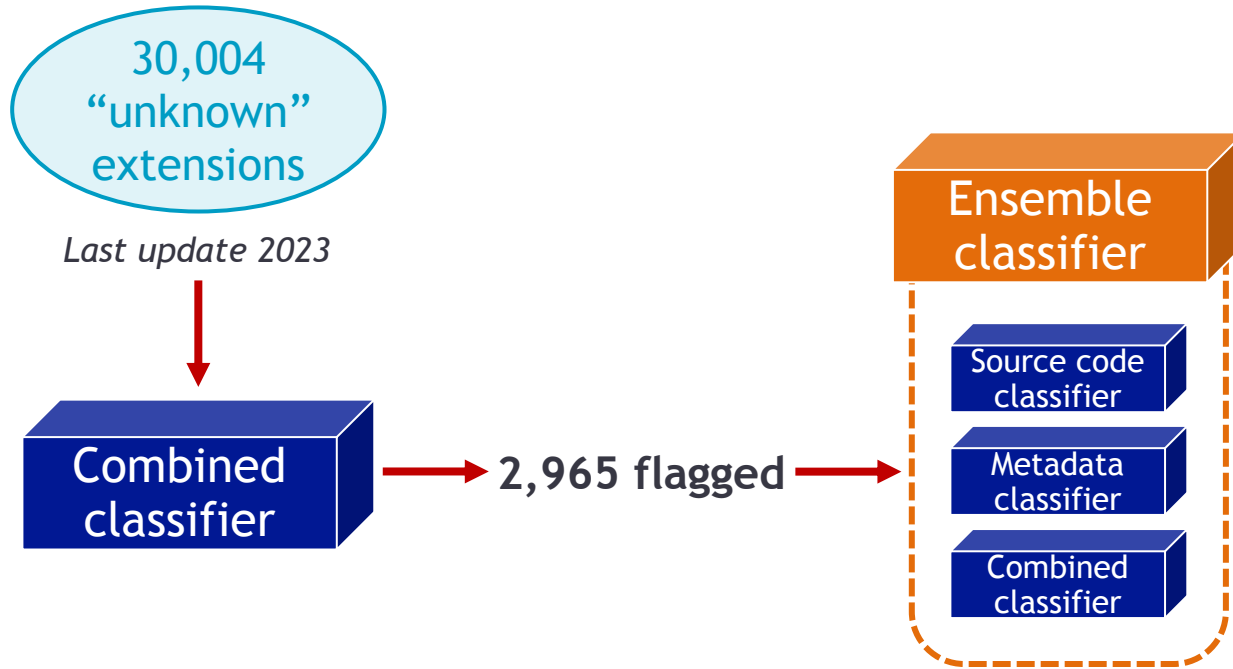
Last update 2023



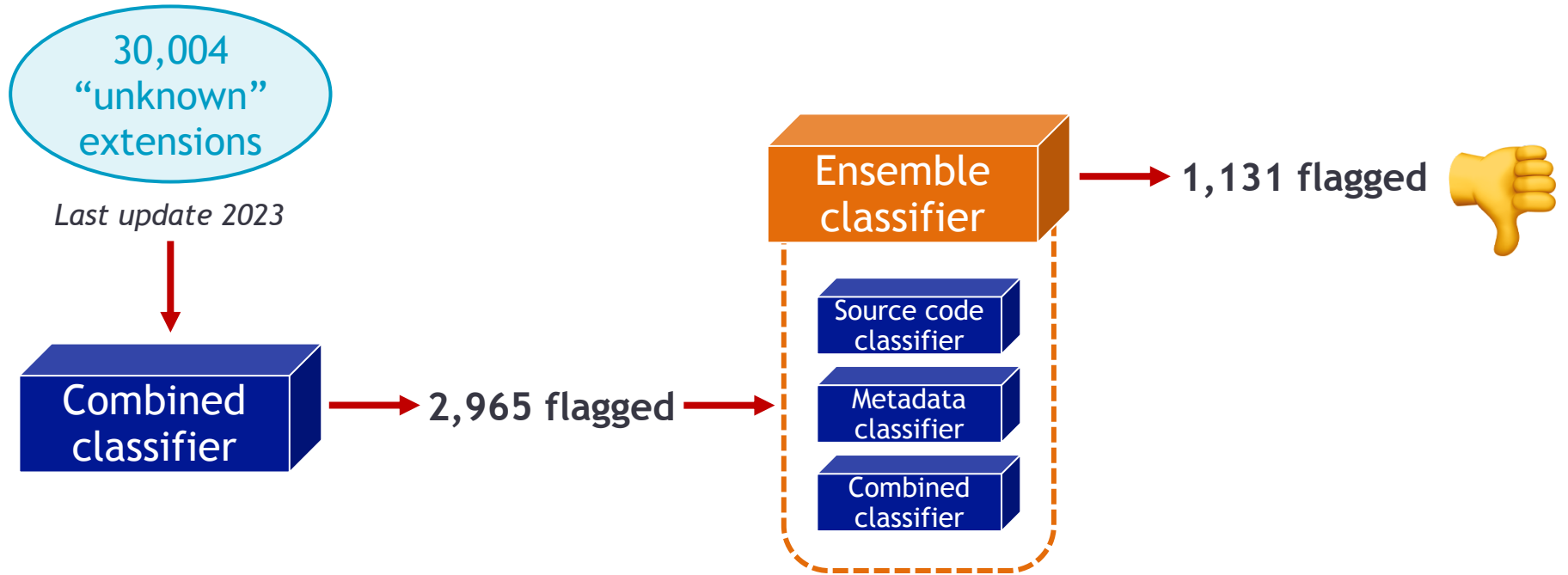
Detecting Malicious Extensions – Real World



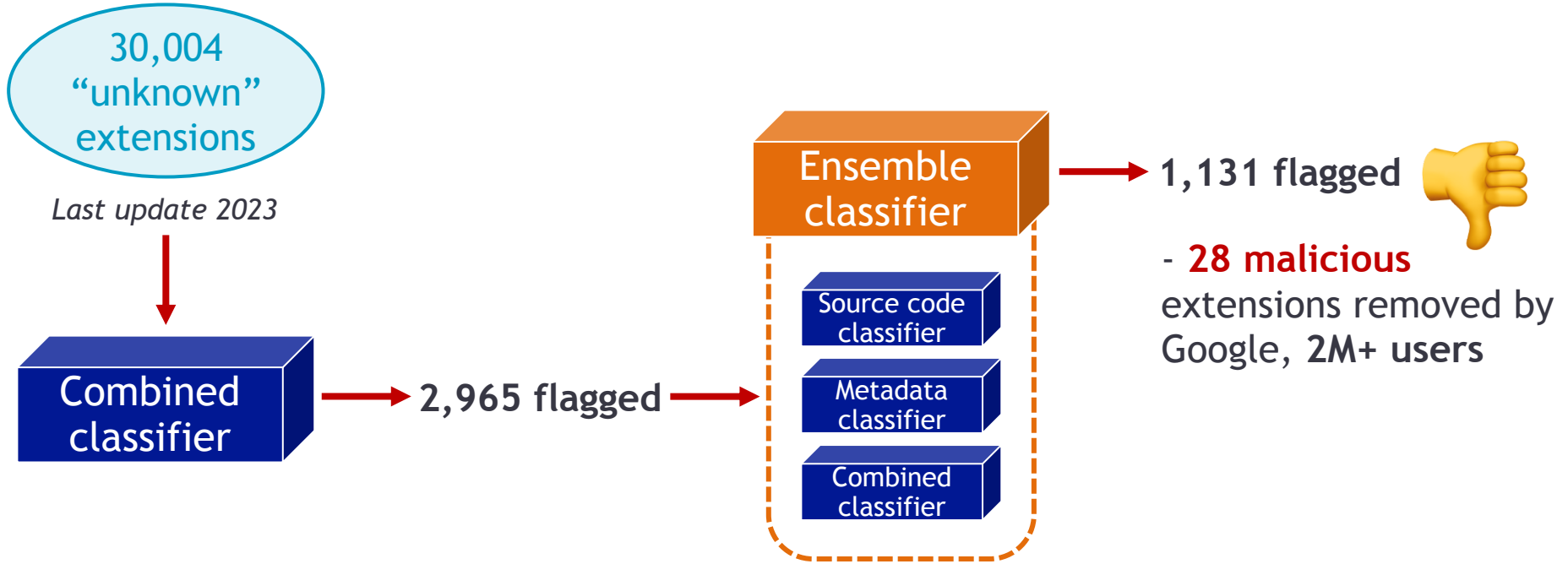
Detecting Malicious Extensions – Real World



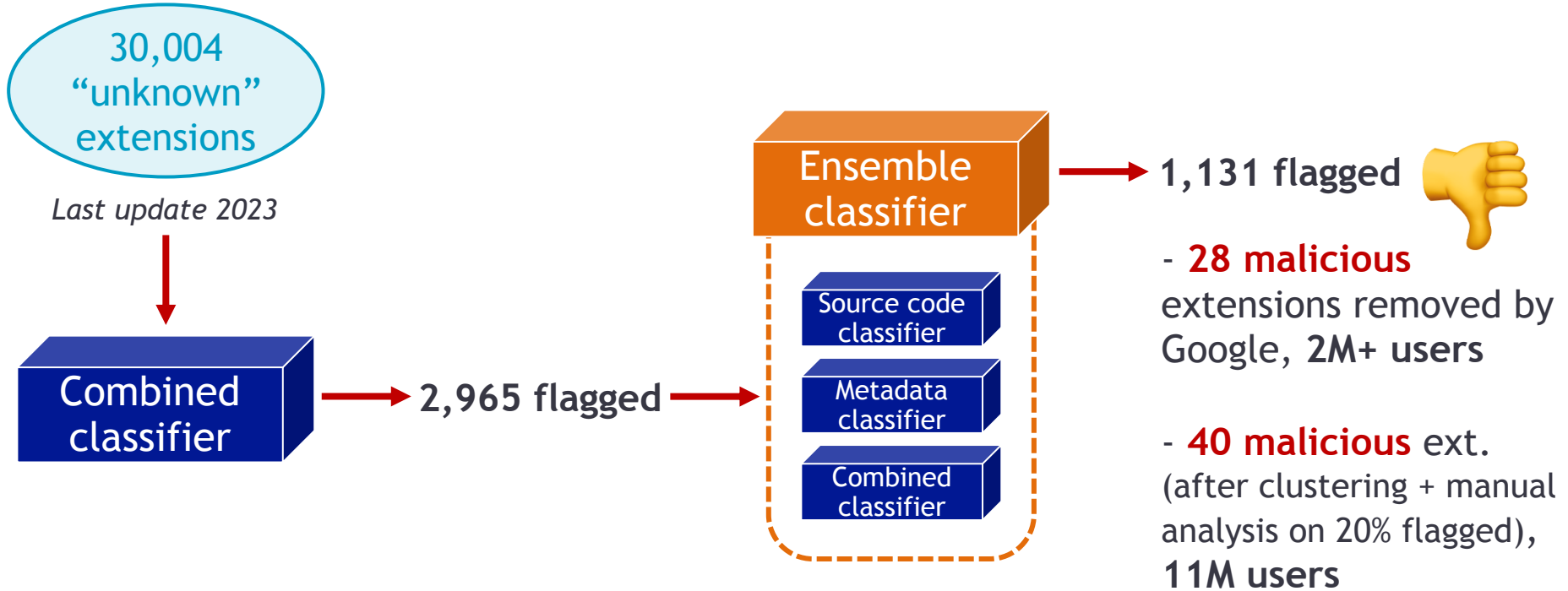
Detecting Malicious Extensions – Real World



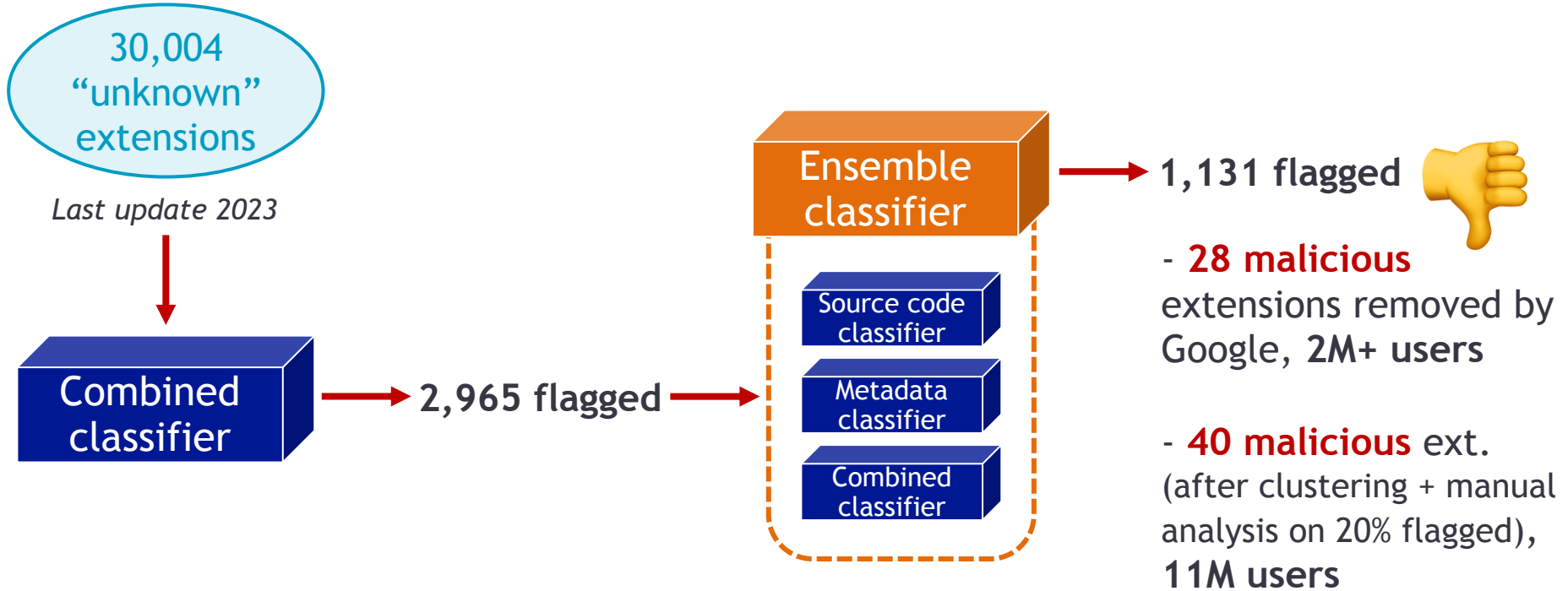
Detecting Malicious Extensions – Real World



Detecting Malicious Extensions – Real World

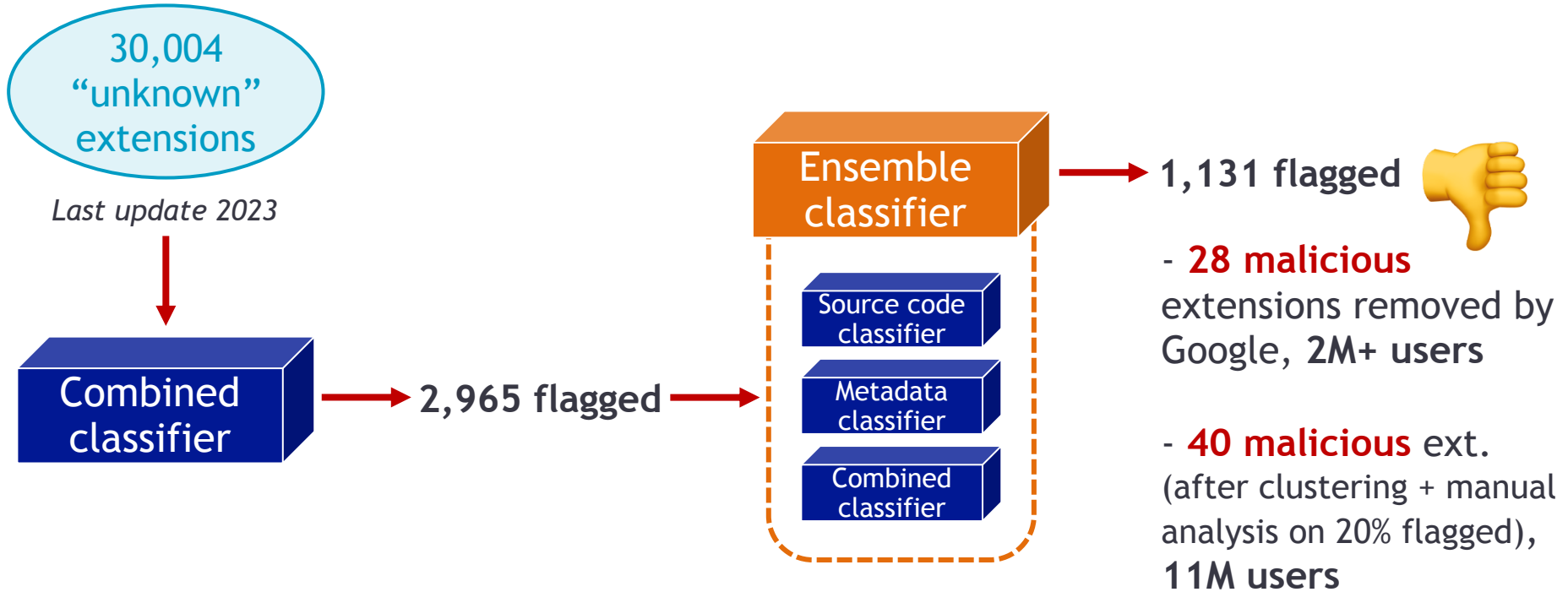


Detecting Malicious Extensions – Real World



Total: 68 malicious extensions, 13M users

Detecting Malicious Extensions – Real World



May 2024: disclosure of the 40 extensions to Google, no response

As of Sept. 2025: 17 / 40 malicious extensions were taken down

Total: 68 malicious extensions, 13M users

Detecting Malicious Extensions – Lab Setting vs. Real World

- **Lab setting:** 98.37% accuracy
- **Real world:** 1,131 / 30,004 extensions flagged | 68 verified malicious extensions

???

Detecting Malicious Extensions – Lab Setting vs. Real World

- **Lab setting:** 98.37% accuracy
- **Real world:** 1,131 / 30,004 extensions flagged | 68 verified malicious extensions

???

- **Concept drift**
 - **Intrinsic evolution** of extensions, hard for supervised ML tools to keep a (high) accuracy over time
 - **How** to validate concept drift? → see experiments in the paper
 - **Why** are extensions affected? → see paper (study feature importance)

Detecting Malicious Extensions – Takeaways

- ML evaluations can be **misleading**: detectors performing well in a lab setting are **no guarantee of practical applicability** in the real world
- Detectors need to be **retrained regularly**
- Investigate **active learning** as a **mitigation to concept drift**
 - E.g., *uncertainty sampling*: monthly retraining of the detector on X labeled extensions where the detector is the most uncertain

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
- Detecting Vulnerable Extensions
 - Threat model and automated tool (DOUBLEX)
 - Case studies, results, and potential defense strategies
- Detecting Malicious Extensions
 - Lab setting vs. real world
- Detecting Fingerprintable Extensions
 - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Hsu et al.
AsiaCCS
2024



Fass et al.
CCS 2021



Rosenzweig
et al. TWEB
2026



Agarwal et al.
CCS 2024

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

Browser Extension Fingerprinting

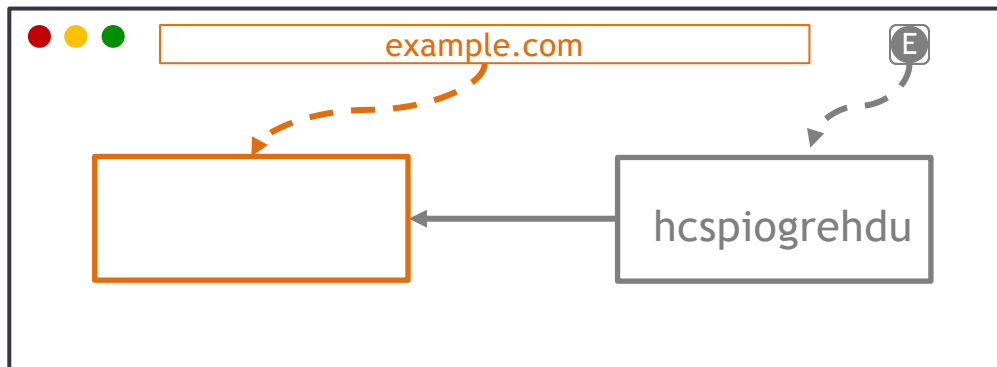
Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages



Browser Extension Fingerprinting

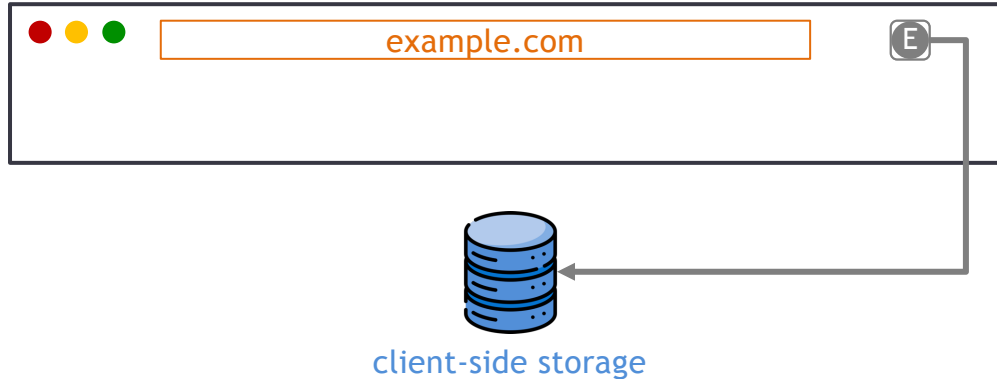
Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

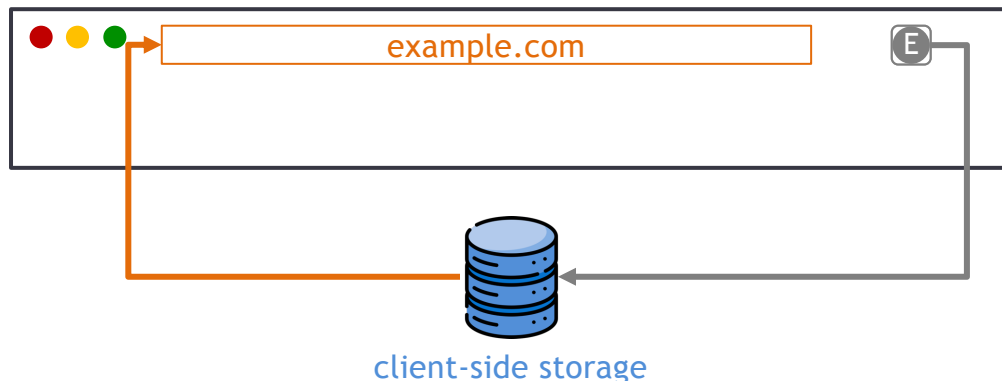
- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)



Browser Extension Fingerprinting

Browser extensions can interact with web pages...

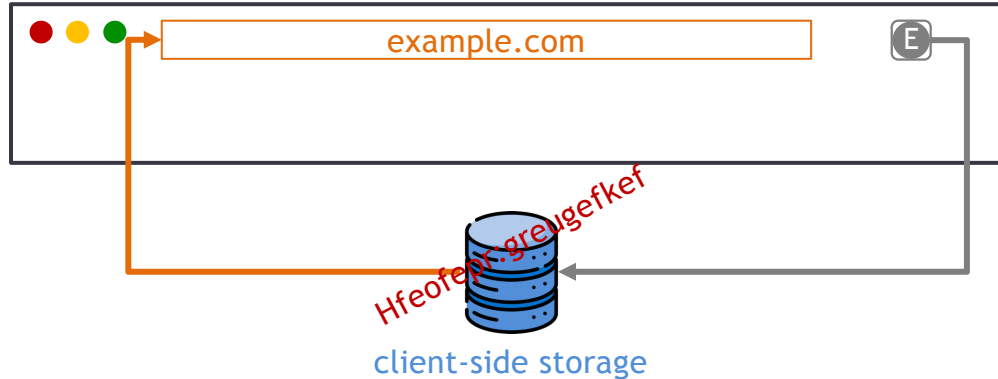
- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)



Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)



Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

... which leaves traces → **observable side effects**, can be seen by a “malicious” site

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

... which leaves traces → **observable side effects**, can be seen by a “malicious” site

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

... which leaves traces → **observable side effects**, can be seen by a “malicious” site

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

... which leaves traces → **observable side effects**, can be seen by a “malicious” site

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

Why is this bad?

- 1) Extensions send Postmessage to web pages
- 2) Extensions store data on the client side through storage APIs (e.g., cookies, local/session storage, indexedDB)
- 3) Extensions inject JavaScript code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

... which leaves traces → observable side effects, can be seen by a “malicious” site

Browser Extension Fingerprinting

Browser extensions can interact with web pages...

Why is this bad?

- 1) Extensions send Postmessage to web pages
- 2) Extensions store data on the client side through storage APIs
 - “Malicious” websites can track users by fingerprinting their extensions
- 3) Extensions inject JavaScript code directly into web pages
 - Extensions can reveal personal user information, e.g., geolocation, ethnicity, social/personal interests, medical issues, religion, etc. [Karami; NDSS'19]

... which leaves traces → observable side effects, can be seen by a “malicious” site

Detecting Fingerprintable Extensions

*How many extensions can be **uniquely fingerprinted** through these observable side effects?*

Detecting Fingerprintable Extensions

How many extensions can be uniquely fingerprinted through these observable side effects?



> Peeking through the window: Fingerprinting Browser Extensions through Page-Visible Execution Traces and Interactions

In ACM CCS 2024. Shubham Agarwal, Aurore Fass, and Ben Stock




Detecting Fingerprintable Extensions with Raider

Analyzed 38k Chrome extensions from 2024 with Raider

- **2,747 fingerprintable Chrome extensions** (lower bound)
- Impacting **169M users**

- **Notified 1,967 developers** about their fingerprintable extension(s)
 - Only 30 (!) replied
 - Of those, only 16 positively acknowledged the issues
 - But: they heavily **rely on our fingerprinting vectors** (e.g., script injection or data storage) **for their extensions' functionality**

- Raider PoC is **available online**  Raider-ext/raider

Mitigations

- Global APIs:
 - ensure that browser extension code runs before the attacker code (inject at `document_start`)
 - ensure that APIs cannot be overwritten (freeze their native definition)
- Global variables: scope appropriately
- Storage: use the `chrome.storage` API instead

Summary

Dangerous Browser Extensions

→ Extensions can put their users' security & privacy at risk:

- Contain **malware**: designed by malicious actors to harm victims
 - E.g., propagate malware, steal users' credentials, track users
- Contain **vulnerabilities**: designed by well-intentioned developers... but buggy
 - E.g., can lead to user-sensitive data exfiltration
- Violate the Chrome Web Store **policies**
 - E.g., deceive users, promote unlawful activities, lack a privacy policy
- Be **fingerprintable**: can be recognized and uniquely identified
 - E.g., can lead to user tracking or inferring of user personal information

Hsu et al.
AsiaCCS
2024

Detecting Malicious Extensions – Lab Setting vs. Real World

- **Lab setting**: 98.37% accuracy
- **Real world**: 1,131 / 30,004 extensions flagged | 68 verified malicious extensions



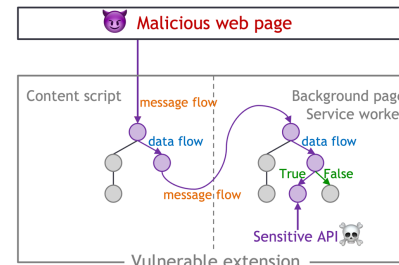
Rosenzweig
et al. TWEB
2026

its-not-easy/tweb25

▪ Concept drift

- **Intrinsic evolution** of extensions, hard for supervised ML tools to keep a (high) accuracy over time
- **How to validate concept drift?** → see experiments in the paper
- **Why are extensions affected?** → see paper (study feature importance)

Detecting Vulnerable Extensions with DOUBLEX



Fass et al.
CCS 2021



Aurore54F/DoubleX

➤ DOUBLEX detects suspicious data flows in browser extensions

184 vulnerable extensions | Precision: 89% | Recall: 93%

Detecting Fingerprintable Extensions with Raider

- 1) Extensions send **PostMessage** to web pages
- 2) Extensions store data on the client side through **storage APIs**
(e.g., cookies, local/session storage, IndexedDB)
- 3) Extensions **inject JavaScript** code directly into web pages
 - registering global variables
 - invocation of global APIs and properties

Agarwal et al.
CCS 2024



Raider-ext/raider

➤ Raider detects 2,747 fingerprintable extensions | 169M users



Corresponding Publications

- [What is in the Chrome Web Store?](#)

Sheryl Hsu, Manda Tran, and [Aurore Fass](#). In *ACM AsiaCCS 2024*

- [DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale](#)

[Aurore Fass](#), Dolière Francis Somé, Michael Backes, and Ben Stock. In *ACM CCS 2021*

- [It's not Easy: Applying Supervised Machine Learning to Detect Malicious Extensions in the Chrome Web Store](#)

Ben Rosenzweig, Valentino Dalla Valle, Giovanni Apruzzese, and [Aurore Fass](#). In *ACM TWEB 2026*

- [Peeking through the window: Fingerprinting Browser Extensions through Page-Visible Execution Traces and Interactions](#)

Shubham Agarwal, [Aurore Fass](#), and Ben Stock. In *ACM CCS 2024*