CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

On the Security and Privacy Risks of Browser Extensions

Dr.-Ing. Aurore Fass

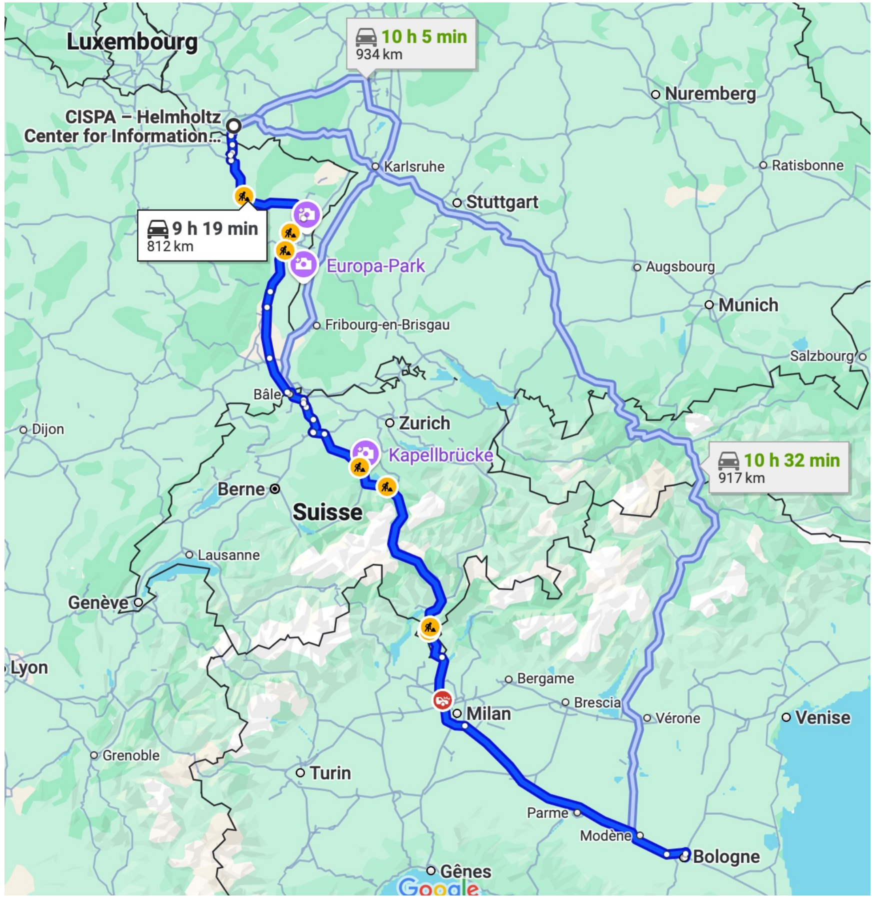
Tenure-Track Faculty at CISPA — Helmholtz Center for Information Security

University of Bologna, March 18th, 2025

Dr.-Ing. Aurore (/ɔʀɔʀ/) FASS

- 🇫🇷 Graduated from TELECOM Nancy (FR, 2017)
- 🇩🇪 PhD Student + Postdoc at CISPA (DE, 2017—21)
- 🇺🇸 Visiting Assistant Professor at Stanford (US, 2021—23)
- 🇩🇪 Tenure-Track Faculty at CISPA (DE, 2023—)

CISPA – Helmholtz Center for Information Security



Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - SNE (comparative) analysis
- Detecting Vulnerable Extensions
  - Threat model & example
  - Case studies, results, and potential defense strategies
- Detecting Fingerprintable Extensions
  - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Outline

- Background: Browser Extensions

- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - SNE (comparative) analysis

- Detecting Vulnerable Extensions
  - Threat model & example
  - Case studies, results, and potential defense strategies

- Detecting Fingerprintable Extensions
  - Presentation of 3 fingerprinting vectors, results, and potential mitigations

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Background — What are Browser Extensions?

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

AdBlock — best ad blocker
Offered by: getadblock.com

Adblock Plus - free ad blocker
Offered by: adblockplus.org

Adobe Acrobat
Offered by: Adobe Inc.

Avast Online Security
Offered by: https://www.avast.com

Cisco Webex Extension
Offered by: webex.com

Google Translate
Offered by: translate.google.com

Grammarly for Chrome
Offered by: grammarly.com

Honey
Offered by: https://www.joinhoney.com

Pinterest Save Button
Offered by: pinterest.com

Skype
Offered by: www.skype.com

uBlock Origin
Offered by: Raymond Hill (gorhill)

LastPass: Free Password Manager
Offered by: LastPass

- *Bundles* of JavaScript, HTML, or CSS files, defined in a `manifest.json`

- ~**145k** Chrome extensions totaling over **1.6B** active users

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/

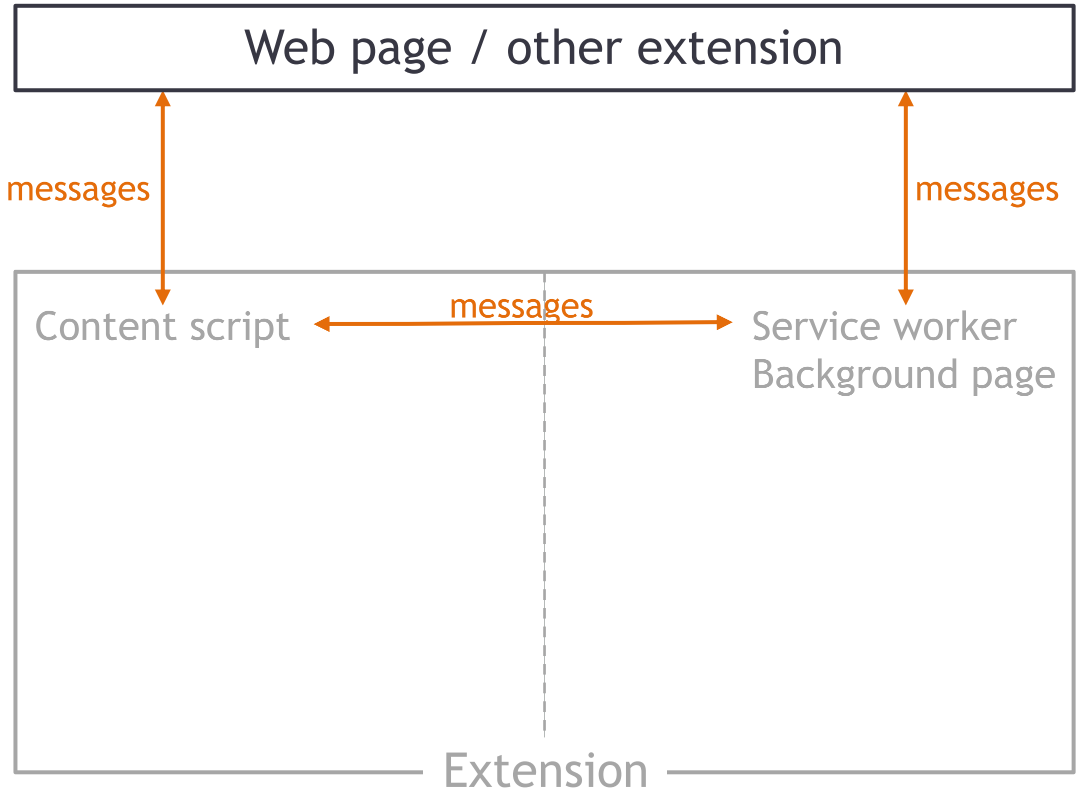
- https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/manifest.json/permissions

Background – 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 – `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`, …)
  - …

https://developer.chrome.com/docs/extensions/reference/manifest

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)
  - SNE definition
  - SNE (comparative) analysis

- Detecting Vulnerable Extensions
  - Threat model & example
  - Case studies, results, and potential defense strategies

- Detecting Fingerprintable Extensions
  - Presentation of 3 fingerprinting vectors, results, and potential mitigations

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

How Safe are Browser Extensions?

- Browser extensions provide additional functionality…

- … so browser extensions need additional & elevated privileges compared to web pages

➢ Browser extensions are an attractive target for attackers 😈

Security-Noteworthy Extensions (SNE)

→ 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
- **Violate the Chrome Web Store policies**
  - E.g., deceive users, promote unlawful activities, lack a privacy policy
- **Contain vulnerabilities**
  - Designed by well-intentioned developers… but contain some vulnerabilities
  - E.g., can lead to user-sensitive data exfiltration

Did you know that…

- **350M users** installed **Security-Noteworthy Extensions** in the last 3 years?
- 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?

Sheryl Hsu
Stanford University
sherylh@stanford.edu

Manda Tran
Stanford University

Aurore Fass
Stanford University, CISPA Helmholtz Center for Information Security
fass@cispa.de

ABSTRACT

This paper is the first attempt at providing a holistic view of the Chrome Web Store (CWS). We leverage historical data provided by ChromeStats to study global trends in the CWS and security implications. We first highlight the extremely *short life cycles of extensions*: roughly 60% of extensions stay in the CWS for one year. Second, we define and show that *Security-Noteworthy Extensions (SNE)* are a significant issue: they pervade the CWS for years and affect almost 350 million users. Third, we identify *clusters of extensions with a similar code base*. We discuss how code similarity techniques could be used to flag suspicious extensions. By developing an approach to extract URLs from extensions' comments, we show that extensions reuse code snippets from public repositories or forums, leading to the propagation of dated code and vulnerabilities. Finally, we underline a critical *lack of maintenance in the CWS*: 60% of the extensions in the CWS have never been updated; half of the extensions known to be vulnerable are still in the CWS and still vulnerable 2 years after disclosure; a third of extensions use vulnerable library versions. We believe that these issues should be widely known in order to pave the way for a more secure CWS.

CCS CONCEPTS

• **Security and privacy → Web application security**; **Browser security**.

KEYWORDS

Browser Extension, Chrome Web Store, ChromeStats, Security-Noteworthy Extension, SNE, Malware, Policy Violation, Vulnerability, Life Cycle, Maintenance, Vulnerable Library, Code Similarity

ACM Reference Format:
Sheryl Hsu, Manda Tran, and Aurore Fass. 2024. What is in the Chrome Web Store?. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3634737.3637636

1 INTRODUCTION

Browser extensions provide additional functionality and customization for browsers. The most popular desktop browser Chrome (with a market share of 66% [70]) has almost 125k extensions, totaling over 1.6 billion active users [6]. Unfortunately, due to their often specialized or privileged capabilities, browser extensions can either be a tool or a target for attackers. Specifically, attackers are developing *malicious extensions* to, e.g., spread malware via malvertising [1], track users [72], spy on users [14], or steal credentials and other sensitive information [22]. At the same time, other extensions have inherently benign functionalities but *contain vulnerabilities* which, if exploited, lead to, e.g., universal cross-site scripting or leaking of sensitive user data [41, 64, 74]. Finally, merely using extensions (independent of whether they have any known flaws) represents a *privacy risk* for Web users. For example, it is possible to infer the set of extensions a user has installed, by observing side effects some extensions induce (browser extension fingerprinting) [50, 59, 60, 62, 63, 67, 69]. This enables an attacker to track users across websites or infer sensitive information about them [46].

To mitigate these issues, browser vendors review extensions prior to publication, e.g., Google engineers vet extensions before their deployment in the Chrome Web Store (CWS) [32]. Despite this vetting process and a decade of work on securing extensions, malicious, vulnerable, and fingerprintable extensions are still found in the CWS [41, 54]. We argue that a potential solution to this issue requires a comprehensive understanding of the underlying ecosystem of browser extensions. Indeed, and perhaps surprisingly, very little is currently known about what lies in browser extension galleries.

This paper provides a holistic view of the browser extension landscape within the CWS. We focus on the CWS because Chrome is the most popular browser, and Chrome extensions are built using the WebExtensions API–a cross-browser technology compatible with Firefox, Opera, Microsoft Edge, etc [56].

We begin our analyses by investigating overall trends in the CWS, along with potential security problems that could arise from having such a big and diverse code base. We first highlight unexpected volatility in terms of extensions in the CWS, e.g., only 60% of extensions are available for one year. Second, we define and investigate "Security-Noteworthy Extensions" (SNE): we analyze malware-containing, policy-violating, and vulnerable extensions. We find that these SNE are a significant problem: over 346 million users installed a SNE in the last 3 years (280M malware, 63M policy violation, and 3M vulnerable). In addition, these extensions are staying in the CWS *for years*, making thorough vetting of extensions and notification of impacted users all the more critical. Third, we uncover thousands of clusters of similar extensions, and we show that investigating extension codes for similarities may enable us to identify unknown SNE. Subsequently, we develop an approach to extract URLs from extensions' comments and attribute instances of code reuse to these cited URLs. We show concrete evidence that extensions reuse code from public repositories or forums directly, and we exemplify how code reuse leads to the propagation of vulnerabilities. Fourth, we show that extensions are globally not maintained: 60% of the extensions in the CWS have never been updated. This has

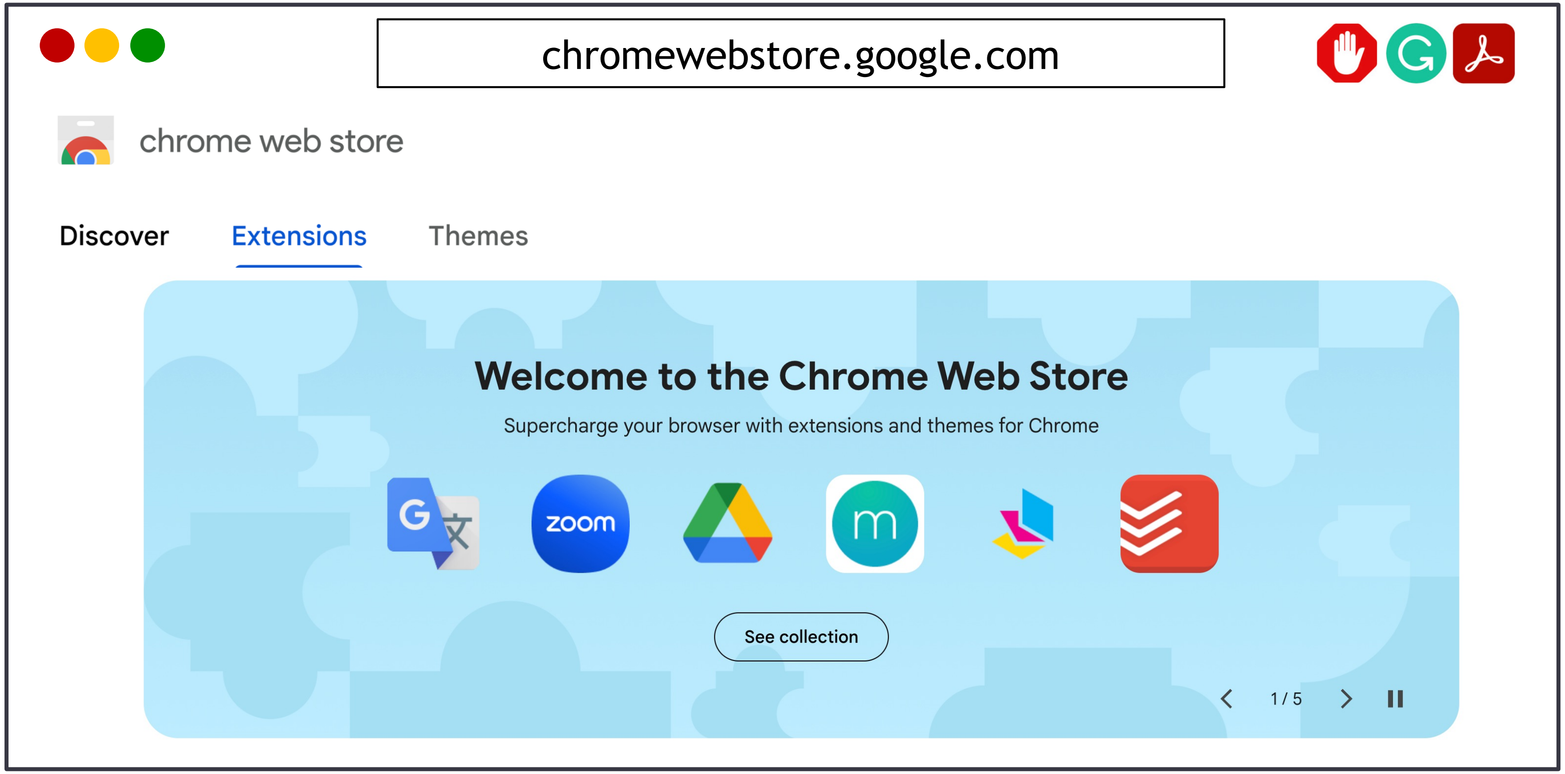
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ASIA CCS '24, July 1–5, 2024, Singapore, Singapore
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0482-6/24/07
https://doi.org/10.1145/3634737.3637636

> What is in the Chrome Web Store?

Hsu et al. AsiaCCS 2024

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

How to Install Extensions or SNE?


chromewebstore.google.com

chrome web store

Discover Extensions Themes

Welcome to the Chrome Web Store

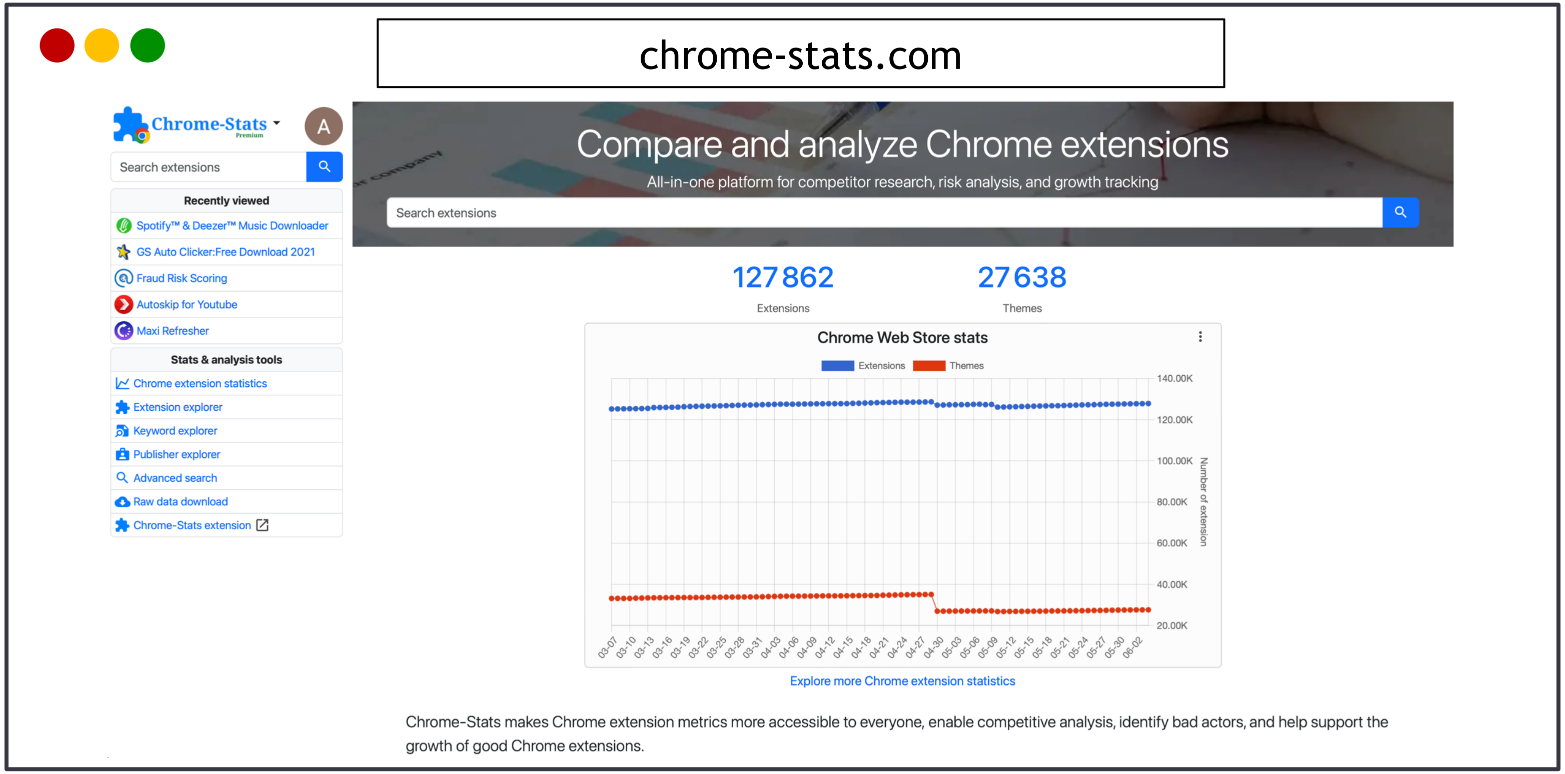
Supercharge your browser with extensions and themes for Chrome

How to Install Extensions or SNE?

>26k SNE

(in the last 3 years)

Browser Extension Collection: Chrome-Stats



chrome-stats.com

Chrome-Stats Premium

Search extensions

Recently viewed

- Spotify™ & Deezer™ Music Downloader
- GS Auto Clicker:Free Download 2021
- Fraud Risk Scoring
- Autoskip for Youtube
- Maxi Refresher

Stats & analysis tools

- Chrome extension statistics
- Extension explorer
- Keyword explorer
- Publisher explorer
- Advanced search
- Raw data download
- Chrome-Stats extension

Compare and analyze Chrome extensions

All-in-one platform for competitor research, risk analysis, and growth tracking

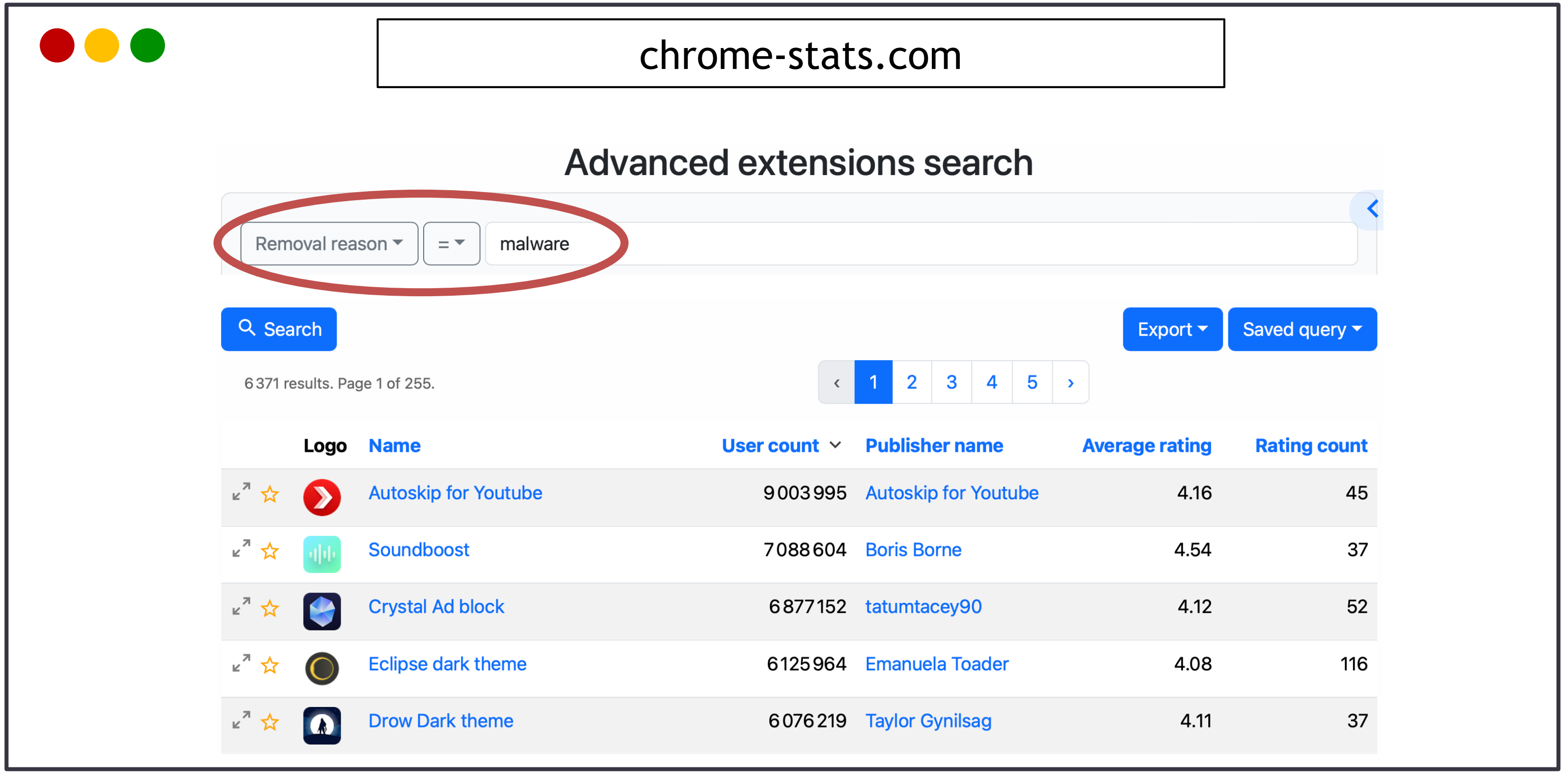
Search extensions

127 862	27 638
Extensions	Themes

Explore more Chrome extension statistics

Chrome-Stats makes Chrome extension metrics more accessible to everyone, enable competitive analysis, identify bad actors, and help support the growth of good Chrome extensions.

Malicious Extension Collection: Chrome-Stats



chrome-stats.com

Advanced extensions search

Removal reason = malware

Search

Export Saved query

6 371 results. Page 1 of 255.

‹ 1 2 3 4 5 ›

Logo	Name	User count	Publisher name	Average rating	Rating count
	Autoskip for Youtube	9 003 995	Autoskip for Youtube	4.16	45
	Soundboost	7 088 604	Boris Borne	4.54	37
	Crystal Ad block	6 877 152	tatumtacey90	4.12	52
	Eclipse dark theme	6 125 964	Emanuela Toader	4.08	116
	Drow Dark theme	6 076 219	Taylor Gynilsag	4.11	37

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 [1]	184	152	March 16, 2021
Benign extensions	226,762	92,482	Before May 1, 2023

[1] Fass et al., CCS 2021

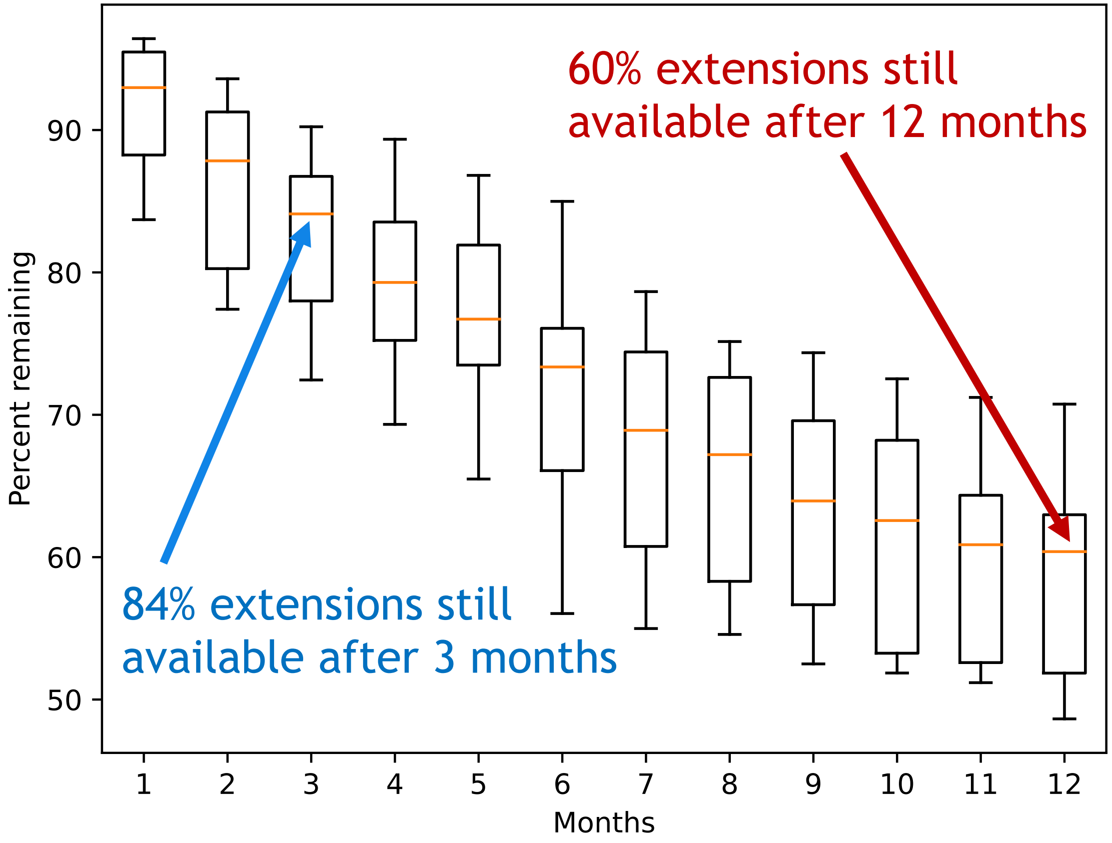
Life Cycle of Extensions

Methodology:

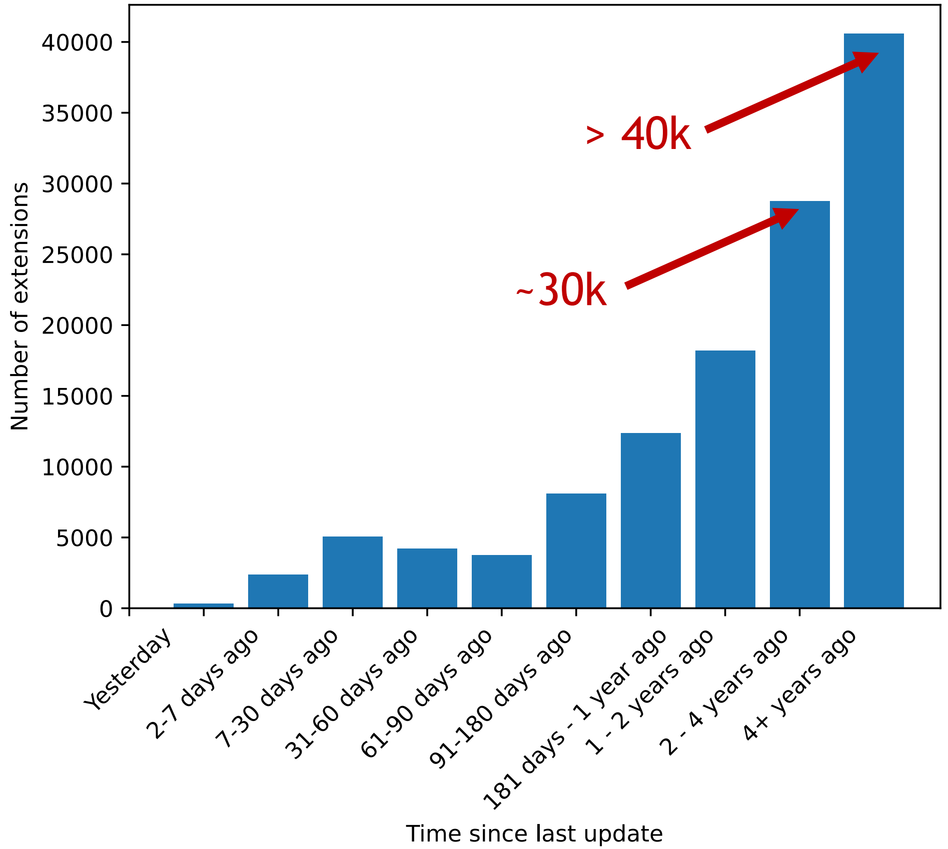
- Collected extensions added to the CWS in Jan—Dec 2021
- Computed the percentage of those extensions still in the CWS 1, 2, ..., 12 months later

➢ **Extensions have a very short life cycle**

➢ Analyses on the CWS should be **run regularly**

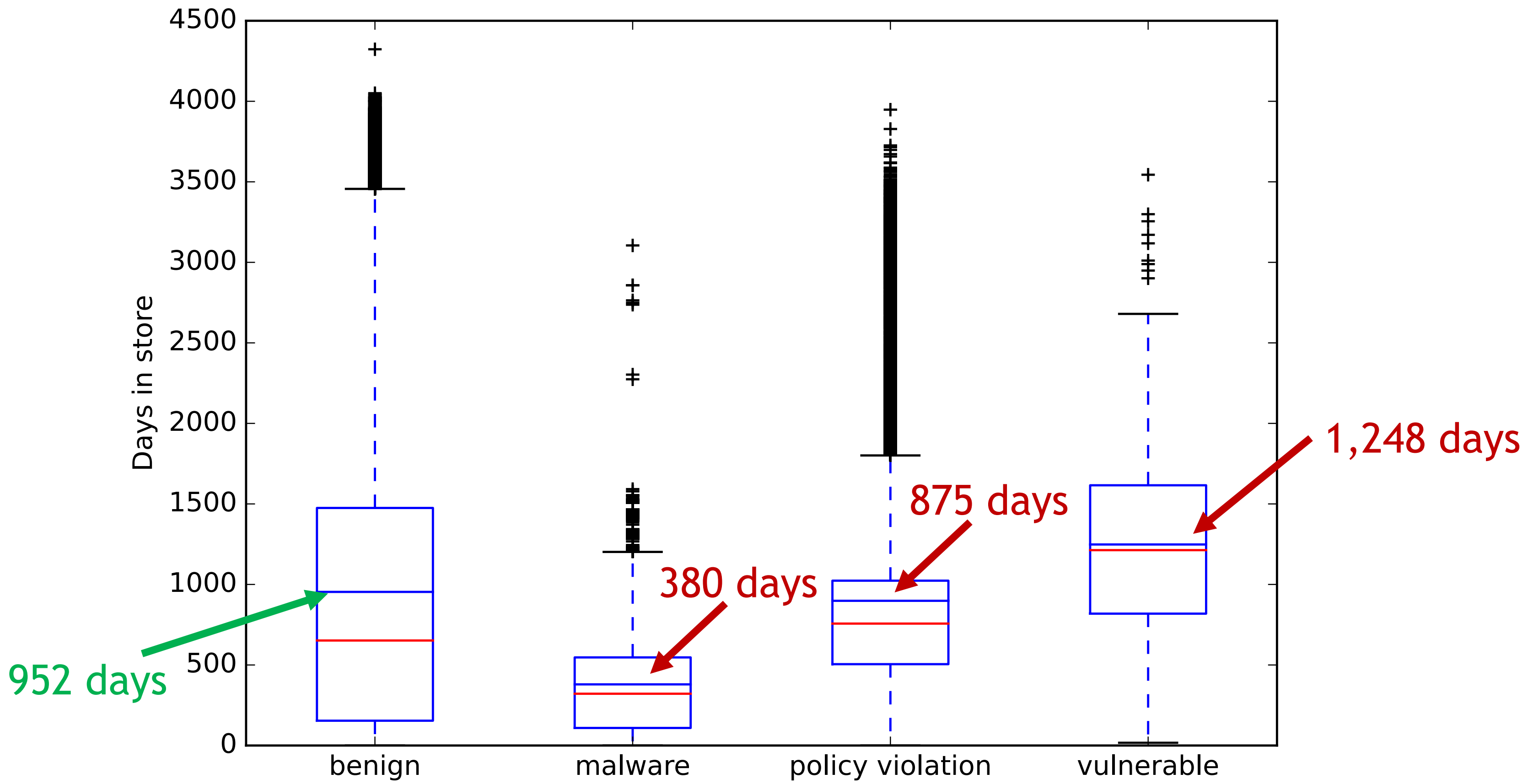


Extension Maintenance and Security

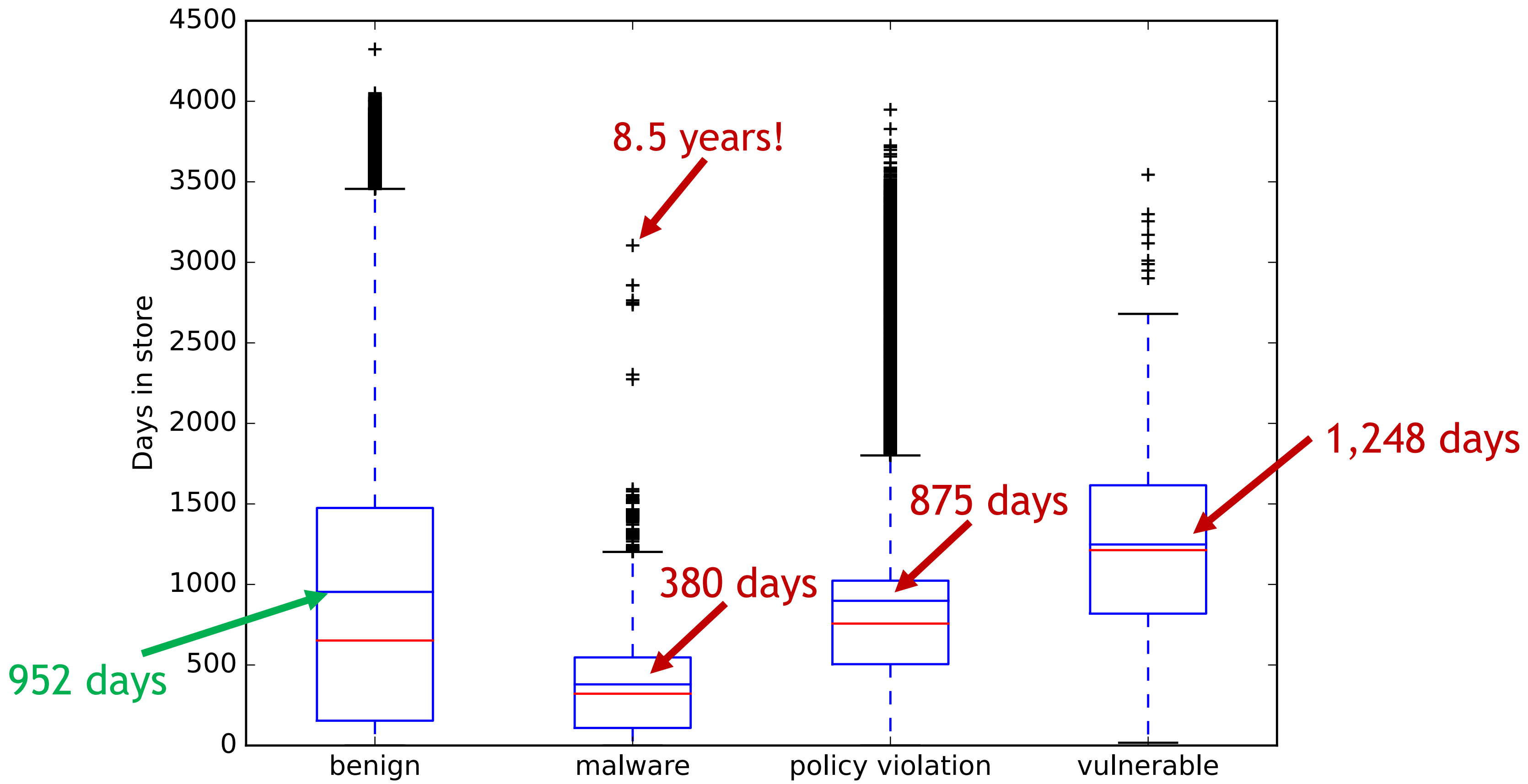


- Critical **lack of maintenance** in the CWS
- **60%** of the extensions have **never been updated**
- Security & privacy implications

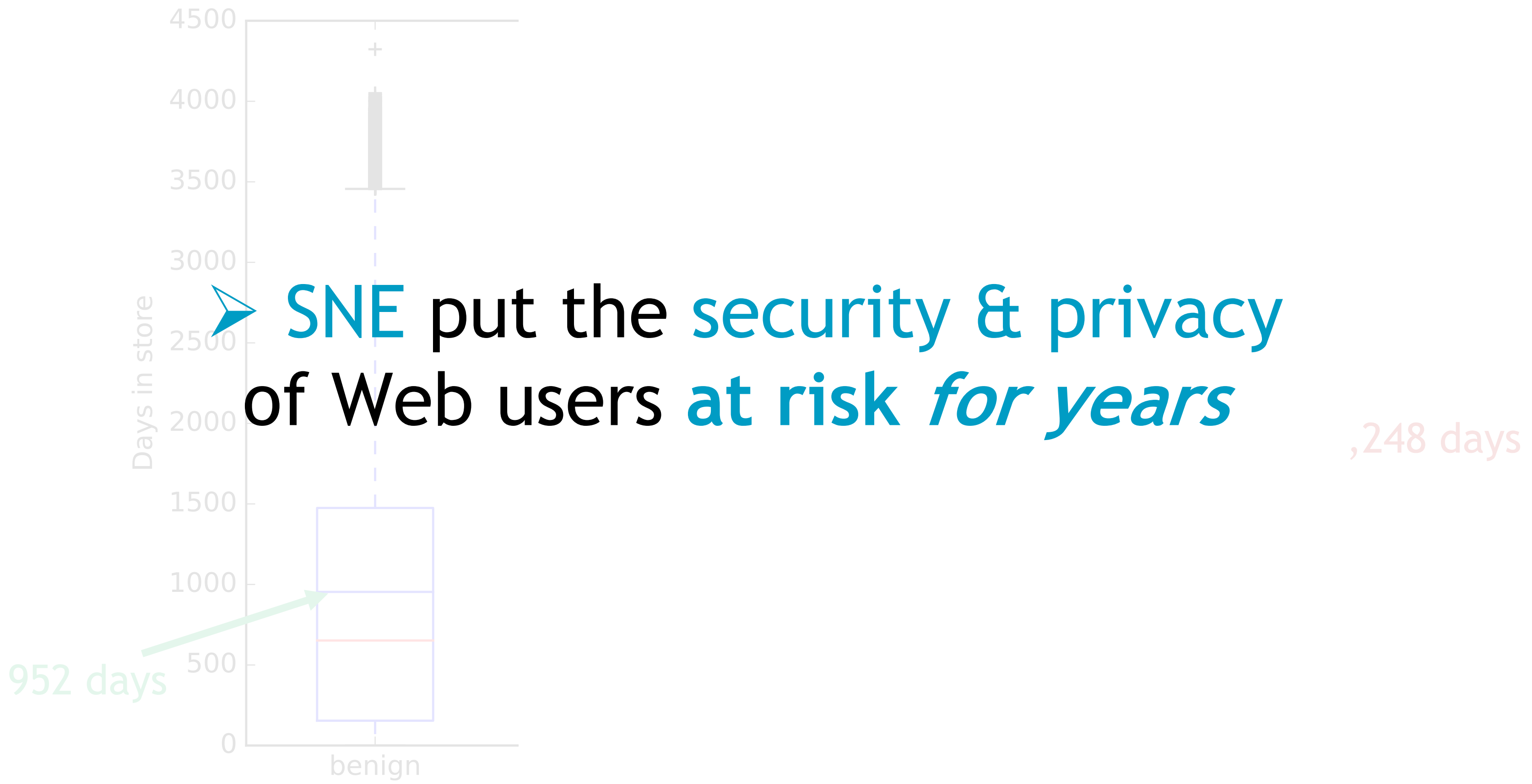
Number of Days in the CWS



Number of Days in the CWS

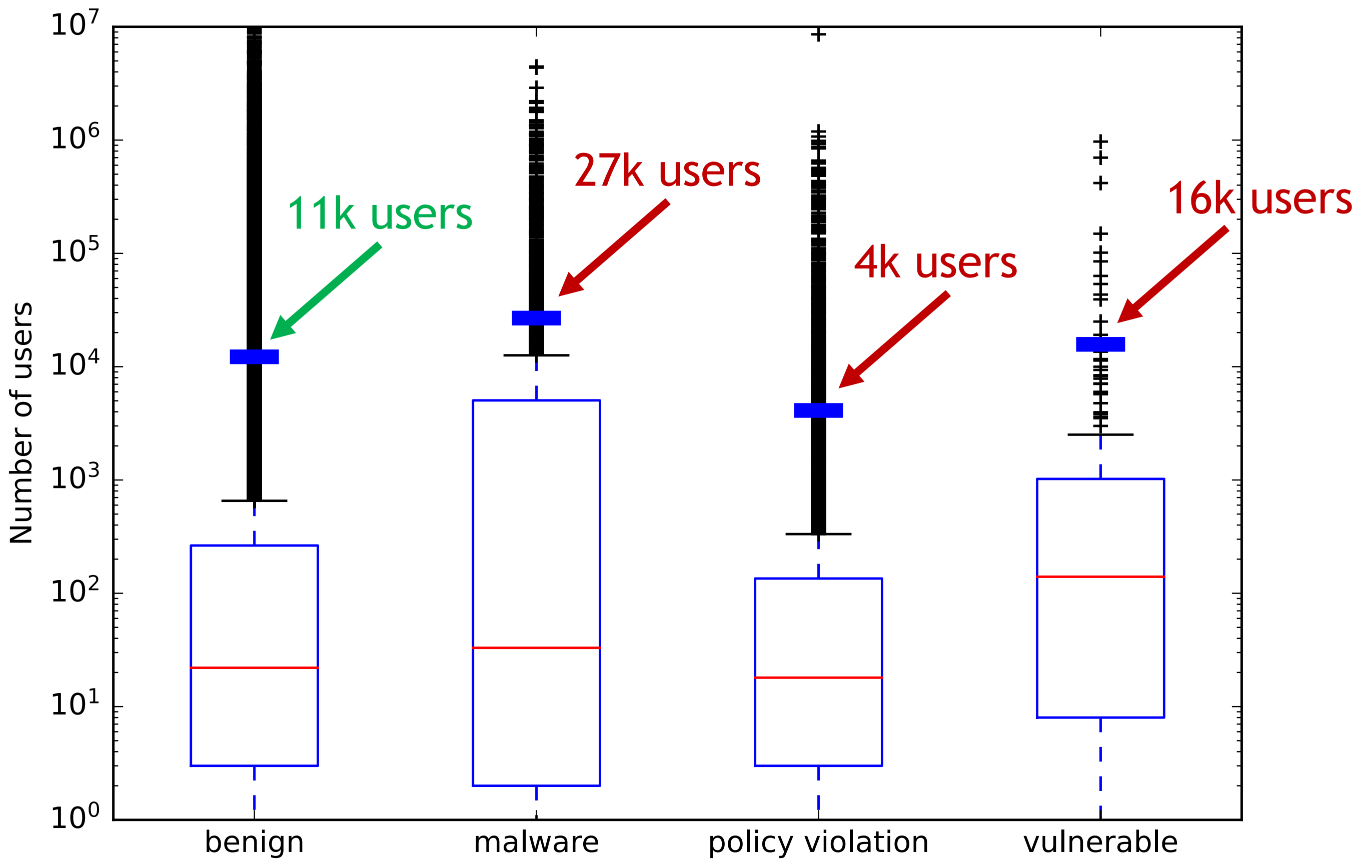


Number of Days in the CWS

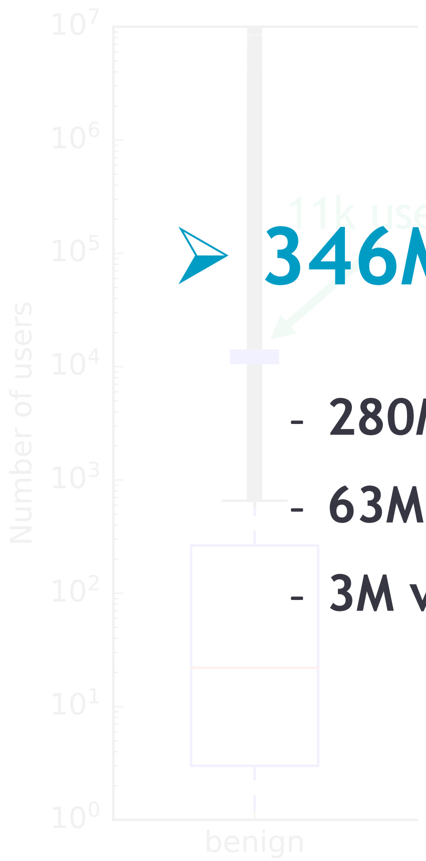


➢ SNE put the security & privacy of Web users **at risk *for years***

Number of Users



Number of Users

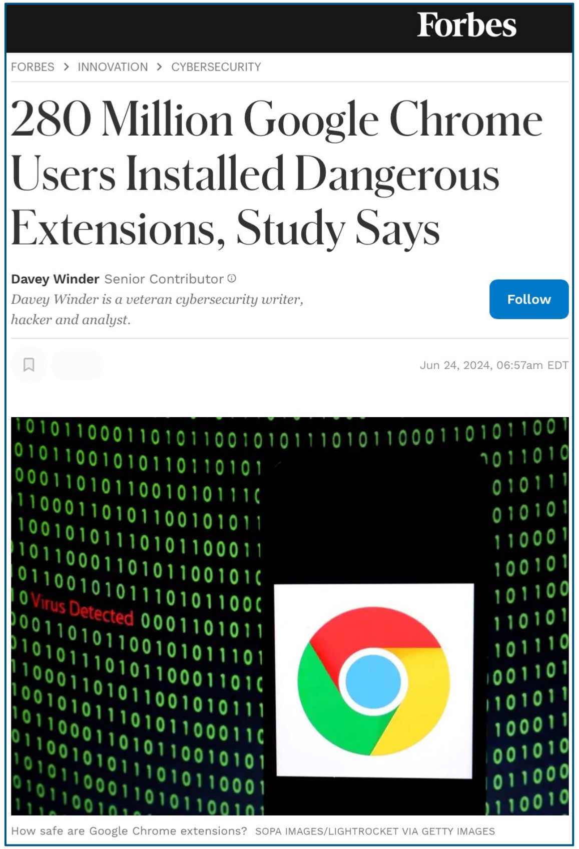


➤ **346M users installed SNE**

(in the last 3 years):

- **280M malware-containing**
- **63M policy-violating**
- **3M vulnerable extensions**

Media Coverage


Forbes

FORBES > INNOVATION > CYBERSECURITY

280 Million Google Chrome Users Installed Dangerous Extensions, Study Says

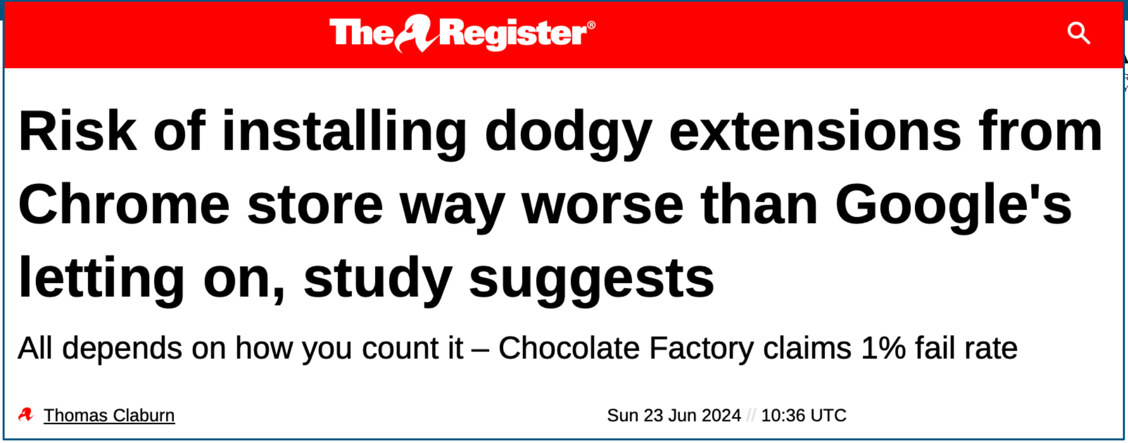
Davey Winder Senior Contributor

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

Follow

Jun 24, 2024, 06:57am EDT

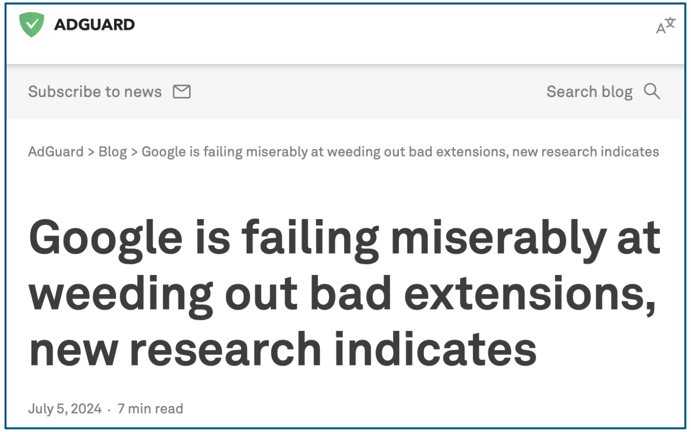
How safe are Google Chrome extensions? SOPA IMAGES/LIGHTROCKET VIA GETTY IMAGES


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

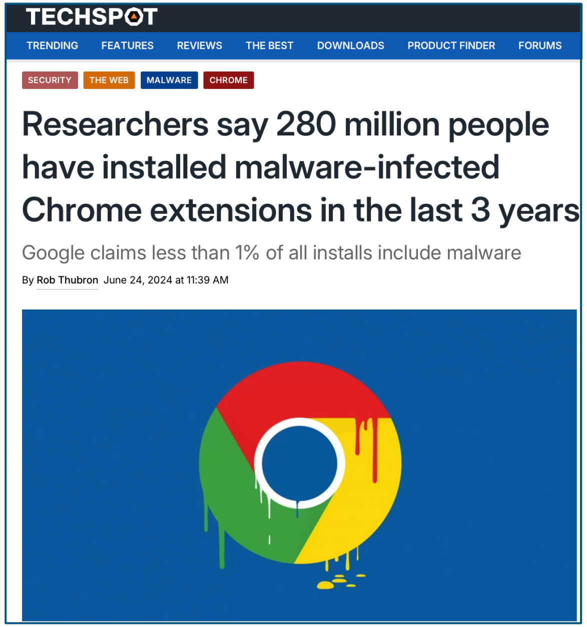

ADGUARD

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

Outline

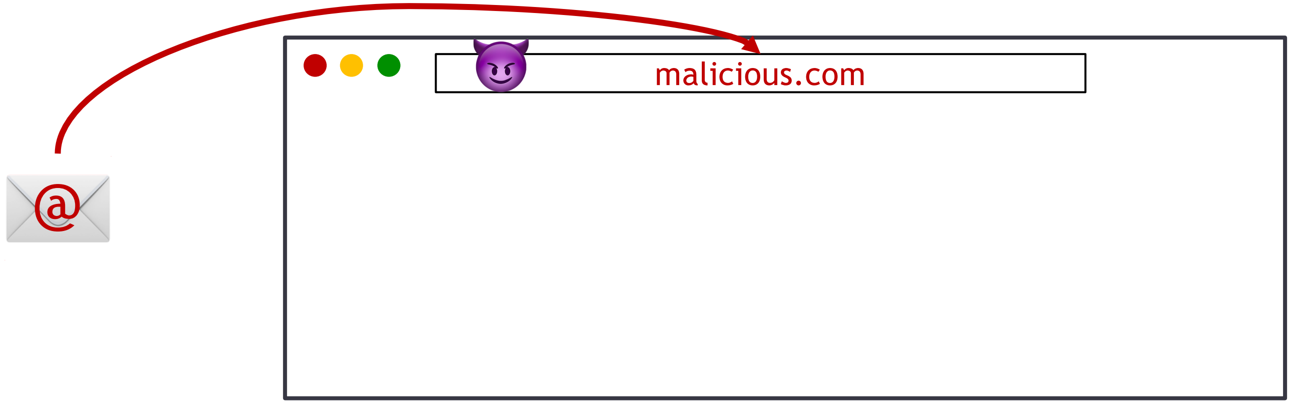
- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - SNE (comparative) analysis
- Detecting Vulnerable Extensions
  - Threat model & example
  - Case studies, results, and potential defense strategies
- Detecting Fingerprintable Extensions
  - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Analysis of Vulnerable Extensions: Web Attacker

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

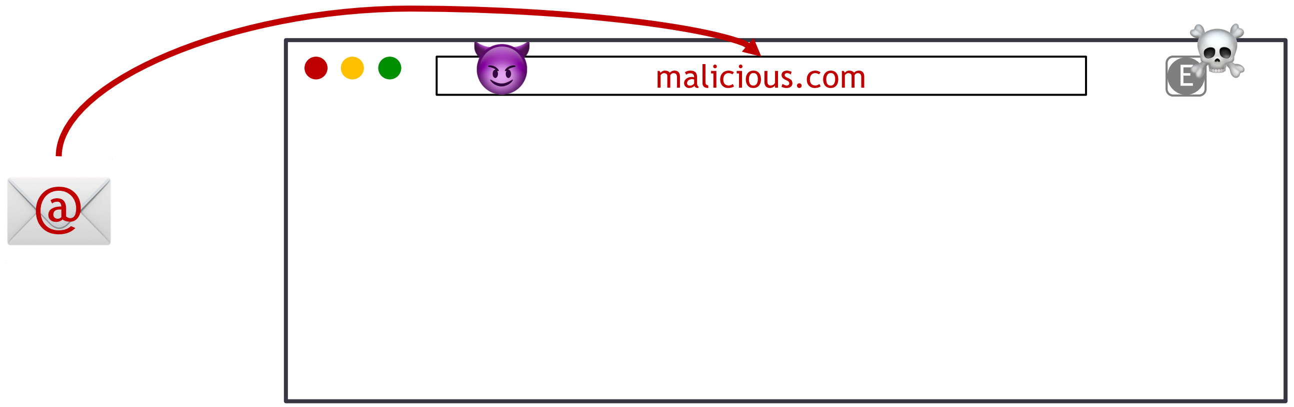
Analysis of Vulnerable Extensions: Web Attacker

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



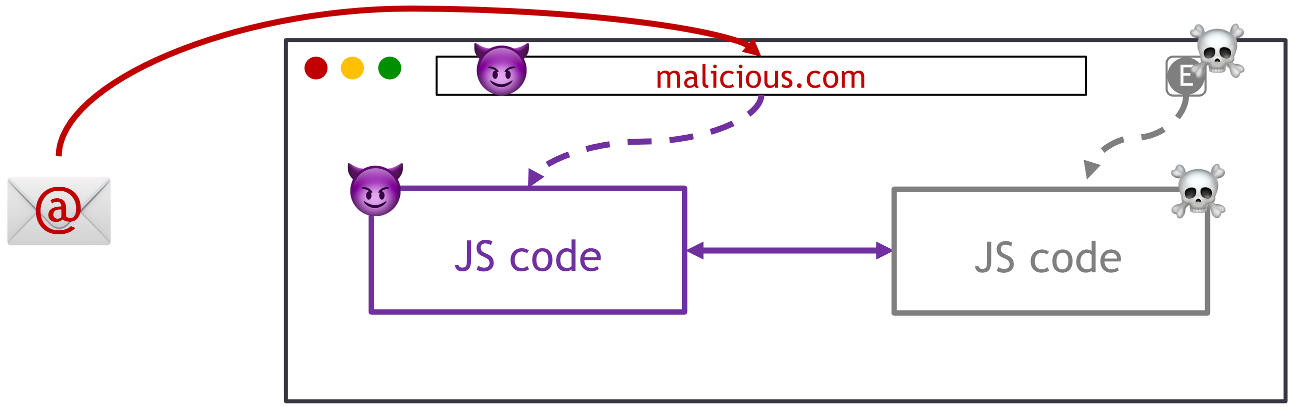
Analysis of Vulnerable Extensions: Web Attacker

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



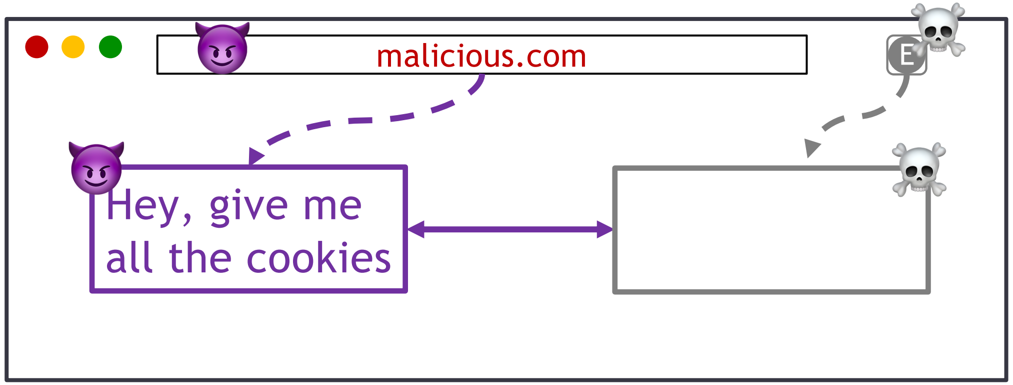
Analysis of Vulnerable Extensions: Web Attacker

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



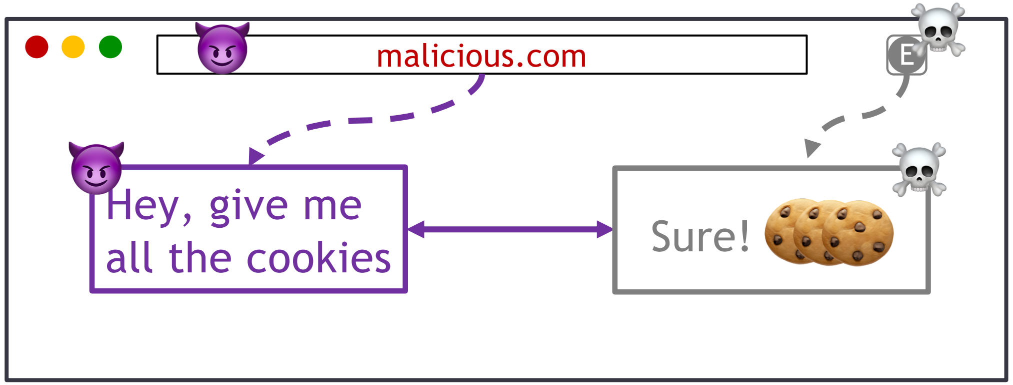
Analysis of Vulnerable Extensions: Web Attacker

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



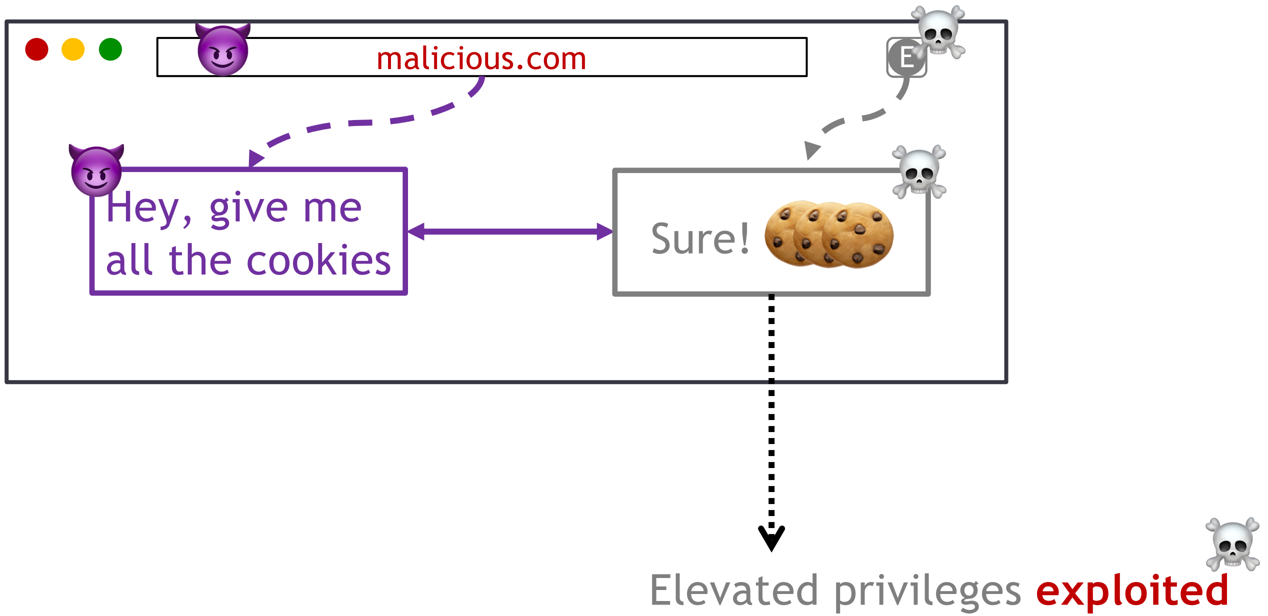
Analysis of Vulnerable Extensions: Web Attacker

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



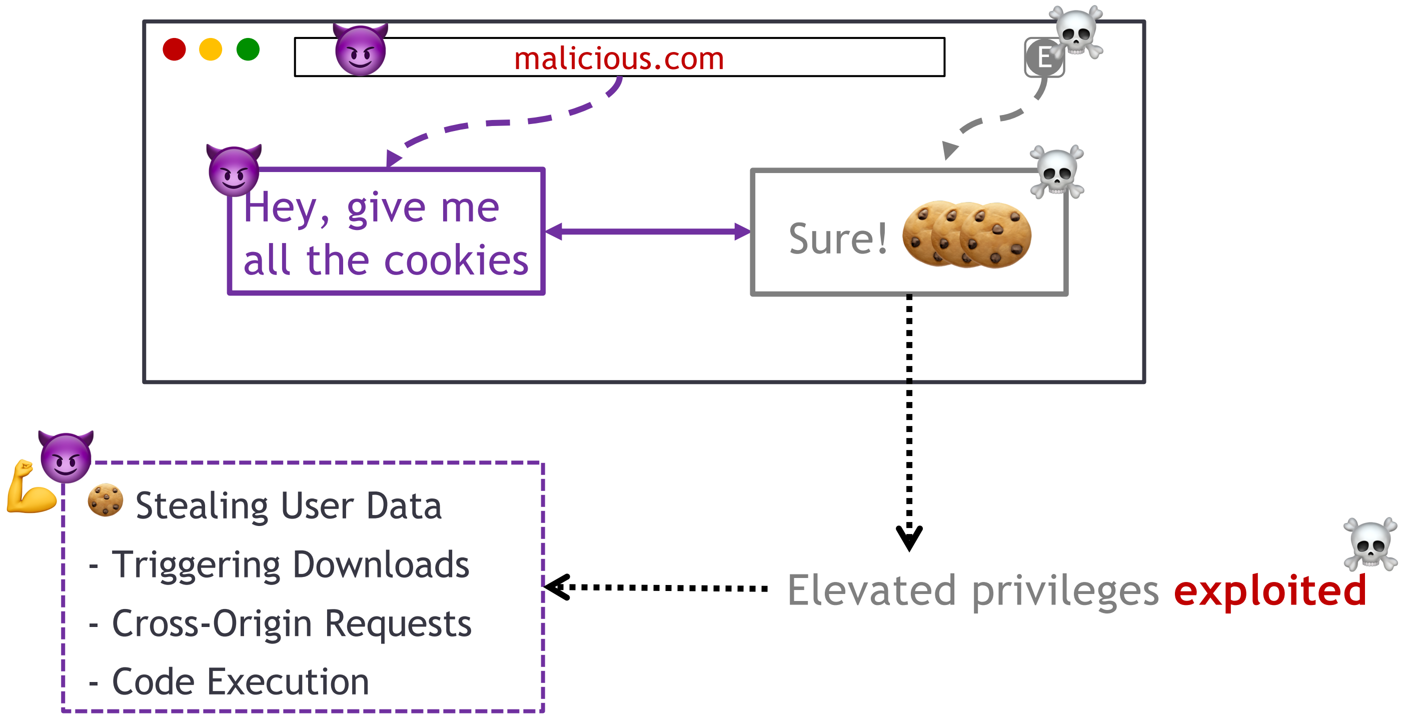
Analysis of Vulnerable Extensions: Web Attacker

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


malicious.com
E
Hey, give me all the cookies
Sure!
Elevated privileges **exploited**

Analysis of Vulnerable Extensions: Web Attacker

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



🍪 Stealing User Data
- Triggering Downloads
- Cross-Origin Requests
- Code Execution

Elevated privileges **exploited**

Background – postMessage

- To send messages:
  - `otherWindow.postMessage(message, targetOrigin)`
- To receive messages:
  - With an *event handler* (`addEventListener` or `onmessage`)
- /!\ The 2 origins must trust each other → verify the origin before processing a message

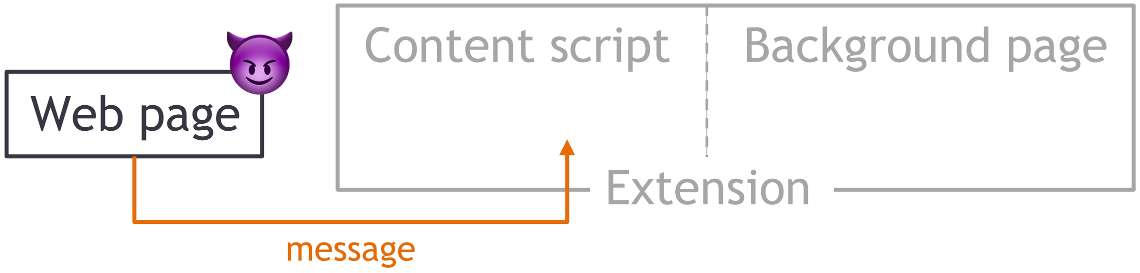
https://developer.mozilla.org/en-US/docs/Web/API/Window/postMessage

Simplified Example of a Vulnerability

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



})
```

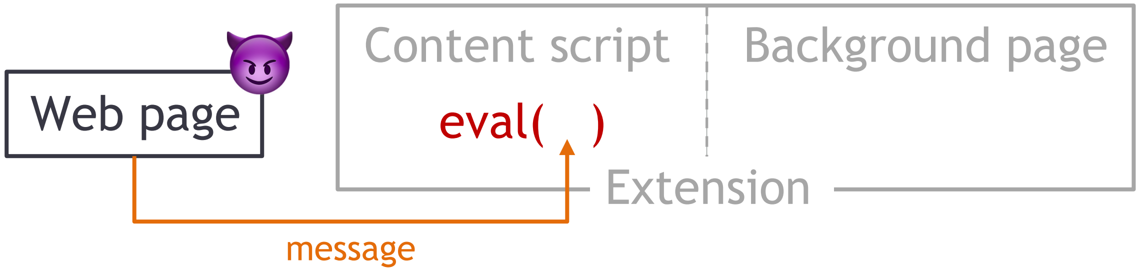


Simplified Example of a Vulnerability

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


    eval(event.data);


})
```

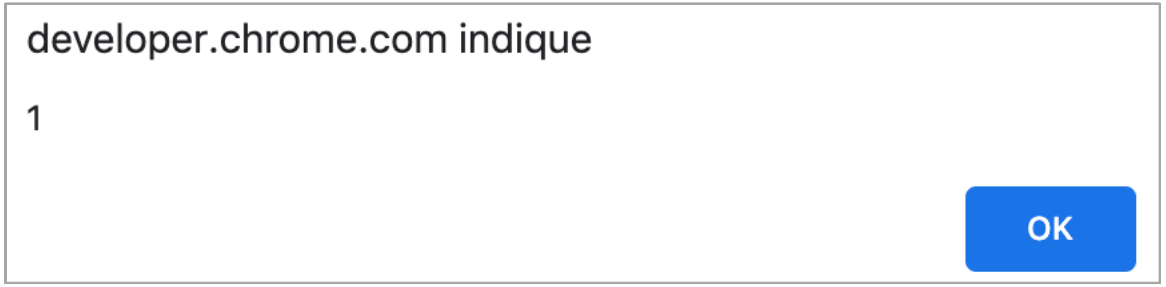


Simplified Example of a Vulnerability

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

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

malicious payload


developer.chrome.com indique

1

Detecting Vulnerable Extensions

Fass et al.
CCS 2021

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DoubleX. DoubleX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DoubleX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DoubleX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy → Web application security; Browser security.**

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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

Detecting Vulnerable Extensions

Fass et al.
CCS 2021

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DoubleX. DoubleX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DoubleX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DoubleX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy → Web application security; Browser security.**

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

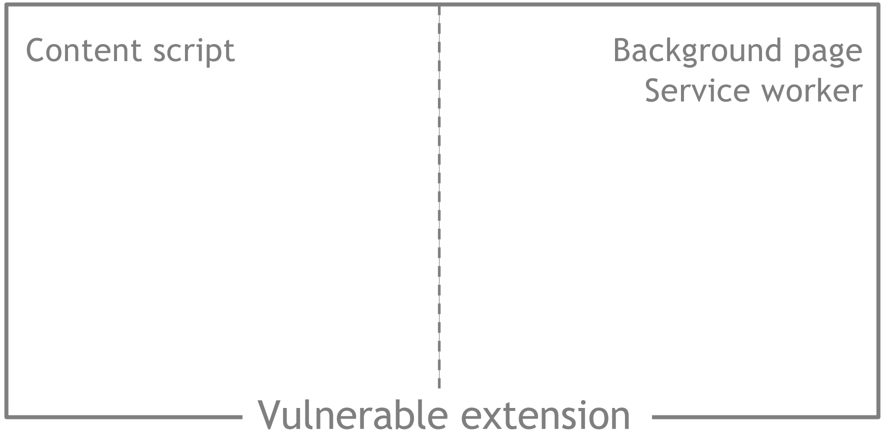
The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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



Detecting Vulnerable Extensions

Fass et al. CCS 2021

DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DOUBLEX. DOUBLEX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DOUBLEX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DOUBLEX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy → Web application security; Browser security.**

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

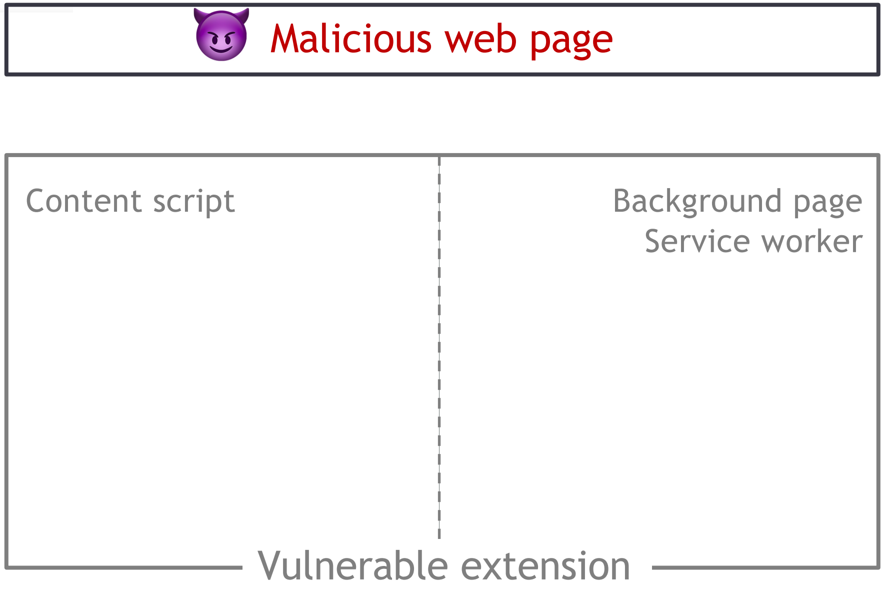
The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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



Detecting Vulnerable Extensions

Fass et al.
CCS 2021

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DoubleX. DoubleX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DoubleX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DoubleX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy → Web application security**; **Browser security**.

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

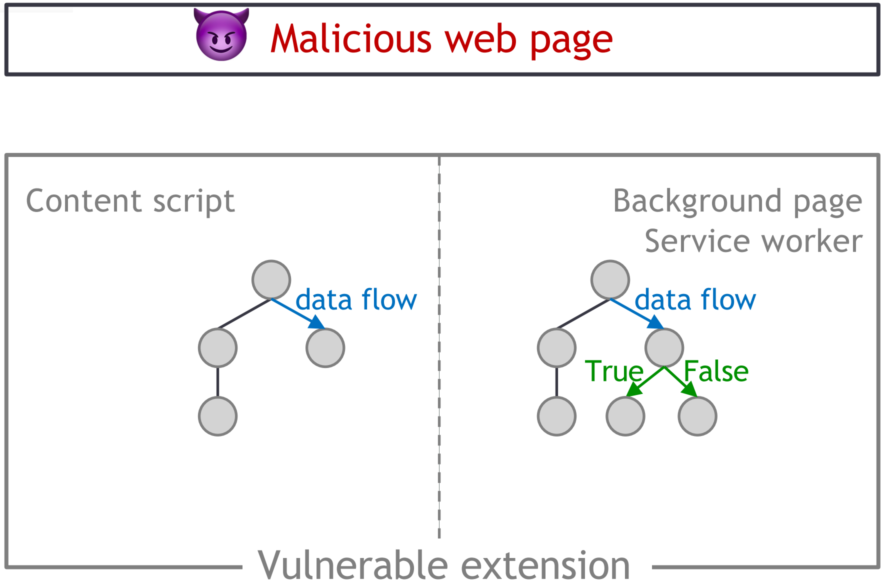
The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> 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 JS code abstraction

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

Detecting Vulnerable Extensions

Fass et al. CCS 2021

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DoubleX. DoubleX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DoubleX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DoubleX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy** → **Web application security**; **Browser security**.

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

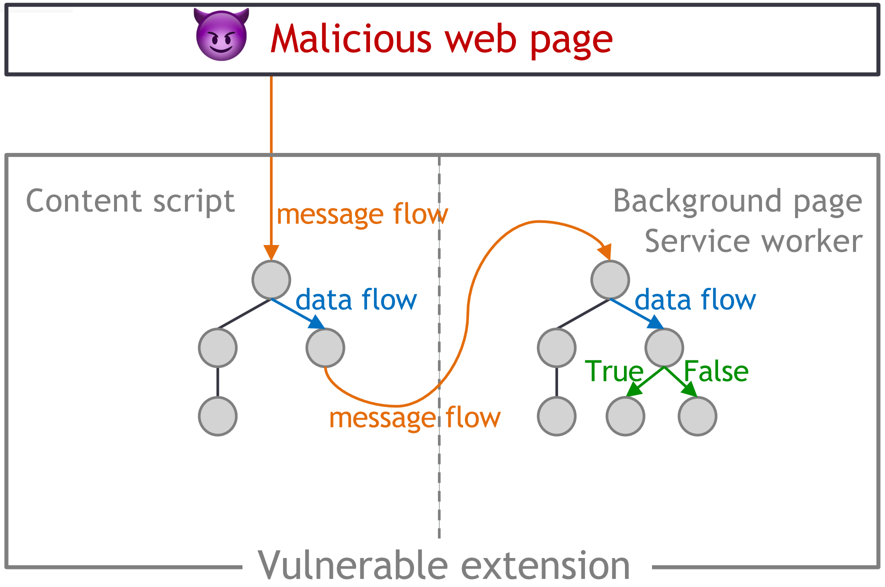
The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> 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 JS code abstraction

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

Extension Dependence Graph (EDG)

➢ Message interactions

Detecting Vulnerable Extensions

Fass et al. CCS 2021

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock
CISPA Helmholtz Center for Information Security
{aurore.fass,doliere.some,backes,stock}@cispa.de

Abstract

Browser extensions are popular to enhance users' browsing experience. By design, they have access to security- and privacy-critical APIs to perform tasks that web applications cannot traditionally do. Even though web pages and extensions are isolated, they can communicate through messages. Specifically, a vulnerable extension can receive messages from another extension or web page, under the control of an attacker. Thus, these communication channels are a way for a malicious actor to elevate their privileges to the capabilities of an extension, which can lead to, e.g., universal cross-site scripting or sensitive user data exfiltration. To automatically detect such security and privacy threats in benign-but-buggy extensions, we propose our static analyzer DoubleX. DoubleX defines an Extension Dependence Graph (EDG), which abstracts extension code with control and data flows, pointer analysis, and models the message interactions within and outside of an extension. This way, we can leverage this graph to track and detect suspicious data flows between external actors and sensitive APIs in browser extensions.

We evaluated DoubleX on 154,484 Chrome extensions, where it flags 278 extensions as having a suspicious data flow. Overall, we could verify that 89% of these flows can be influenced by external actors (i.e., an attacker). Based on our threat model, we subsequently demonstrate exploitability for 184 extensions. Finally, we evaluated DoubleX on a labeled vulnerable extension set, where it accurately detects almost 93% of known flaws.

CCS Concepts

• **Security and privacy → Web application security; Browser security.**

Keywords

Web Security, Browser Extension, JavaScript, Vulnerability Detection, Static Analysis, Data Flow Analysis

ACM Reference Format:
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock. 2021. DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS '21), November 15–19, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3460120.3484745

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CCS '21, November 15–19, 2021, Virtual Event, Republic of Korea
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8454-4/21/11...$15.00
https://doi.org/10.1145/3460120.3484745

1 Introduction

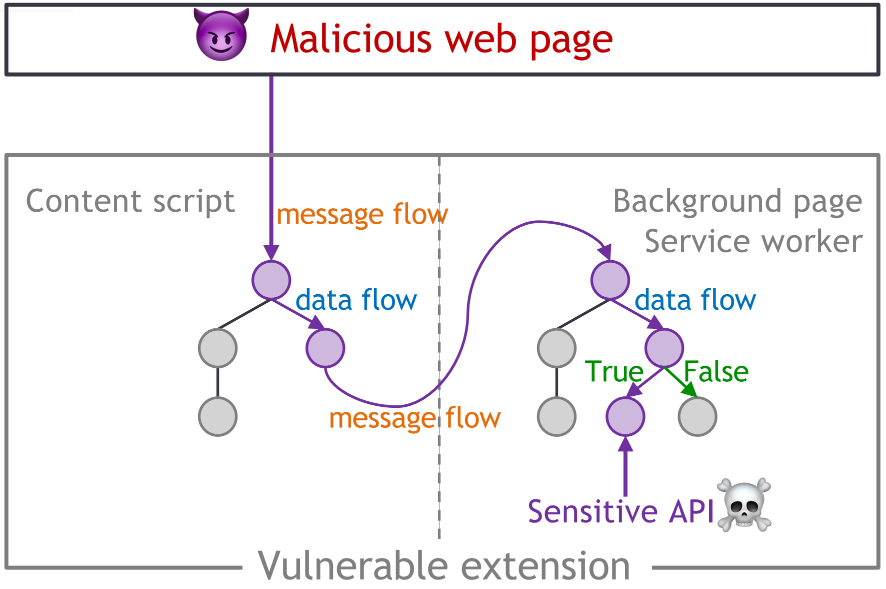
The Web has become a popular ecosystem used by billions of people every day. To extend their browser functionality, users install browser extensions. For the most popular desktop browser Chrome (with a market share of 67% [78]), there are almost 200,000 extensions, totaling over 1.2 billion installs [26]. While some extensions merely customize user browser interface, others serve more security- and privacy-critical tasks, e.g., to be effective, an ad-blocker needs to modify web page content or intercept network requests. To achieve this, and contrary to regular JavaScript in web pages, extensions have privileged capabilities, such as downloading arbitrary files or accessing arbitrary cross-domain data.

Given their elevated privileges, extensions attract the interest of attackers [1, 35, 39, 81, 82]. Still, Google engineers are actively working on detecting such malicious extensions in their store. In February 2020, they removed 500 extensions that were exfiltrating user data [41]; in April 2020, 49 additional extensions that were hijacking users' cryptocurrency wallets [42]; in June 2020, an extra 70 spying extensions [48]; and, in December 2020, extensions redirecting to phishing websites [43]. In addition, before being published, extensions are reviewed by Chrome's vetting system to flag extensions requiring many or powerful privileges for further analyses [16] and to directly detect the ones that may contain or spread malicious software. This process makes it harder to have malicious extensions in the store today.

Still, malicious extensions represent only a fraction of the extensions that may lead to security or privacy issues. In fact, an attacker can also abuse vulnerable extensions to elevate its privileges through the capabilities of an extension. To this end, an attacker can leverage an extension's communication channels to send payloads to this extension, tailored to exploit its vulnerabilities. Such vulnerabilities can lead to, e.g., universal cross-site scripting (XSS) (i.e., the ability to execute code in every website even without a vulnerability in the site itself) or sensitive user data exfiltration. Due to their inherently benign intent, these vulnerable extensions are more challenging to detect than malicious ones, e.g., as they are not doing anything suspicious. Furthermore, even though they do require powerful privileges, their benign nature allows them to pass the review process. While some previous works have focussed on vulnerable extensions, they were either purely formal [7], specific to the deprecated Firefox XPCOM [52] infrastructure [5, 6], or based on primarily manual analysis [8]. To the best of our knowledge, only EmPoWeb from Somé [72] focuses on analyzing extensions' susceptibility to attacks through external messages at scale. In practice, though, his work is based on a lightweight call graph analysis and yields an extremely high number of reported extensions to manually vet: of the 66k Chrome extensions analyzed, it flagged 3.3k as suspicious, and 95% of them were false alarms. In fact, we

> 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 JS code abstraction

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

Extension Dependence Graph (EDG)

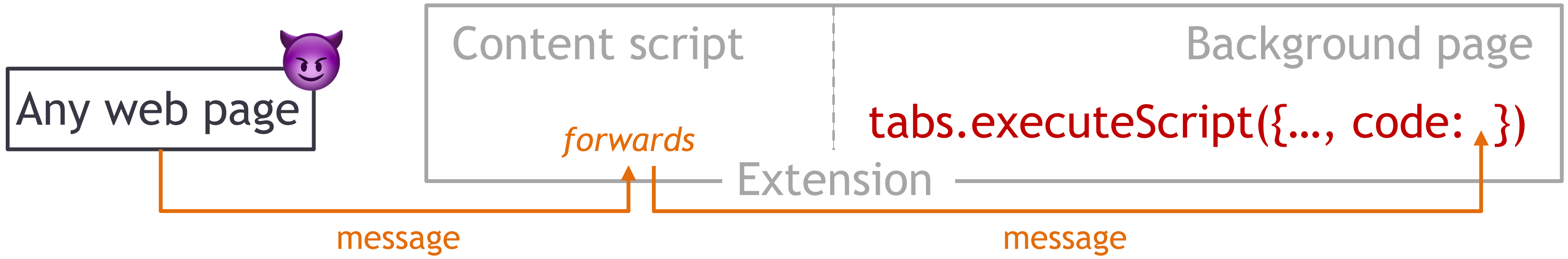
- ➢ Message interactions

Suspicious data flow tracking

- ➢ Detects any path between an attacker & sensitive APIs

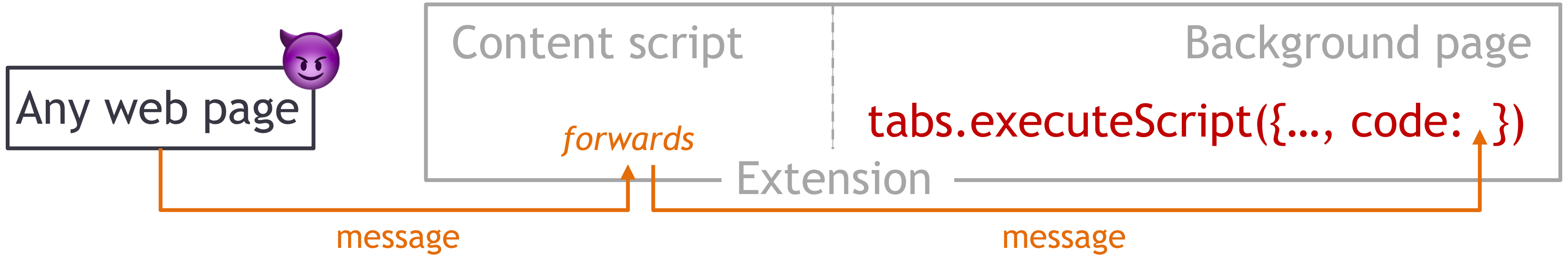
Case Studies of Vulnerable Chrome Extensions

- Arbitrary code execution (*cdi*…, 4k+ users)

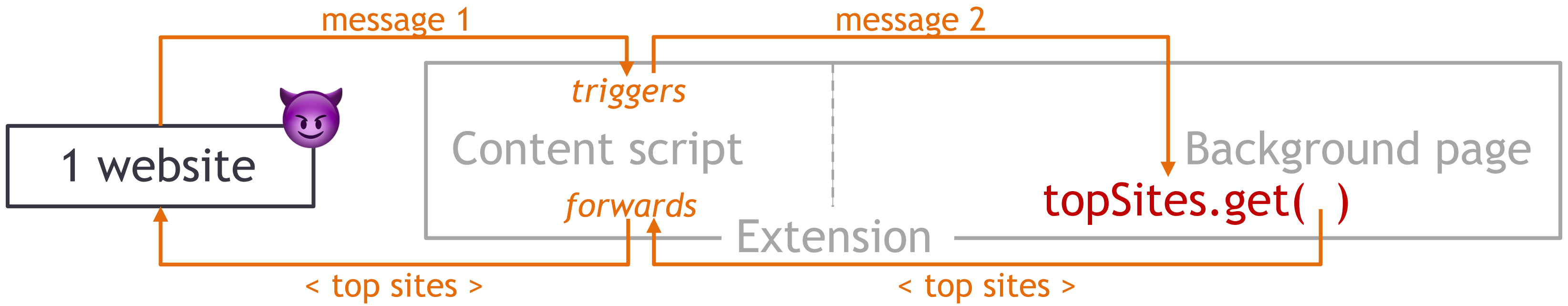


Case Studies of Vulnerable Chrome Extensions

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



- Most visited website exfiltration (*lkl*..., 700k+ users)



Detecting Vulnerable Extensions with DOUBLEX

Analyzed 155k Chrome extensions from 2021 with DOUBLEX

- **184 vulnerable Chrome extensions**
- Impacting **3M users**

- **Precision: 89%** of the flagged extensions are vulnerable
- **Recall: 93%** of known vulnerabilities [2] are detected

- **Integration** in the **vetting process** conducted by Google
- **Available online**, for developers

(even in other fields!)

Aurore54F/DoubleX

[2] Somé, IEEE S&P 2019

Defenses & Perspectives

- Know that communication with external actors may be dangerous
- Only allow communication with specified extensions or web pages
- Limit:
  - code execution by sanitizing messages
  - SOP bypass by preferring CORS for cross-origin requests
- DOUBLEX could provide a feedback channel for developers
- Migrate an extension to Manifest V3

Outline

- Background: Browser Extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - SNE (comparative) analysis
- Detecting Vulnerable Extensions
  - Threat model & example
  - Case studies, results, and potential defense strategies
- Detecting Fingerprintable Extensions
  - Presentation of 3 fingerprinting vectors, results, and potential mitigations

Browser Extension Fingerprinting

Browser extensions can **interact with web pages…**

Browser Extension Fingerprinting

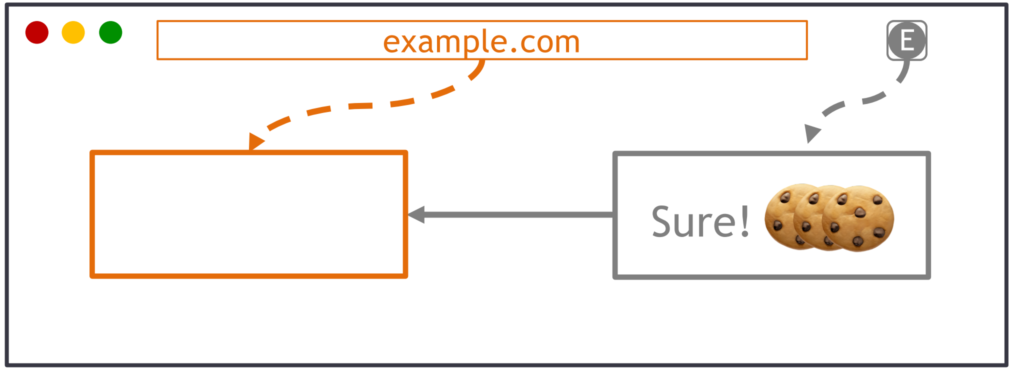
Browser extensions can **interact with web pages…**

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

Browser Extension Fingerprinting

Browser extensions can **interact with web pages…**

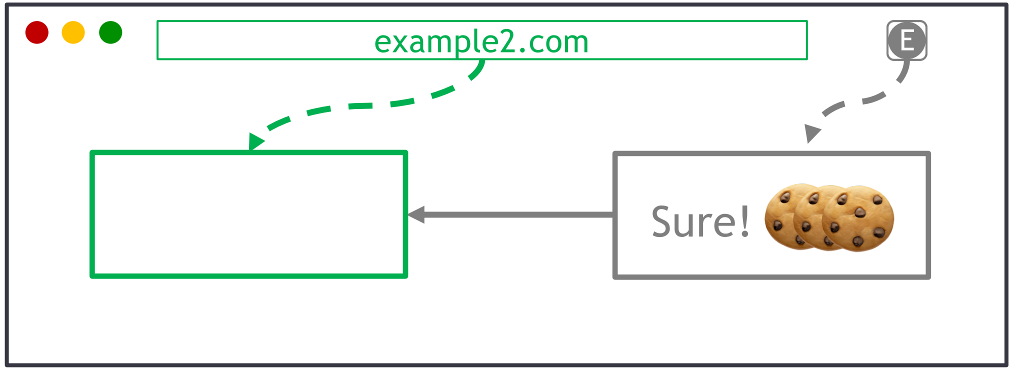
- 1) Extensions send **PostMessage** to web pages



Browser Extension Fingerprinting

Browser extensions can **interact with web pages…**

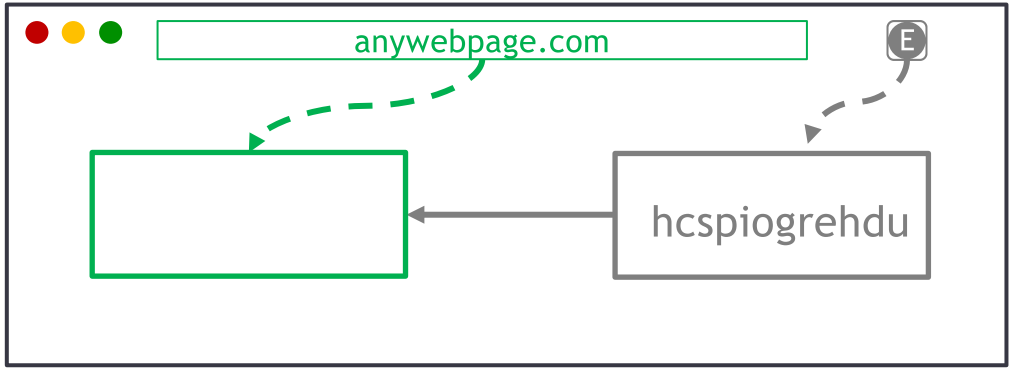
- 1) Extensions send **PostMessage** to web pages



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**

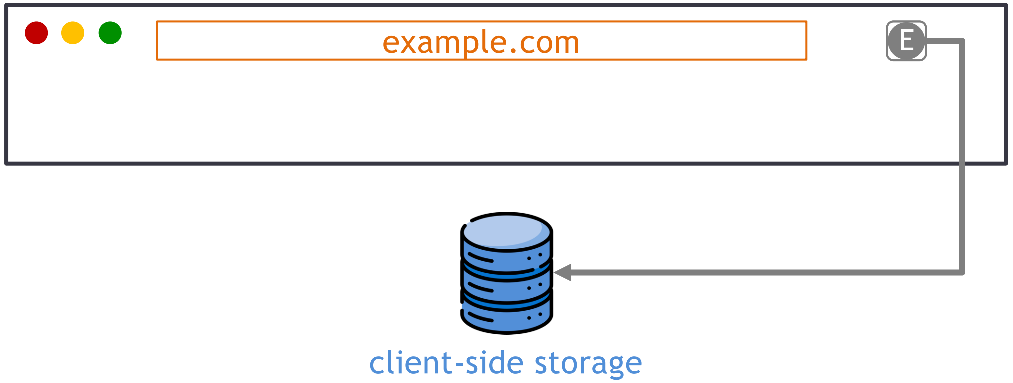
  (e.g., cookies, local/session storage, IndexedDB)

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)

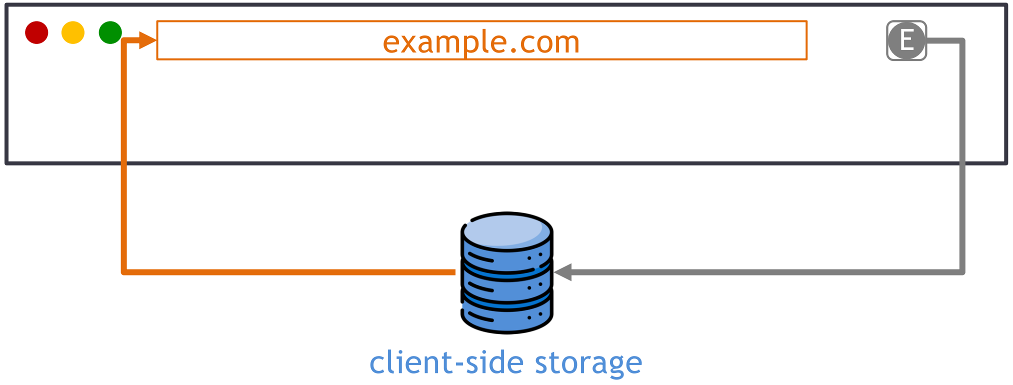


Browser Extension Fingerprinting

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)

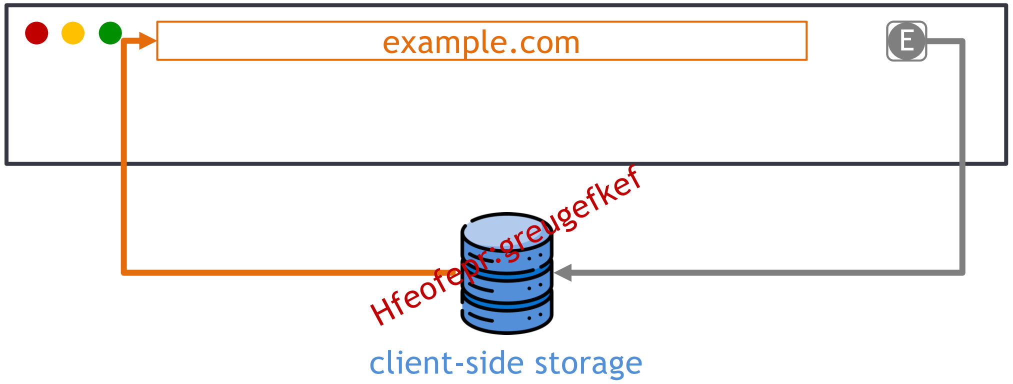


Browser Extension Fingerprinting

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

… which leaves traces

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

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, indexDB)
- 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, indexDB)
- 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

Why is this bad?

- “Malicious” websites can **track users** by fingerprinting their extensions

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, indexDB)
- “Malicious” websites can **track users** by fingerprinting their extensions
- 3) Extensions **inject JavaScript** code directly into web pages
  → registering global variables
  → invocation of global APIs and properties
- Extensions can **reveal personal user information**, e.g., geolocation, ethnicity, social/personal interests, medical issues, religion, etc. [3]

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

[3] Karami, NDSS 2020

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?*

Agarwal et al.
CCS 2024

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

Shubham Agarwal
CISPA Helmholtz Center for Information Security
shubham.agarwal@cispa.de

Aurore Fass
CISPA Helmholtz Center for Information Security
fass@cispa.de

Ben Stock
CISPA Helmholtz Center for Information Security
stock@cispa.de

ABSTRACT

Browser extensions are third-party add-ons that provide myriads of features to their users while browsing on the Web. Extensions often interact with the websites a user visits and perform various operations such as DOM-based manipulation, script injections, and so on. However, this also enables nefarious websites to track their visitors by fingerprinting extensions. Researchers in the past have shown that extensions are susceptible to fingerprinting based on the resources they include, the styles they deploy, or the DOM-based modifications they perform. Fortunately, the current extension ecosystem contains safeguards against many such known issues through appropriate defense mechanisms.

We present the first study to investigate the fingerprinting characteristics of extension-injected code in pages' JavaScript namespace and through other observable side-effects like changed cookies. Doing so, we find that many extensions inject JavaScript that pollutes the applications' global namespace by registering variables. It also enables the attacker application to monitor the execution of the injected code by overwriting the JavaScript APIs and capturing execution traces through the *stacktrace*, the set of APIs invoked, etc. Further, extensions also store data on the client side and perform event-driven functionalities that aid in attribution. Through our tests, we find 2,747 Chrome and 572 Firefox extensions to be susceptible to fingerprinting. Unfortunately, none of the existing defense mechanisms prevent extensions from being fingerprinted through our proposed vectors. Therefore, we also suggest potential measures for developers and browser vendors to safeguard the extension ecosystem against such fingerprinting attempts.

CCS CONCEPTS

• **Security and privacy → Web application security**.

KEYWORDS

Client-side Security, Extension Fingerprinting, Browser Extensions

ACM Reference Format:
Shubham Agarwal, Aurore Fass, and Ben Stock. 2024. Peeking through the window: Fingerprinting Browser Extensions through Page-Visible Execution Traces and Interactions. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security (CCS '24), October 14–18, 2024, Salt Lake City, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the rising trend of cookie-less tracking, online trackers are in an arms race with Web Privacy advocates. They continuously compete with various anti-tracking and fingerprinting measures to uniquely identify users on the Web. Recent studies on browser fingerprinting techniques have shown many cookie-less vectors, such as Canvas APIs [5], WebGL APIs [6], and other side channels [1, 29, 32, 44, 51, 58]. These techniques allow a malicious website to effectively harvest client-side information specific to individual users on the Web and further track their activity across websites.

Browser extensions have emerged as one of the most interesting fingerprinting vectors for these trackers in recent times, owing to their unique position in the overall Web ecosystem. Extensions can perform privileged operations implemented in their background script or even execute code in the Web applications' execution context through the content scripts. Tailored to provide a specific set of features to their respective users, browser extensions may also inadvertently reveal personal information about their users, such as their geolocation, background, ethnicity, or social and personal interests [26]. These extensions perform a specific and often highly privileged set of operations, such as DOM-based modifications, changes to cookies, or script injections, to implement their intended functionality at runtime. These operations, however, could also expose them to being uniquely identified by online trackers.

Prior work has shown different ways of uniquely identifying browser extensions. Some of these include Web-accessible resources-based (WAR-based) fingerprints [48], the side effect of code bloating [53], behavior-based fingerprints [26, 50, 54], user-induced side effects [49], and stylesheet-based fingerprints [30]. On the other hand, several mitigation techniques also intend to thwart any fingerprinting attempts based on the above techniques [27, 47, 60]. The studies have led to a cat-and-mouse game of newly introduced fingerprinting vectors and subsequent defense mechanisms. A recent example is the optional usage of randomized runtime identifiers for extensions [41] to thwart WAR-based fingerprinting.

However, ultimately, every extension has some intended functionality, much of which is to interact with the loaded page. This also means that the extension needs to interact with the page through various APIs from the content script. However, when not done carefully, this can leave traces of the extension's execution in the JavaScript namespace of the document, i.e., within the reach of a fingerprinting attacker. These range from client-side storage (e.g., *localStorage* or *cookies*) through the invocation of global APIs

> 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

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

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

Takeaways —Extension Security & Privacy Risks

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Security-Noteworthy Extensions (SNE)

+ Can be **fingerprinted**

- **Contain malware**
  - Designed by malicious actors to harm victims
  - E.g., propagate malware, steal users' credentials, track users
- **Violate the Chrome Web Store policies**
  - E.g., deceive users, promote unlawful activities, lack a privacy policy
- **Contain vulnerabilities**
  - Designed by well-intentioned developers… but contain some vulnerabilities
  - E.g., can lead to user-sensitive data exfiltration

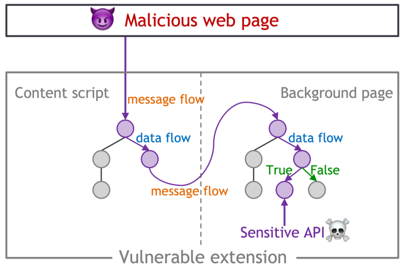
What is in the Chrome Web Store?

Hsu et al. AsiaCCS 2024

- **350M users** installed **SNE** in the last 3 years
- These **SNE** stay in the Chrome Web Store ***for years***
- Extensions have a **short life cycle** in the CWS (60% stay 1 year)
- Critical **lack of maintenance** in the CWS (60% received no update)

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

Agarwal et al. CCS 2024

- 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

Raider-ext/raider

➢ Raider **detects 2,747 fingerprintable extensions | 169M users**

fass@cispa.de   https://aurore54f.github.io   @AuroreFass

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

- 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