CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Browser Extension (In)Security

Aurore Fass

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

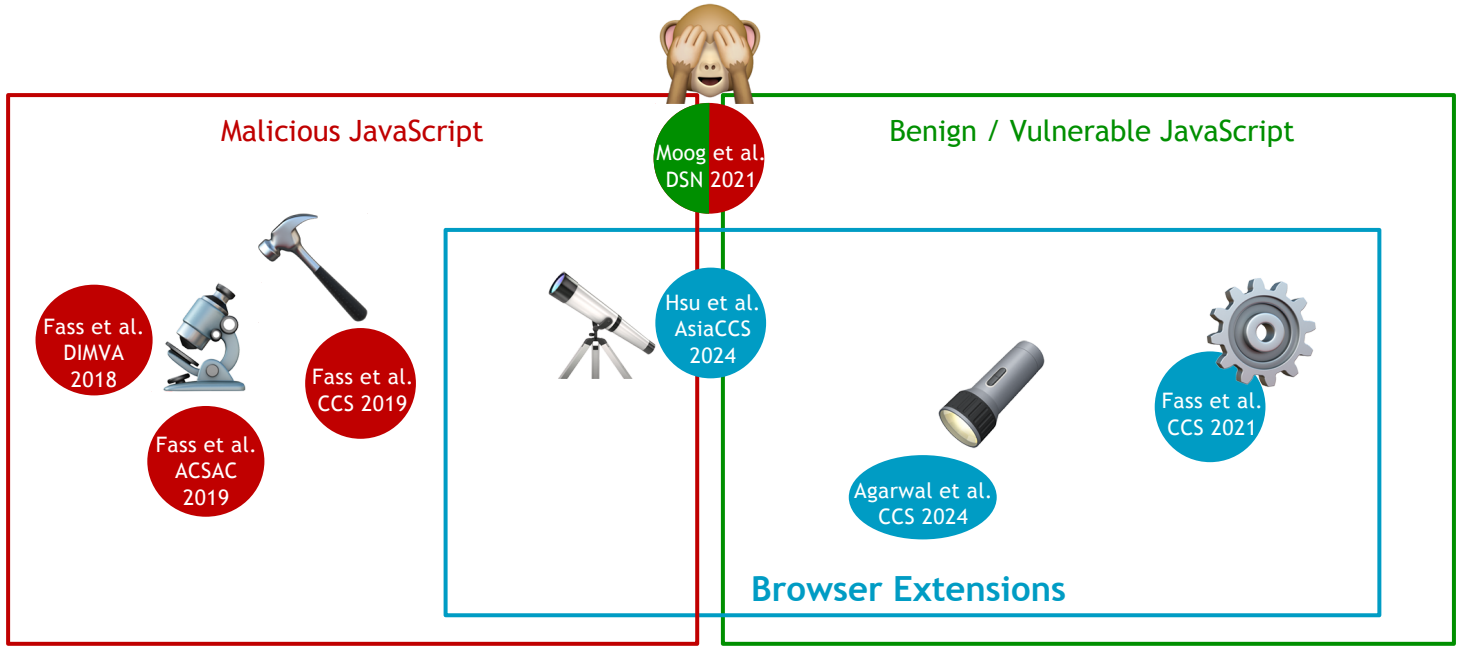
UNIMORE, Modena, December 18th, 2024

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

Research Work: Web Security & Privacy



Ruth et al. IMC 2022

Internet Measurements

Izhikevich et al. IMC 2023

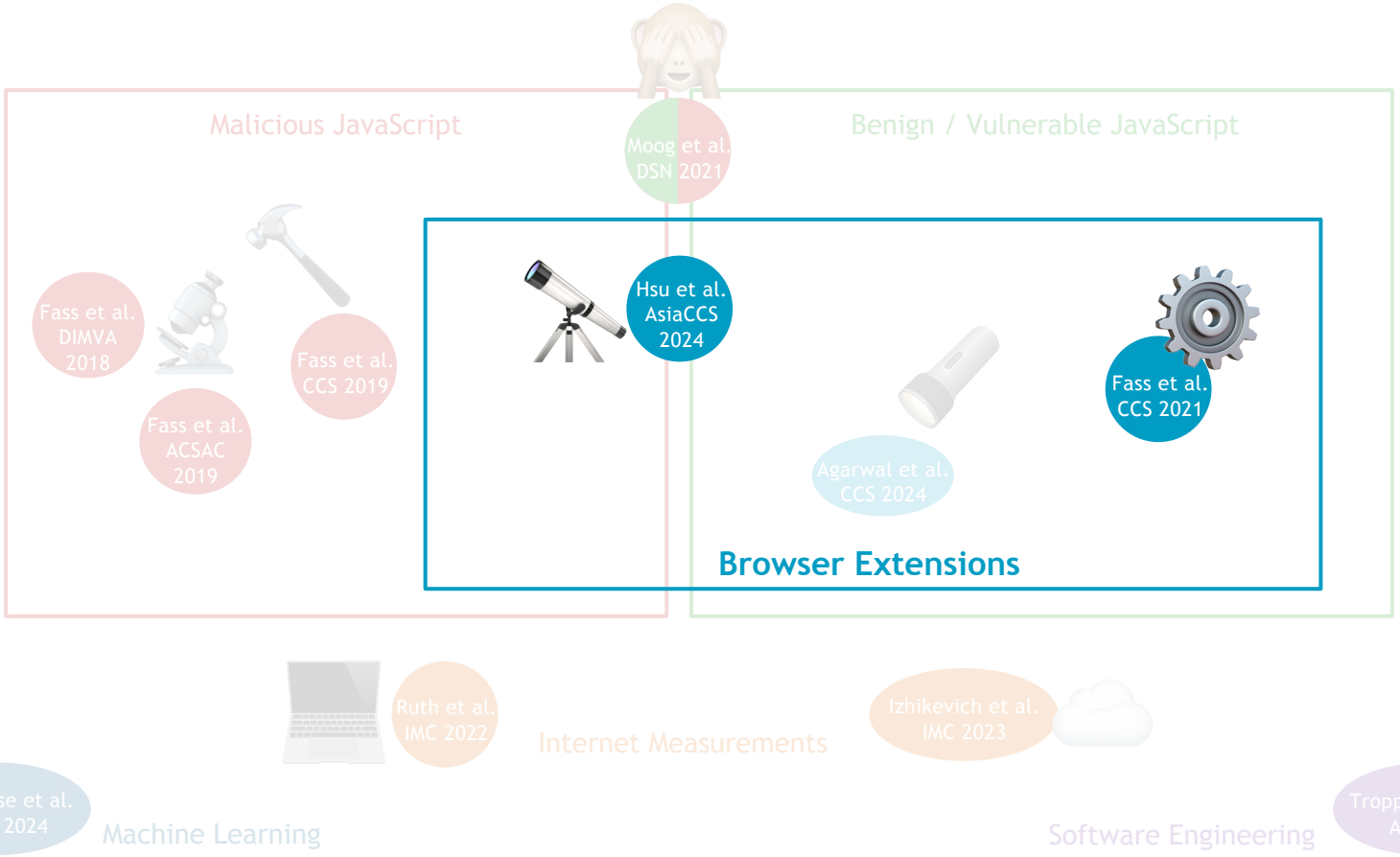
Apruzzese et al. AISec 2024

Machine Learning

Software Engineering

Troppmann et al. ASE 2024

Research Work: Web Security & Privacy



Outline

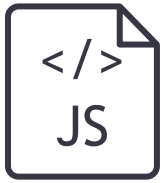
- Background
  - Malicious/vulnerable JavaScript
  - Browser extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - Extension analysis & SNE detection
- Detecting Vulnerable Extensions
  - Threat models & examples
  - Static analysis of extensions (JavaScript) & examples
  - Case studies, results, and potential defense strategies

Outline

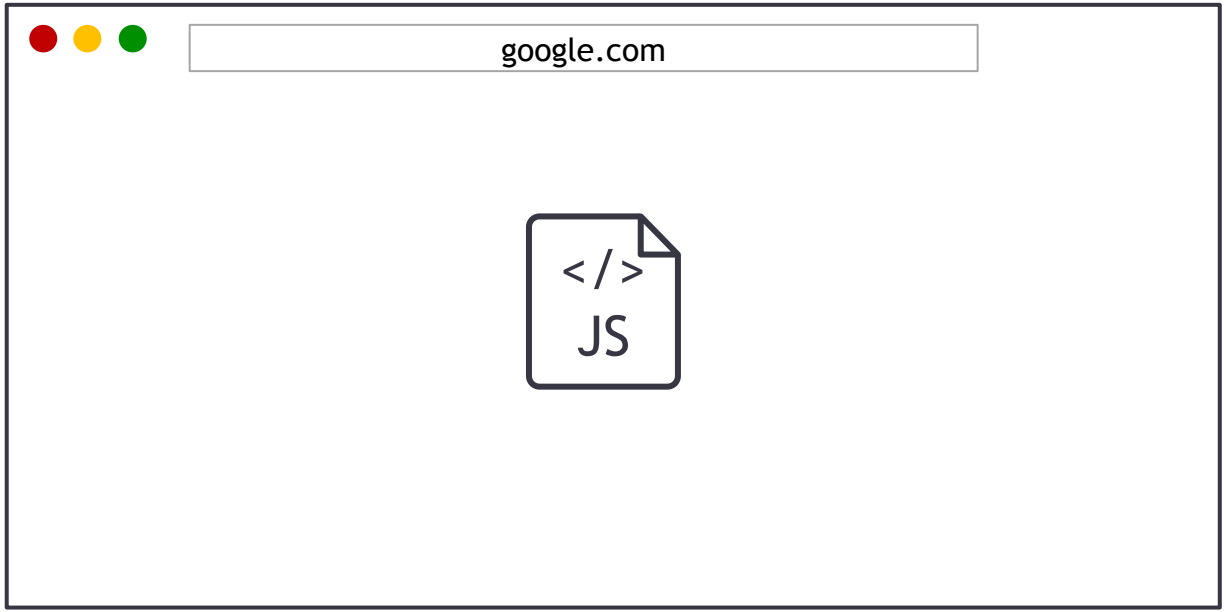
- Background
  - Malicious/vulnerable JavaScript
  - Browser extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - Extension analysis & SNE detection
- Detecting Vulnerable Extensions
  - Threat models & examples
  - Static analysis of extensions (JavaScript) & examples
  - Case studies, results, and potential defense strategies

CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

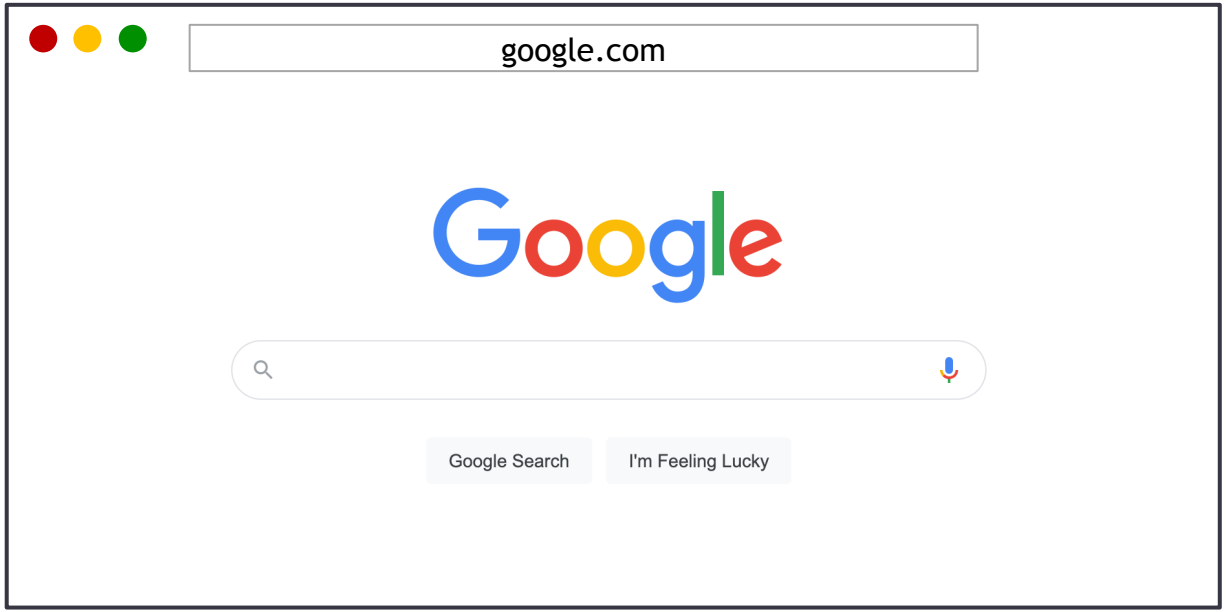
Background — JavaScript



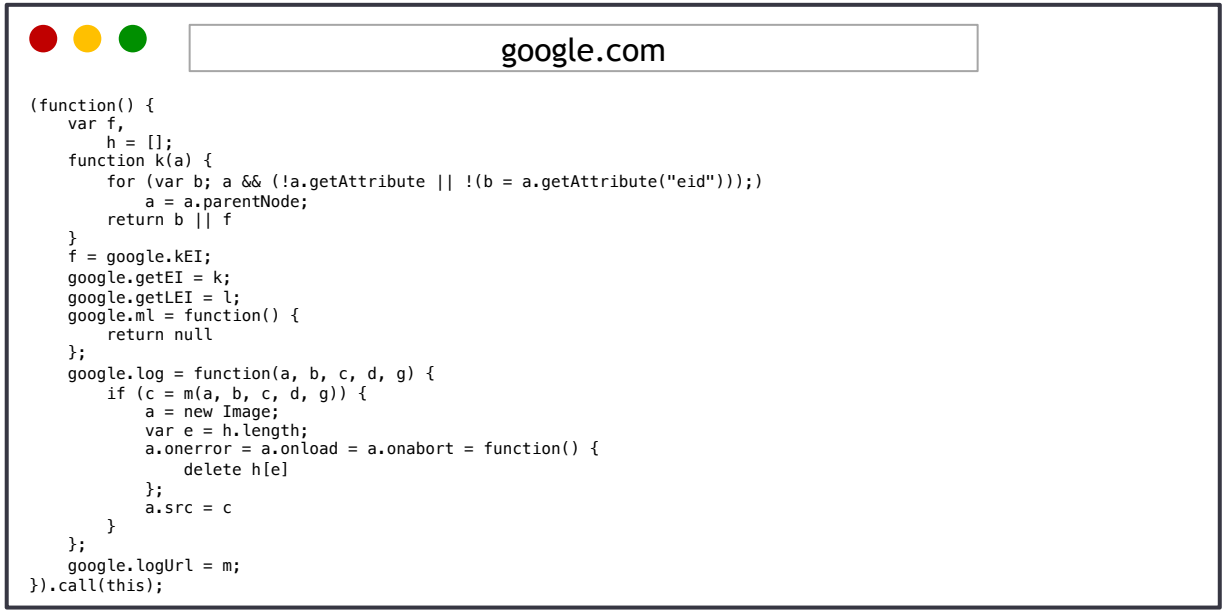
Background — JavaScript



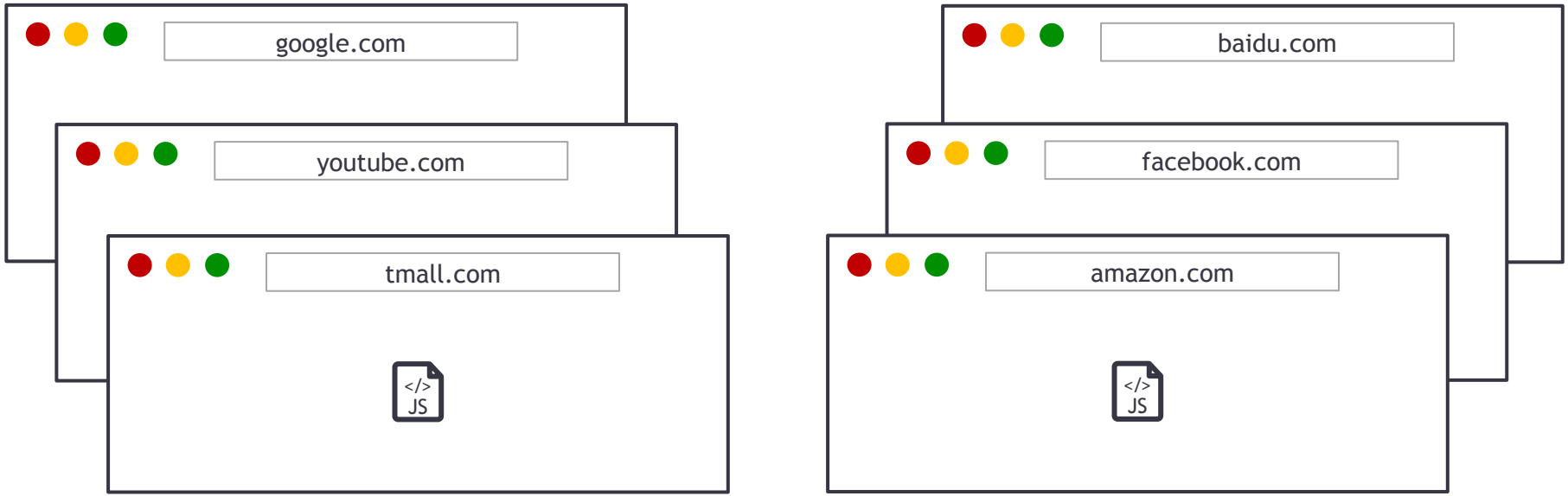
Background — JavaScript



Background — JavaScript



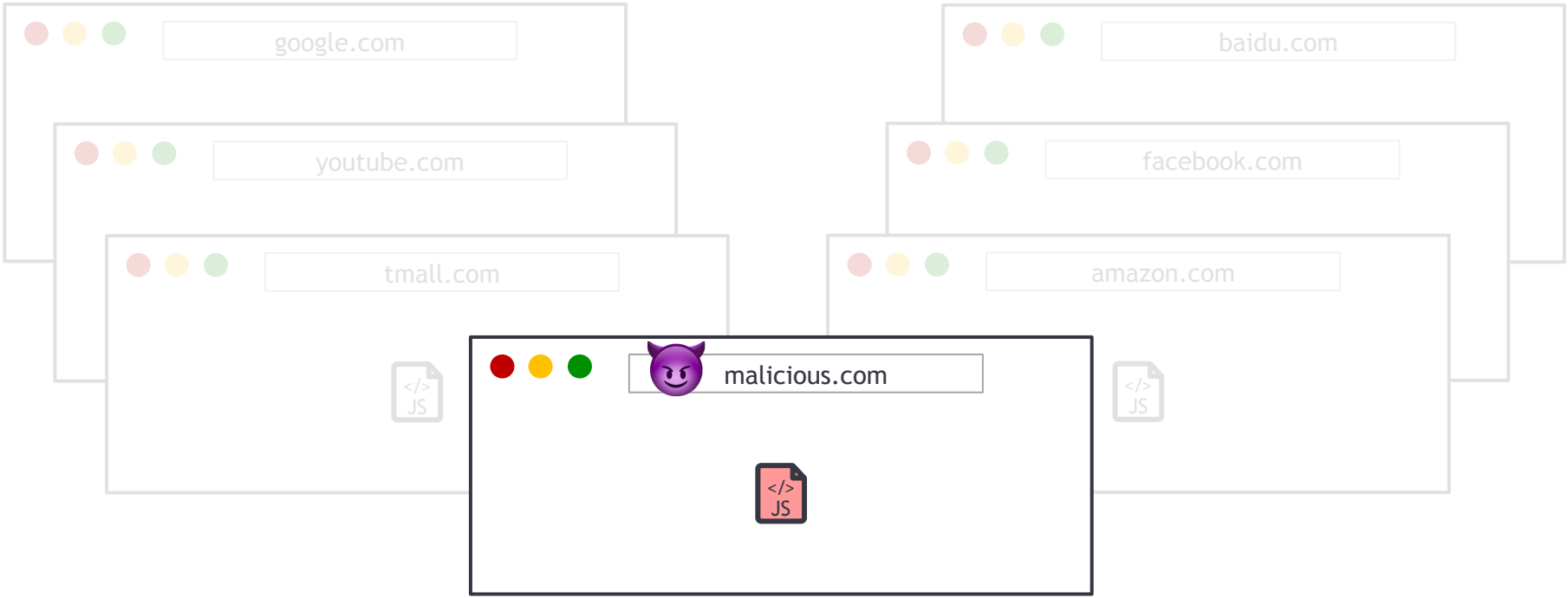
Background — JavaScript



JavaScript usage: > 99% websites

https://w3techs.com

Background — JavaScript



JavaScript usage: > 99% websites

https://w3techs.com

Background — JavaScript: Basis for Attacks

Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims

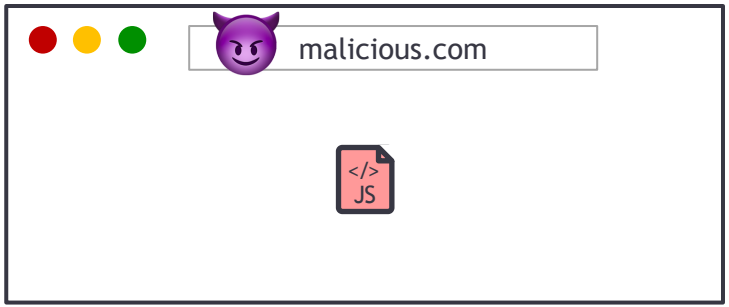
  → e.g., exploiting vulnerabilities, stealing sensitive user data

Background — JavaScript: Basis for Attacks

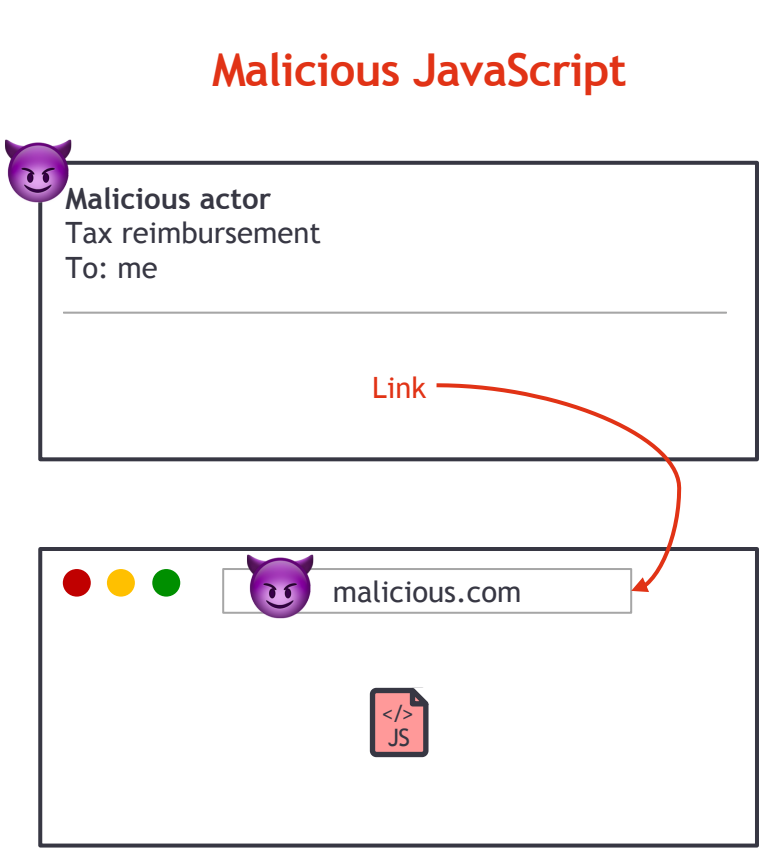
Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims

  → e.g., exploiting vulnerabilities, stealing sensitive user data

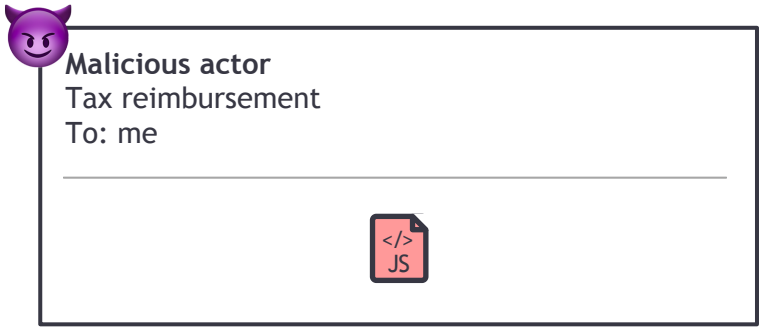


Background — JavaScript: Basis for Attacks

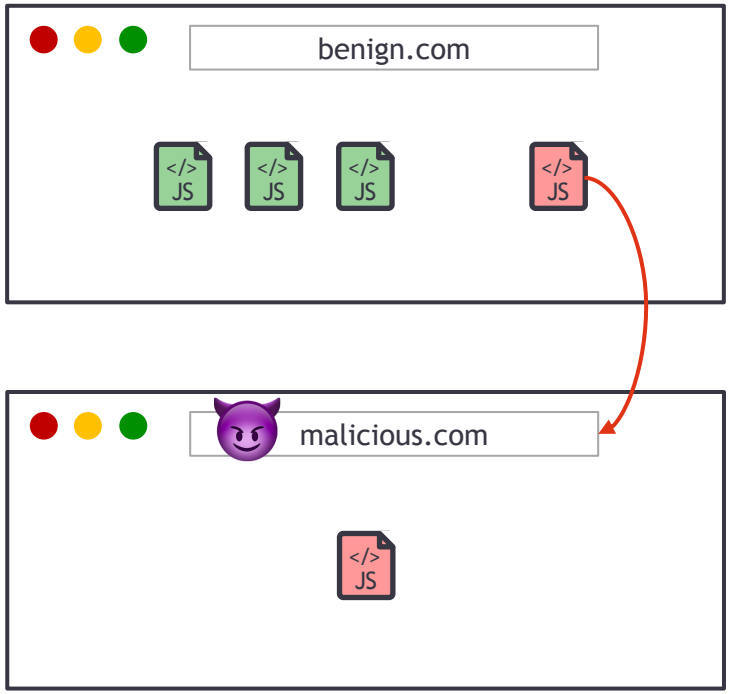


Background — JavaScript: Basis for Attacks

Malicious JavaScript



Background — JavaScript: Basis for Attacks



Background — JavaScript: Basis for Attacks

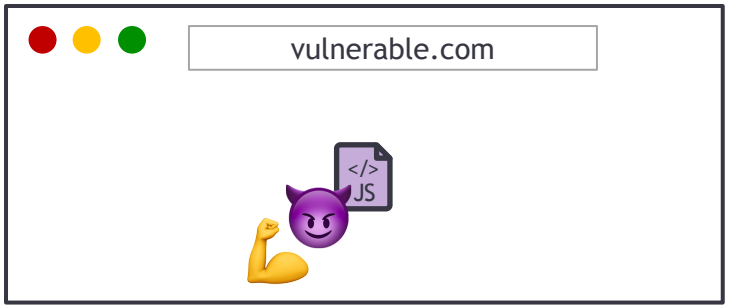
Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims, e.g.,
  - exploiting vulnerabilities to download & execute malware,
  - stealing sensitive user data

-but-buggy Benign JavaScript

- Designed by well-intentioned developers
- … but contains some vulnerabilities

  → can be exploited by malicious actors



Background — JavaScript: Basis for Attacks

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims, e.g.,
  - exploiting vulnerabilities to download & execute malware,
  - stealing sensitive user data

-but-buggy

Benign JavaScript

- Designed by well-intentioned developers
- … but contains some vulnerabilities
  - can be exploited by malicious actors

What about if you install a *malicious* or *vulnerable* browser extension?

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

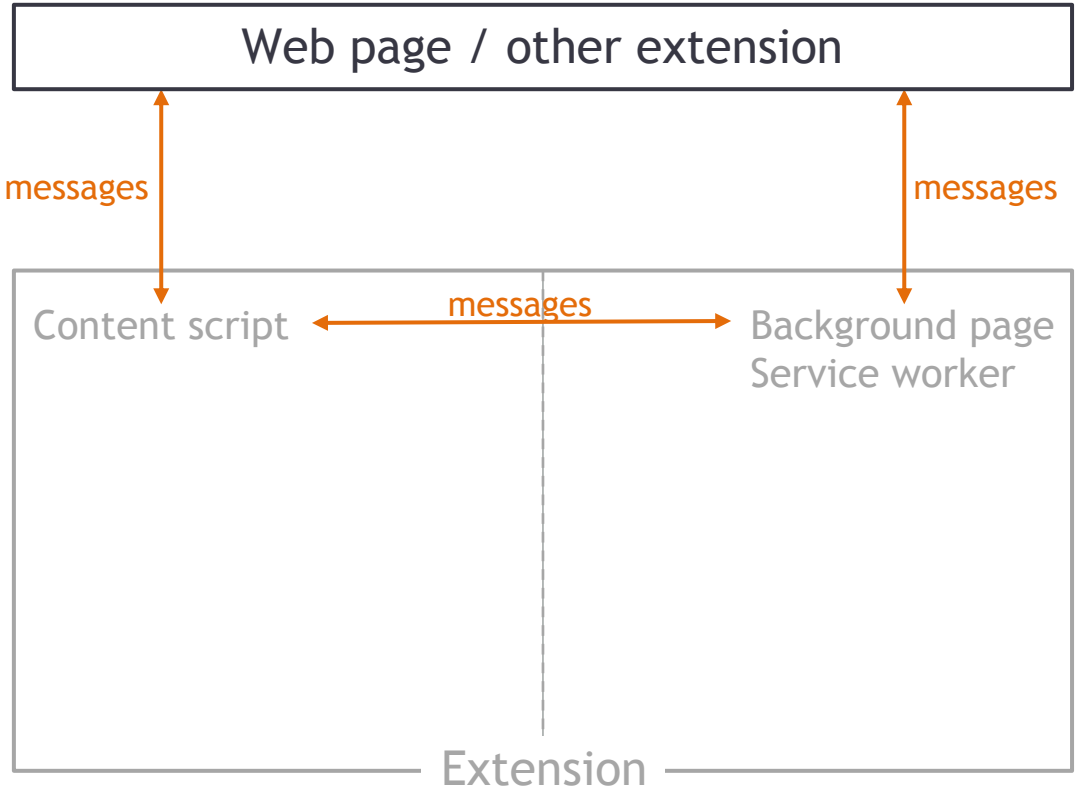
- https://developer.chrome.com/docs/extensions/mv3/declare_permissions/

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

Background — Extension Architecture

- Background page (BP) / Service worker (SW):
  - 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
  - Malicious/vulnerable JavaScript
  - Browser extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - Extension analysis & SNE detection
- Detecting Vulnerable Extensions
  - Threat models & examples
  - Static analysis of extensions (JavaScript) & examples
  - Case studies, results, and potential defense strategies

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)

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

→ 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…

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

- **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 · Manda Tran, Stanford University · Aurore Fass, Stanford University, CISPA Helmholtz Center for Information Security

ABSTRACT

CCS CONCEPTS

KEYWORDS

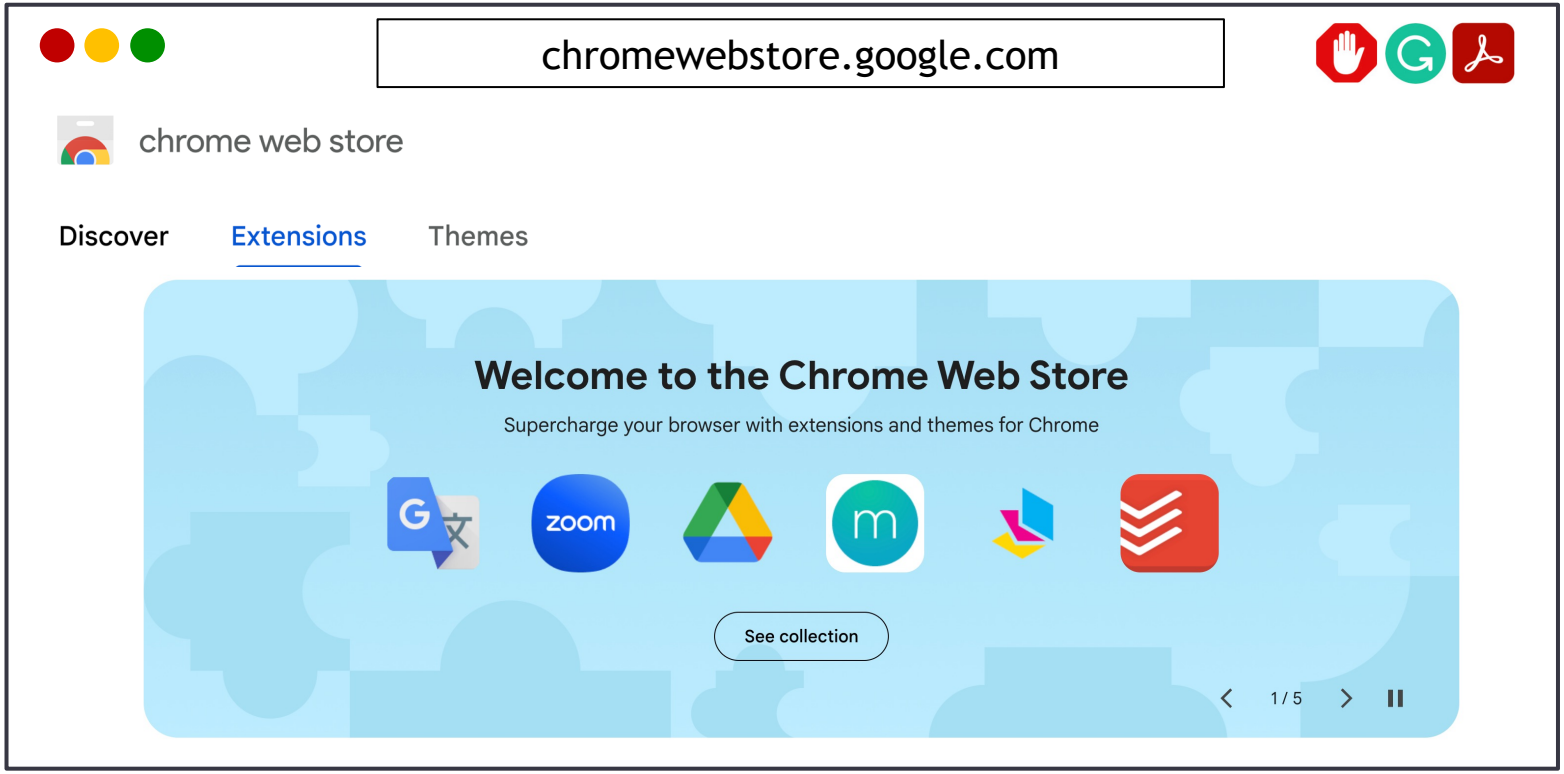
1 INTRODUCTION

> What is in the Chrome Web Store?

Hsu et al.
AsiaCCS
2024

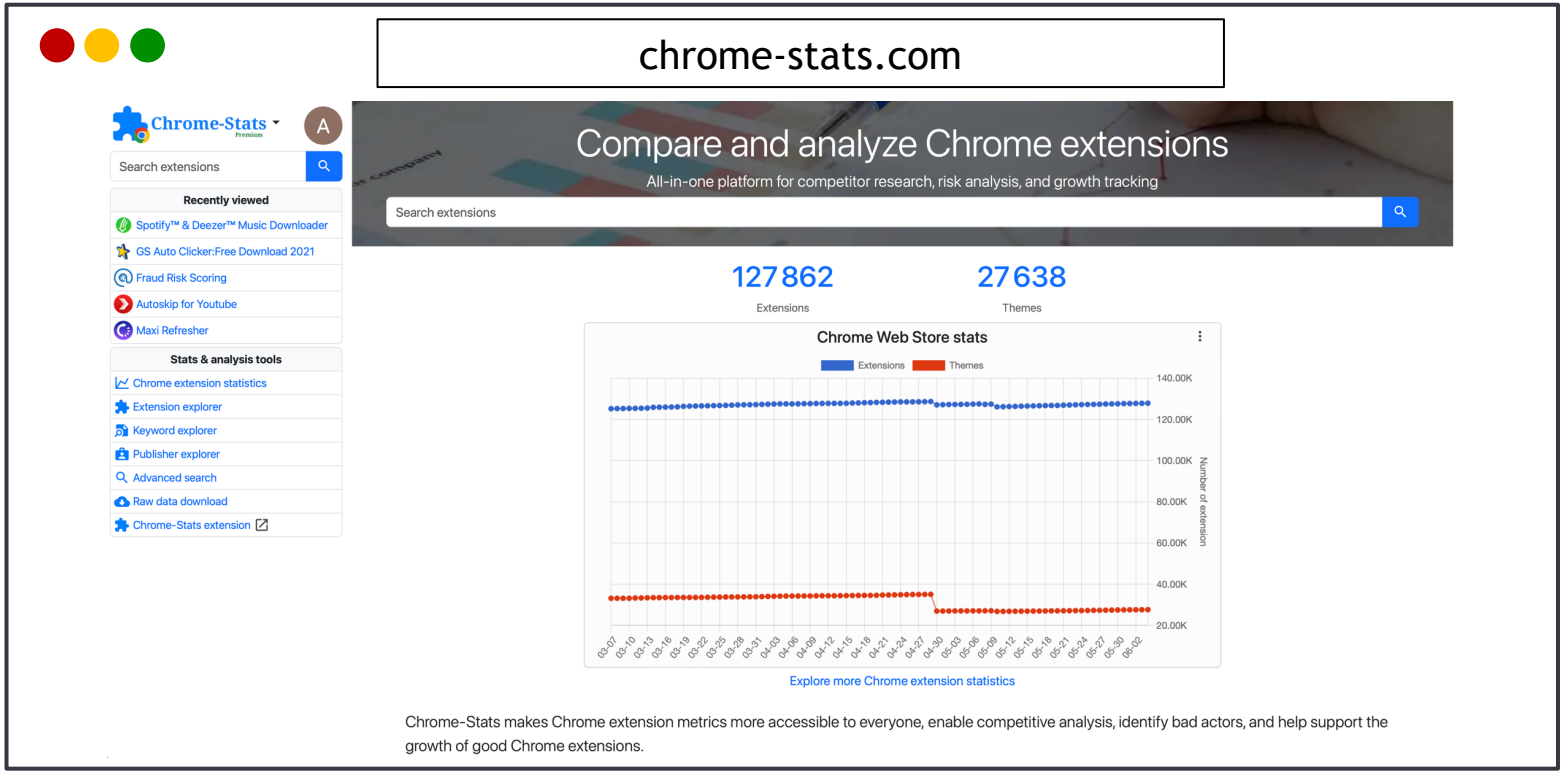
In *ACM AsiaCCS* 2024. Sheryl Hsu, Manda Tran, and Aurore Fass

How to Install Extensions or SNE?

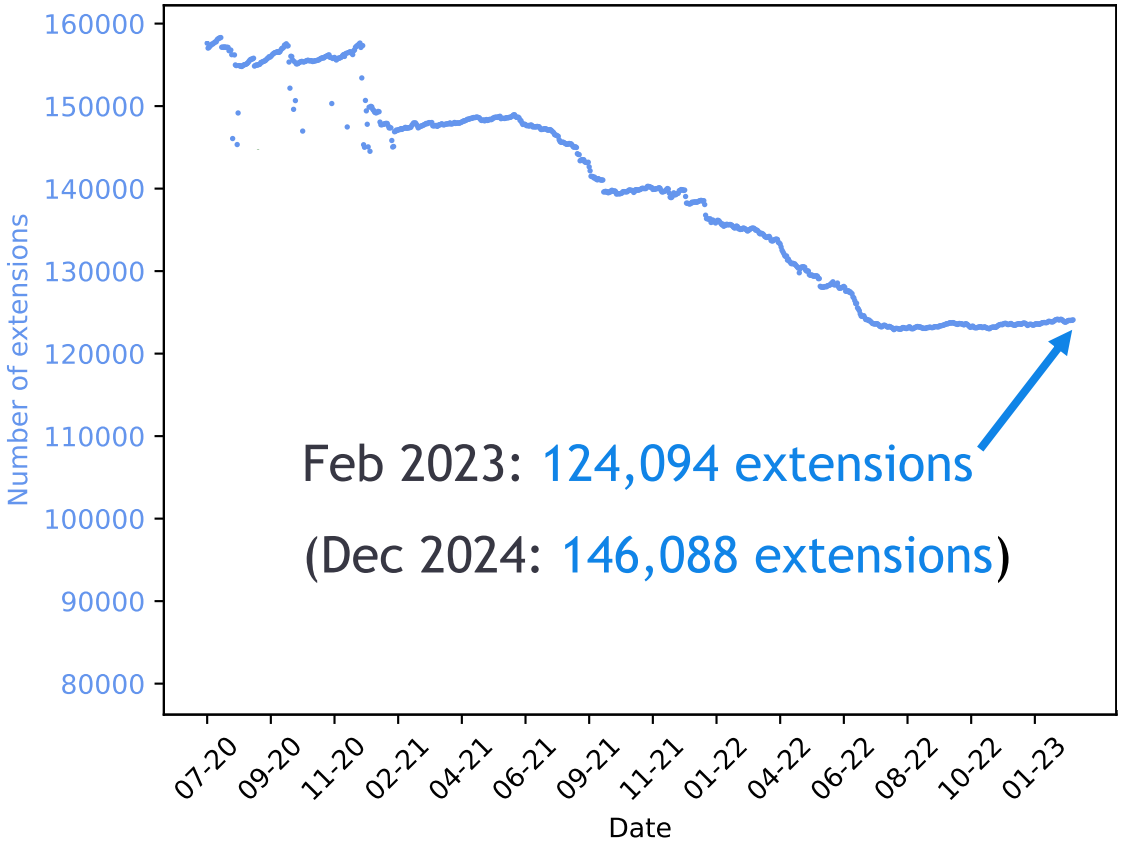


How to Install Extensions or SNE?

Browser Extension Collection: Chrome-Stats



Number of Extensions in the Chrome Web Store

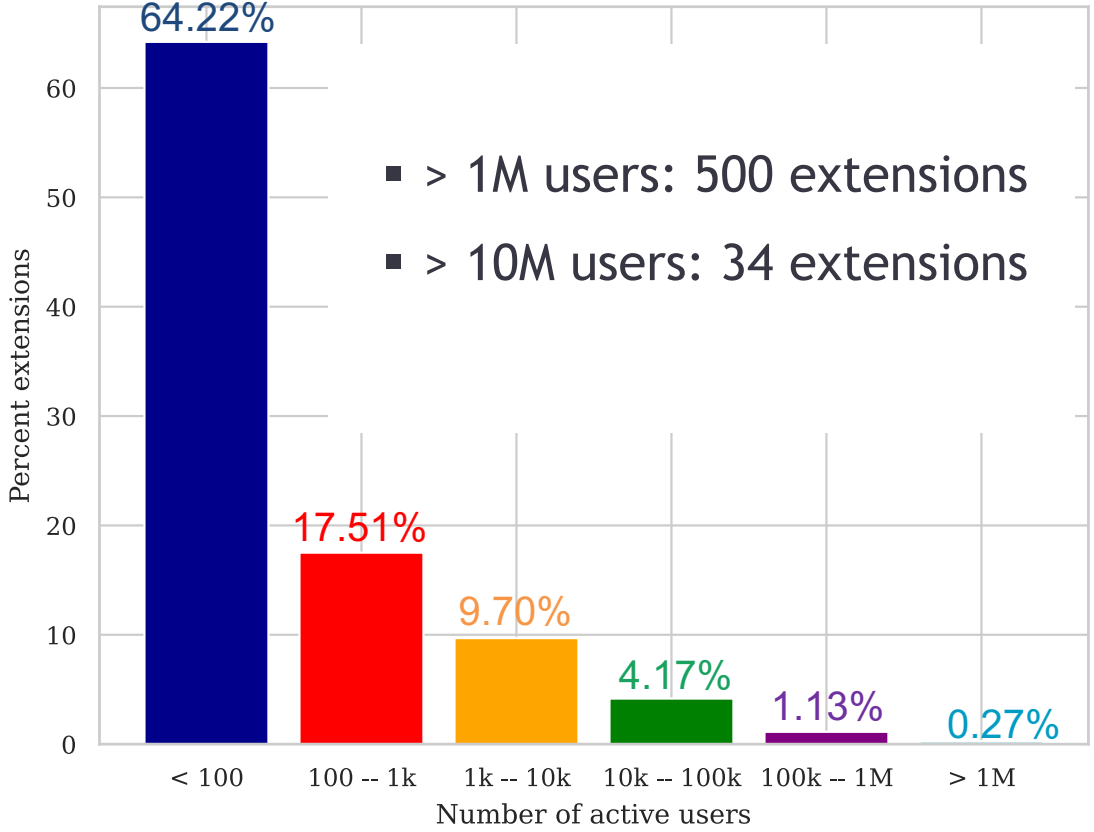


Every month:

- -3,775 extensions removed
- +2,687 extensions added

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

Breakdown of Extension Users



Extension Users

The "**number of users**" on the CWS for a given extension corresponds to:

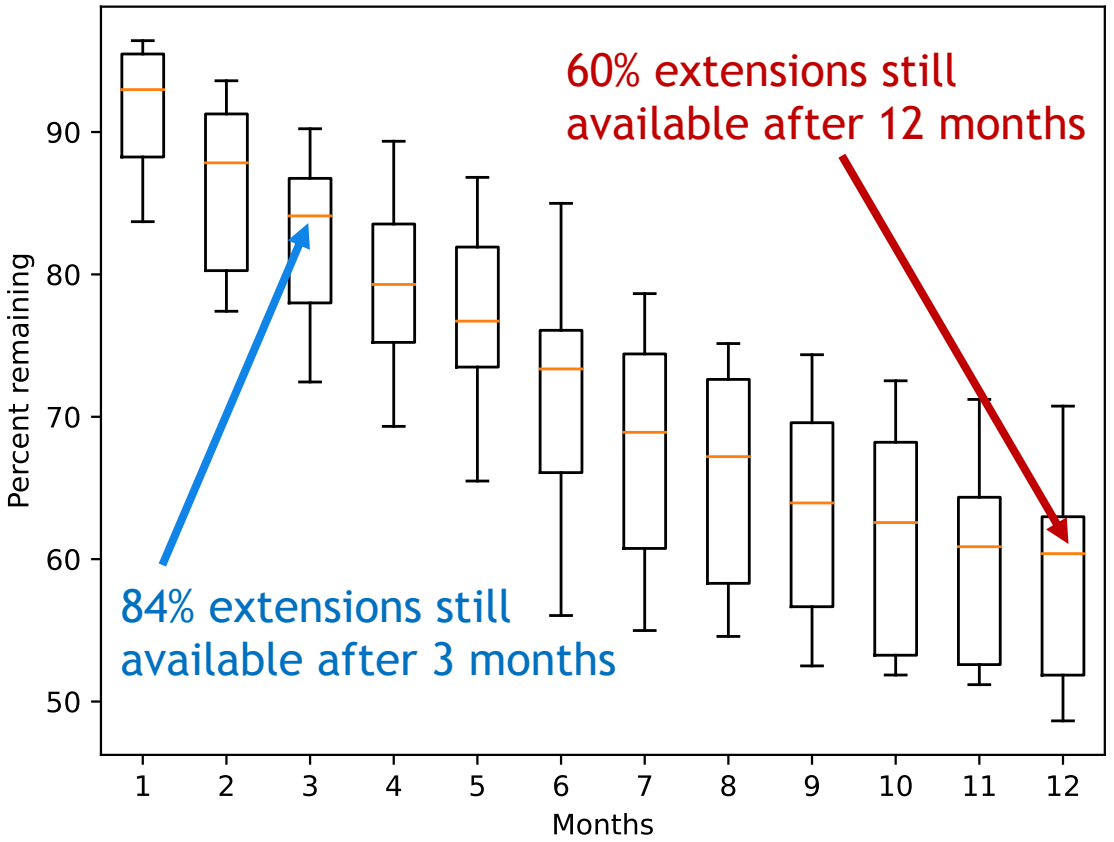
> *"the number of Chromes with the extension installed that are active and checking in to [their] update servers over the previous seven days only, not for all time. It is not equal to the sum of historic installs minus the sum of historic uninstalls"*

~ Chrome Web Store Developer Support

Life Cycle of Extensions

Methodology:

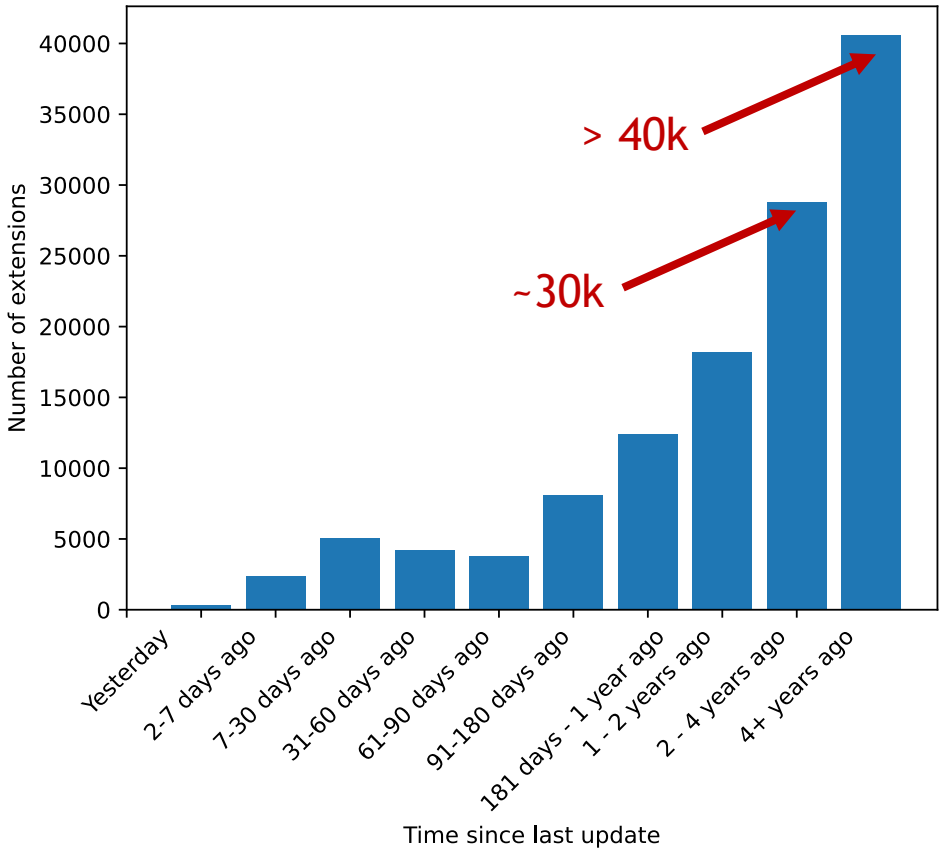
- Collected extensions added to the CWS in Jan—Dec 2021
- Computed the percentage of those extensions still in the CWS 1, 2, …, 12 months later



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

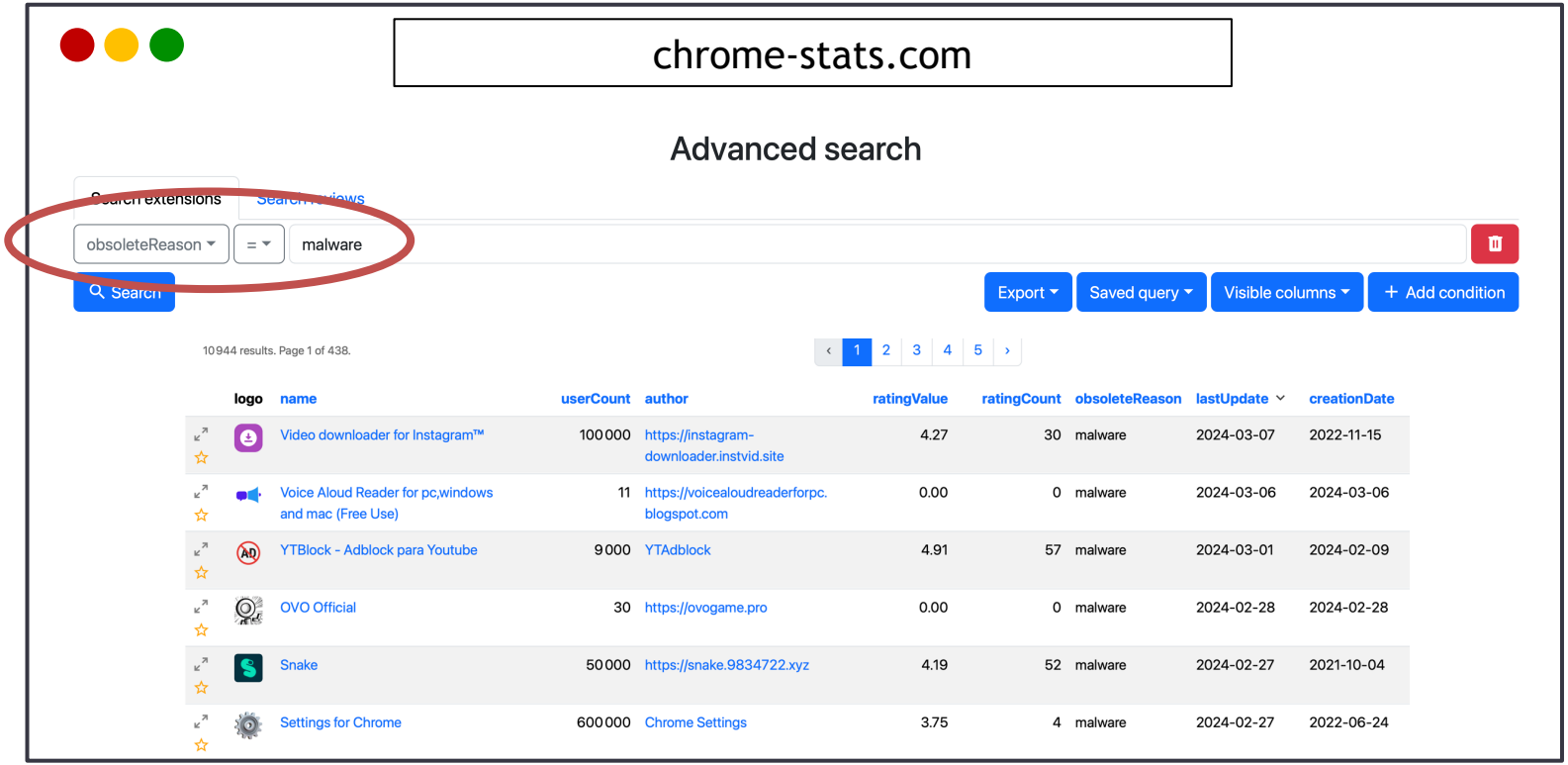
➢ Analyses on the CWS should be **run regularly**

Extension Maintenance and Security



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

Malicious Extension Collection: Chrome-Stats



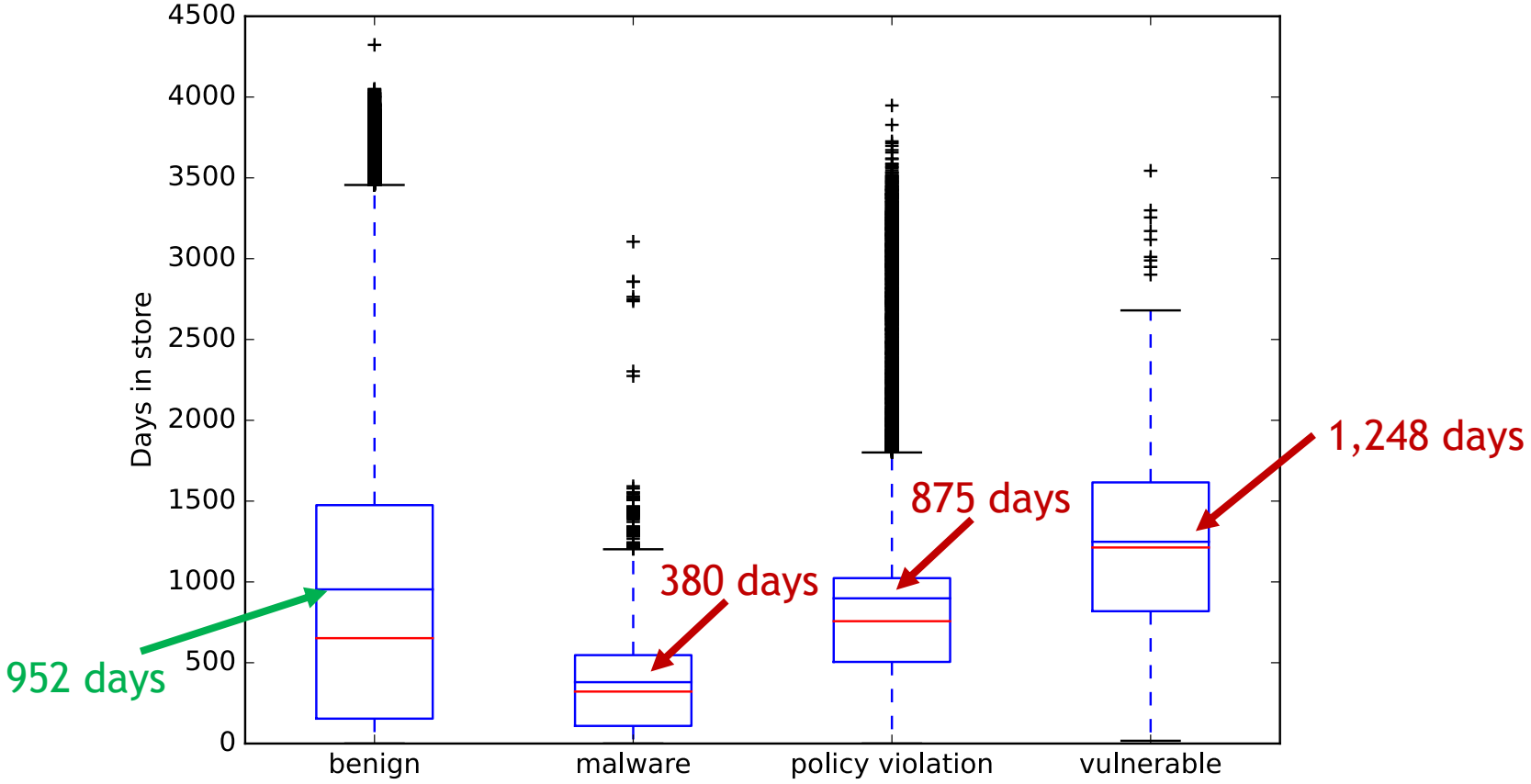
logo	name	userCount	author	ratingValue	ratingCount	obsoleteReason	lastUpdate	creationDate
	Video downloader for Instagram™	100000	https://instagram-downloader.instvid.site	4.27	30	malware	2024-03-07	2022-11-15
	Voice Aloud Reader for pc,windows and mac (Free Use)	11	https://voicealoudreaderforpc.blogspot.com	0.00	0	malware	2024-03-06	2024-03-06
	YTBlock - Adblock para Youtube	9000	YTAdblock	4.91	57	malware	2024-03-01	2024-02-09
	OVO Official	30	https://ovogame.pro	0.00	0	malware	2024-02-28	2024-02-28
	Snake	50000	https://snake.9834722.xyz	4.19	52	malware	2024-02-27	2021-10-04
	Settings for Chrome	600000	Chrome Settings	3.75	4	malware	2024-02-27	2022-06-24

Browser Extension Collection: Chrome-Stats

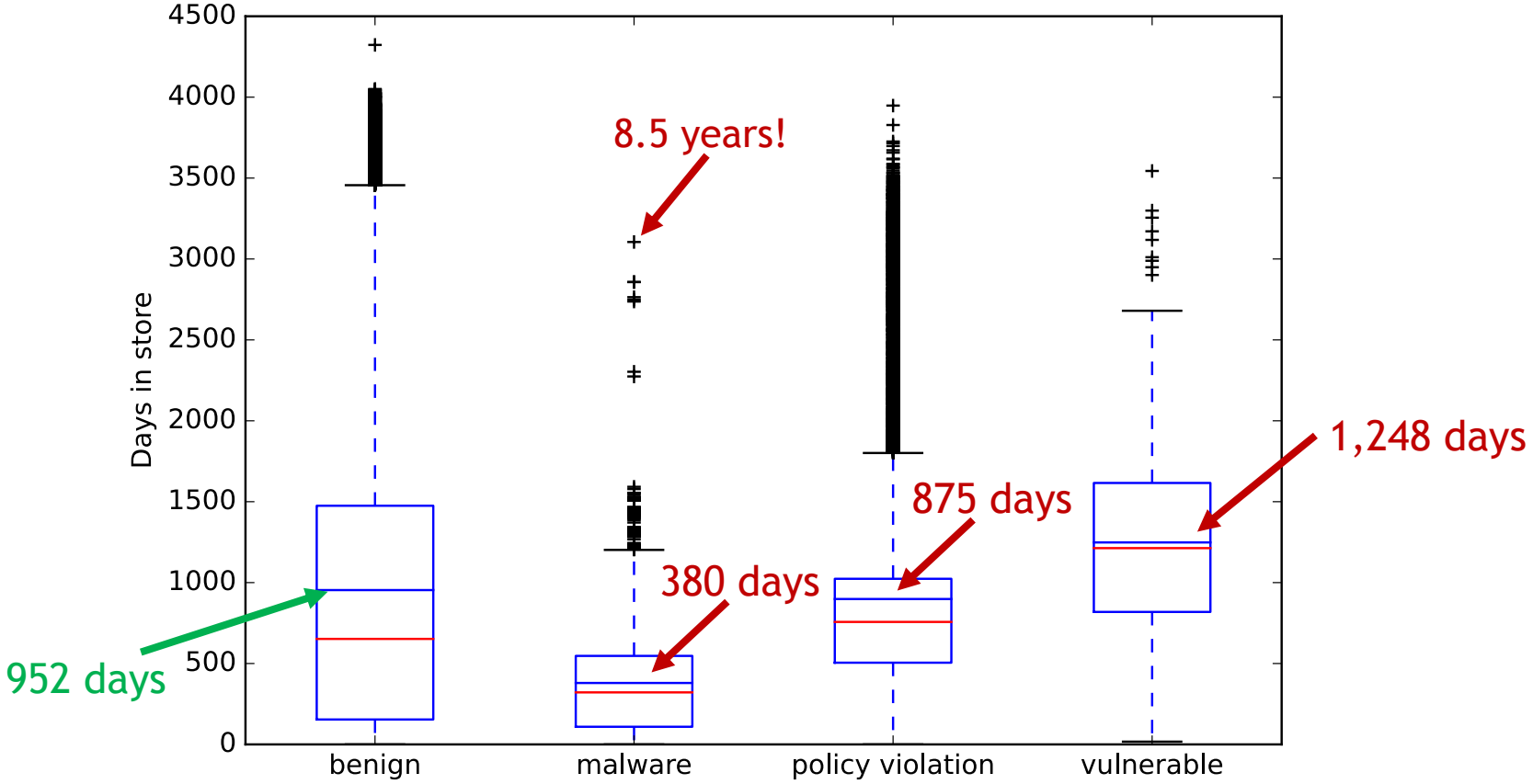
Category	#Extensions Metadata collected	#Extensions Code collected	When collected
SNE	26,014	16,377	Before May 1, 2023
- Malware-containing	10,426	6,587	Before May 1, 2023
- Policy-violating	15,404	9,638	Before May 1, 2023
- Vulnerable [1]	184	152	March 16, 2021
Benign extensions	226,762	92,482	Before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

[1] Fass et al., CCS 2021

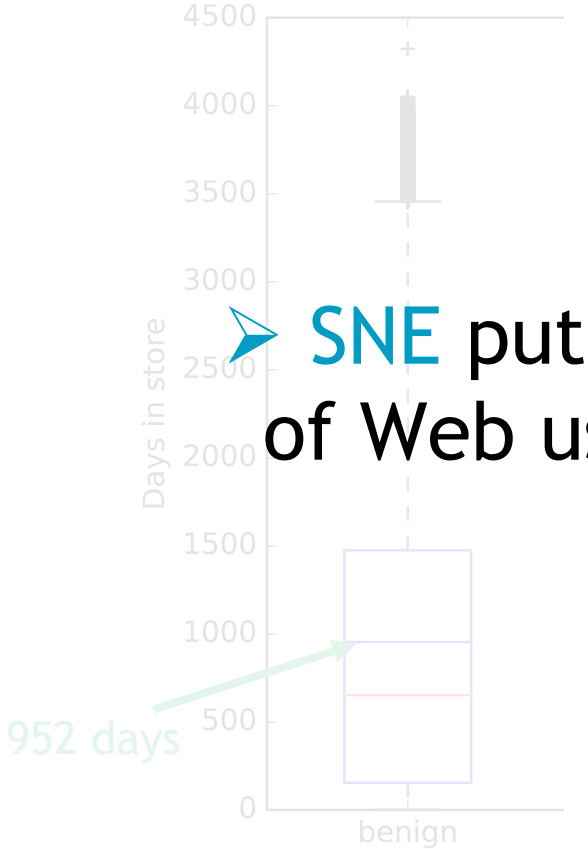
Number of Days in the CWS



Number of Days in the CWS

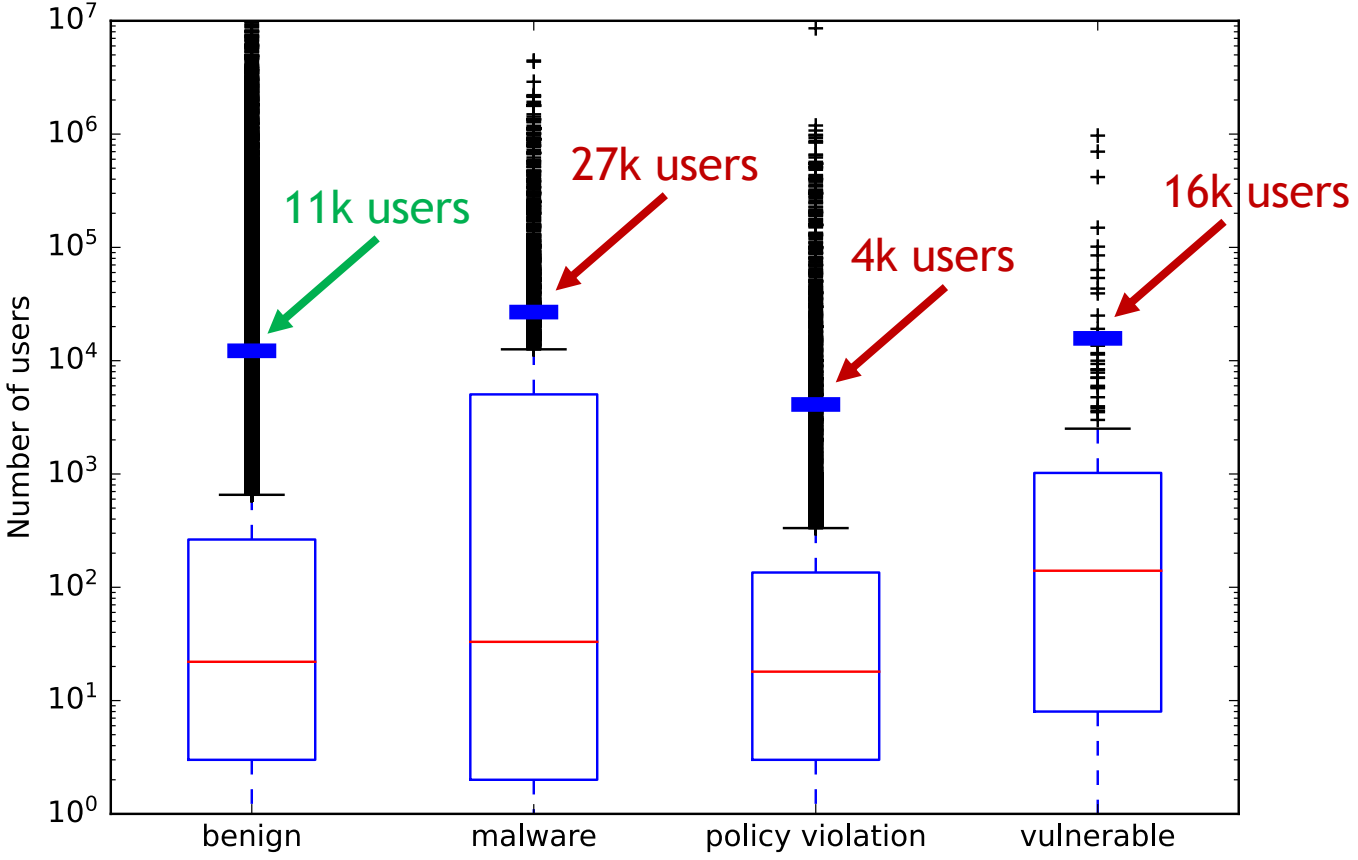


Number of Days in the CWS

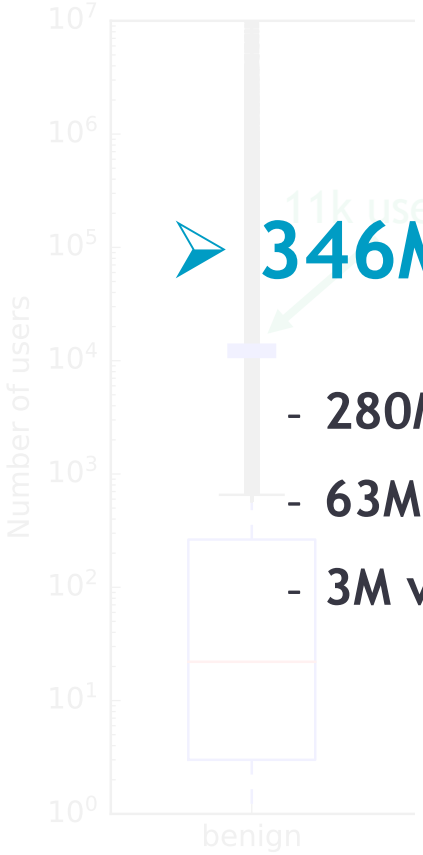


- SNE put the security & privacy of Web users **at risk *for years***

Number of Users



Number of Users



- **346M users installed SNE**
  (in the last 3 years):
  - **280M malware-containing**
  - **63M policy-violating**
  - **3M vulnerable extensions**

Detecting SNE with Clustering-based Approaches

- Source-code comparison across extensions (*ssdeep* fuzzy hash)
- Clustering similar extensions together (i.e., 100% *ssdeep* overlap)
- 3,270 clusters with ⟦2; 1,397⟧ extensions (20,822 extensions clustered)

Detecting SNE with Clustering-based Approaches

- 3,270 clusters:
  - 2,296 clusters contain just benign extensions
  - 321 clusters only SNE
    - 14 clusters with > 100 SNE and 2 with > 863 SNE each
    - ➢ Analyzing extensions for similarities could enable to **detect SNE**
  - 653 clusters of benign (5,552 extensions) and SNE (5,126)
    - ➢ Extensions in a cluster with SNE should be **flagged for more analyses**

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
  - Malicious/vulnerable JavaScript
  - Browser extensions
- Investigating Security-Noteworthy Extensions (SNE)
  - SNE definition
  - Extension analysis & SNE detection
- Detecting Vulnerable Extensions
  - Threat models & examples
  - Static analysis of extensions (JavaScript) & examples
  - Case studies, results, and potential defense strategies

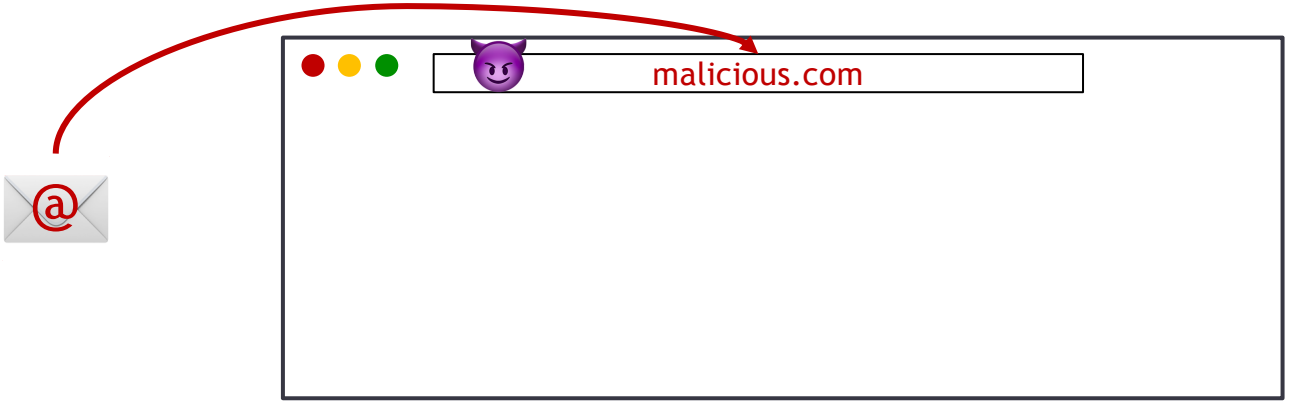
CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Analysis of Vulnerable Extensions: Web Attacker

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

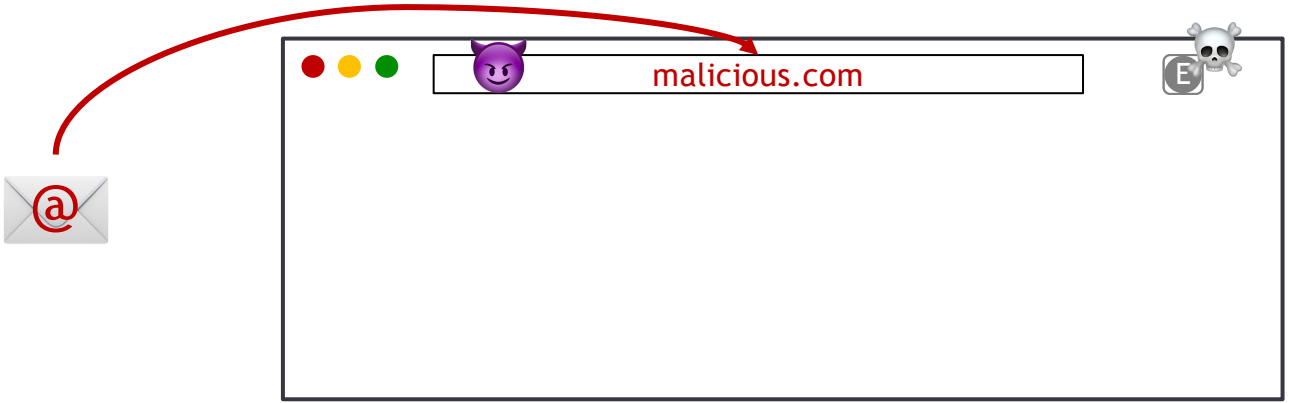
Analysis of Vulnerable Extensions: Web Attacker

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



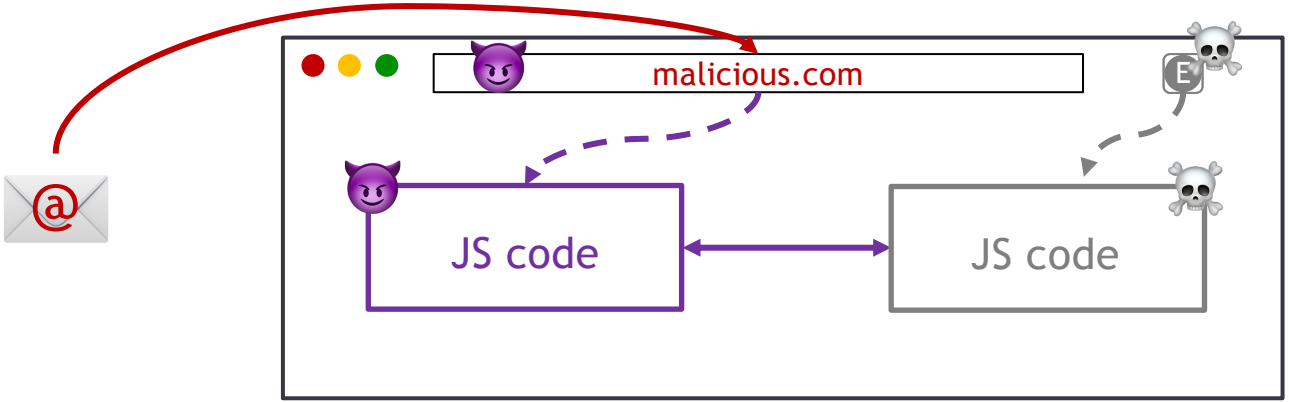
Analysis of Vulnerable Extensions: Web Attacker

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



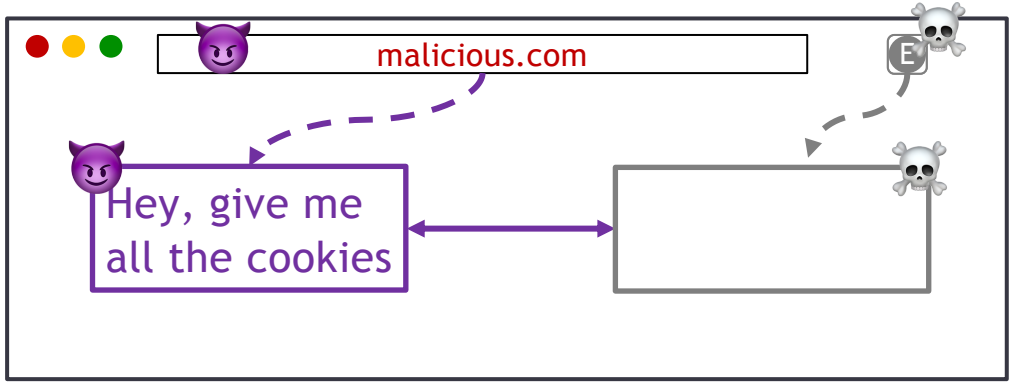
Analysis of Vulnerable Extensions: Web Attacker

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



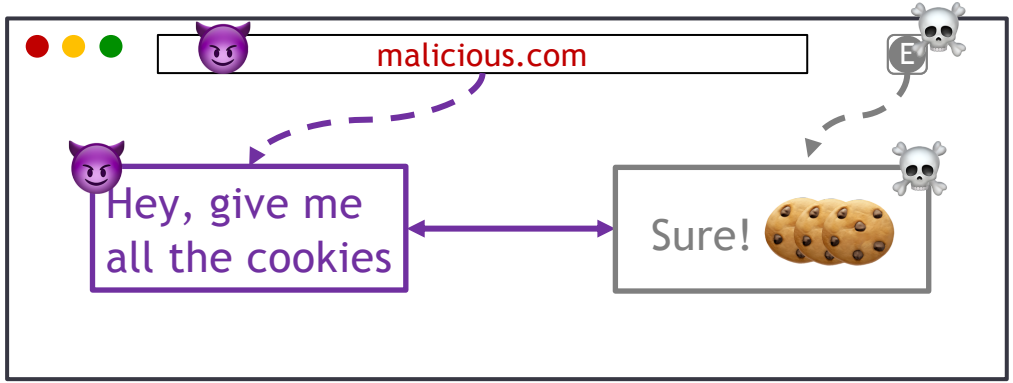
Analysis of Vulnerable Extensions: Web Attacker

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



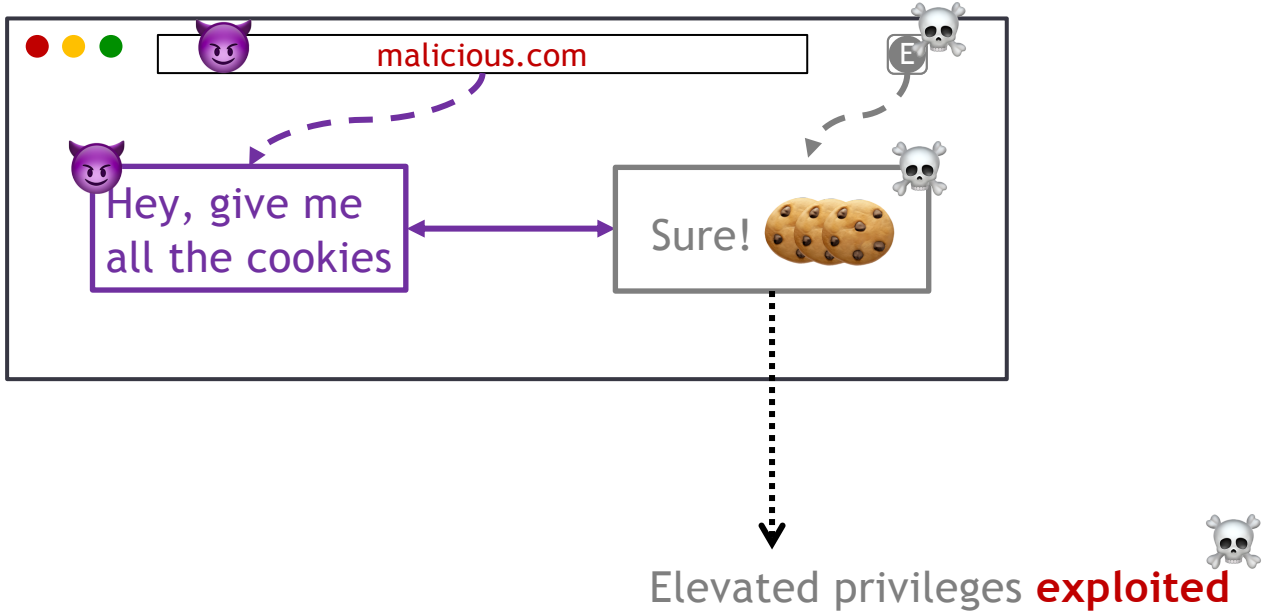
Analysis of Vulnerable Extensions: Web Attacker

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



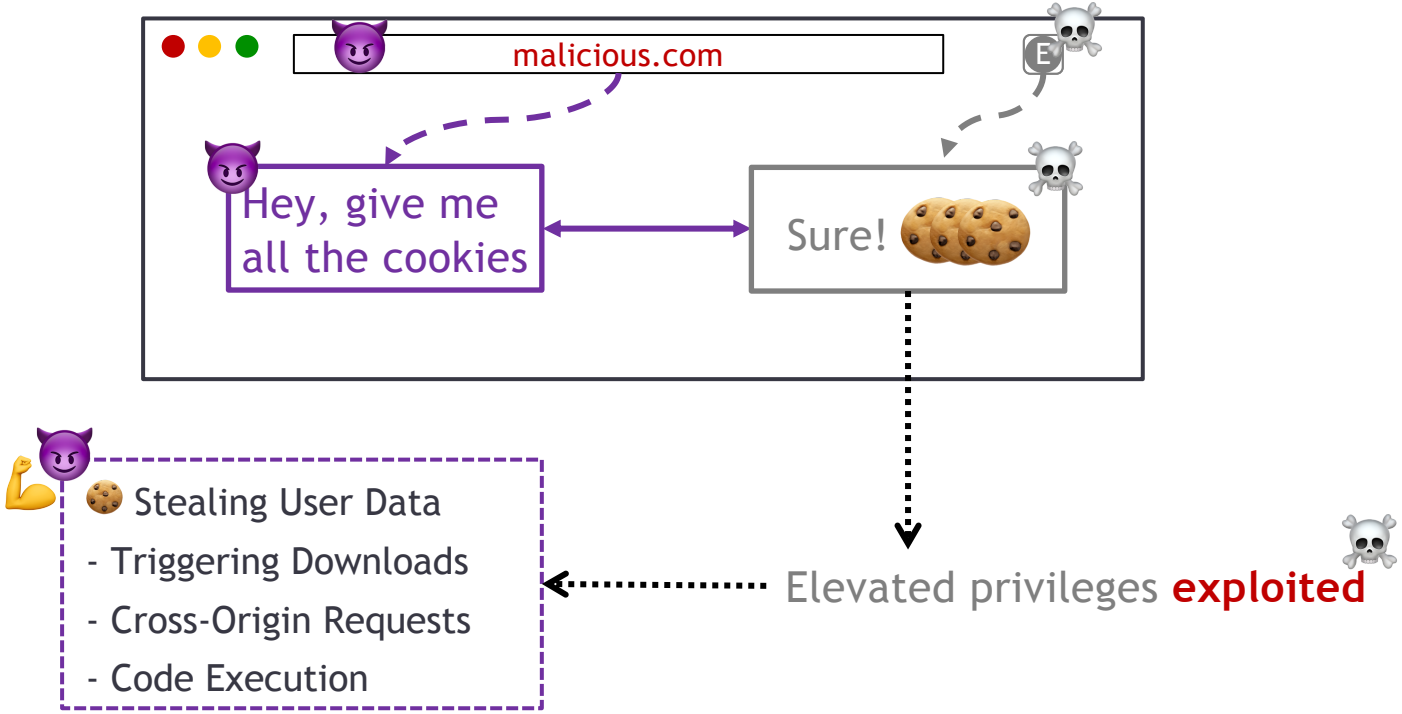
Analysis of Vulnerable Extensions: Web Attacker

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



Analysis of Vulnerable Extensions: Web Attacker

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



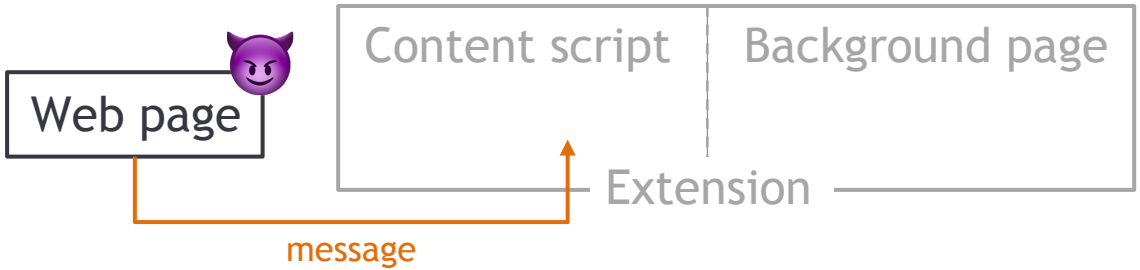
Background — postMessage

- 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

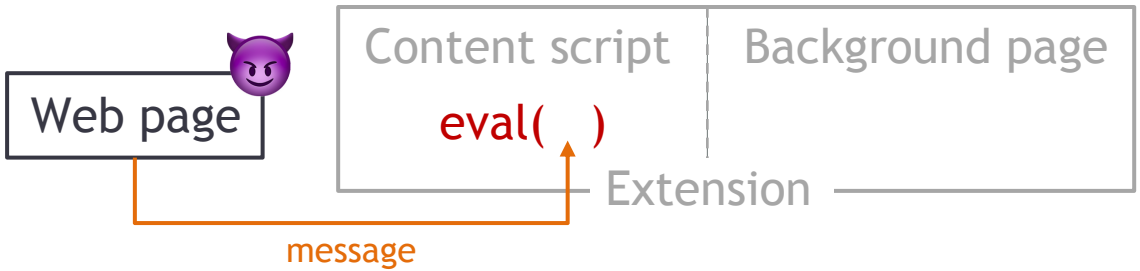
Simplified Example of a Vulnerability

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




})
```



Simplified Example of a Vulnerability

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

    eval(event.data);

})
```

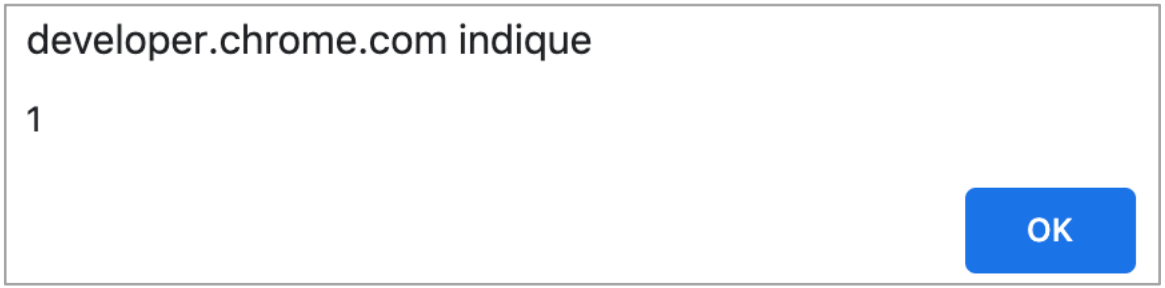


Simplified Example of a Vulnerability

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

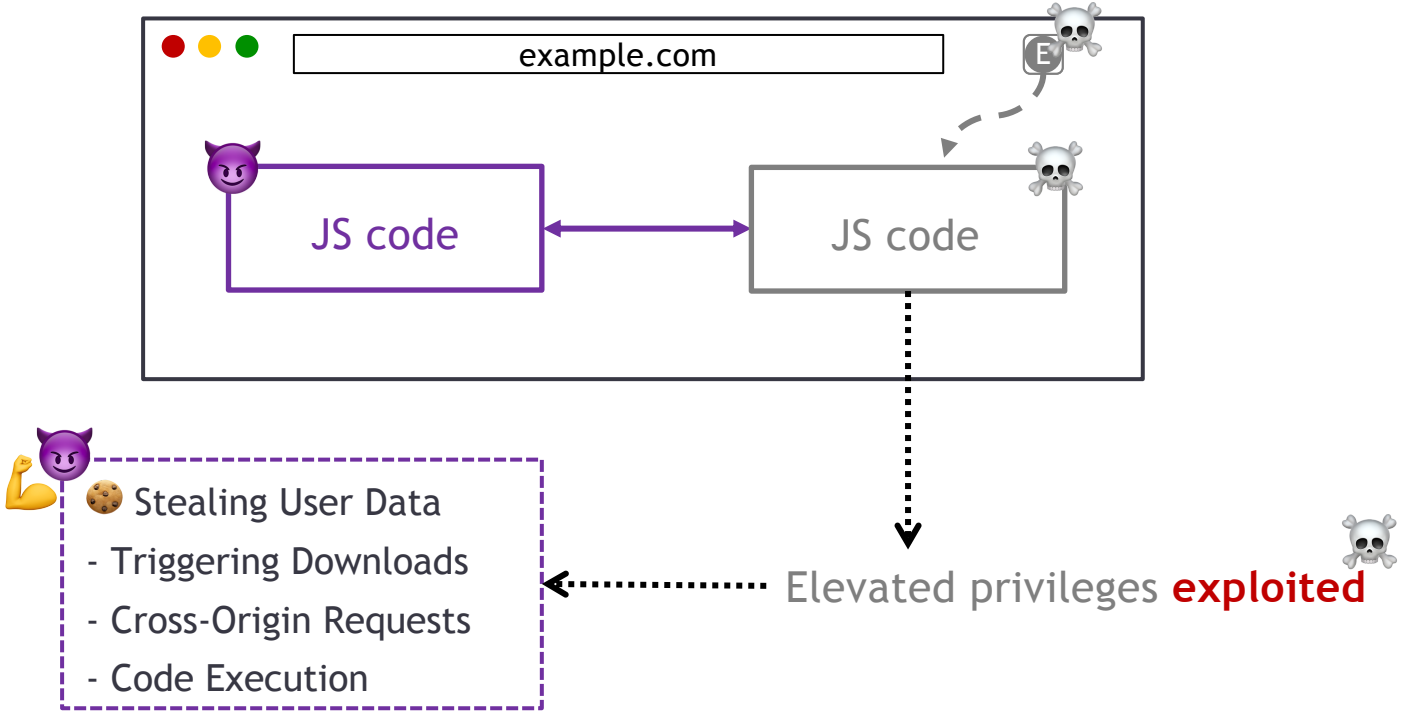
```
// Attacker code = from the targeted web page
postMessage("alert(1)", "*")
```

malicious payload



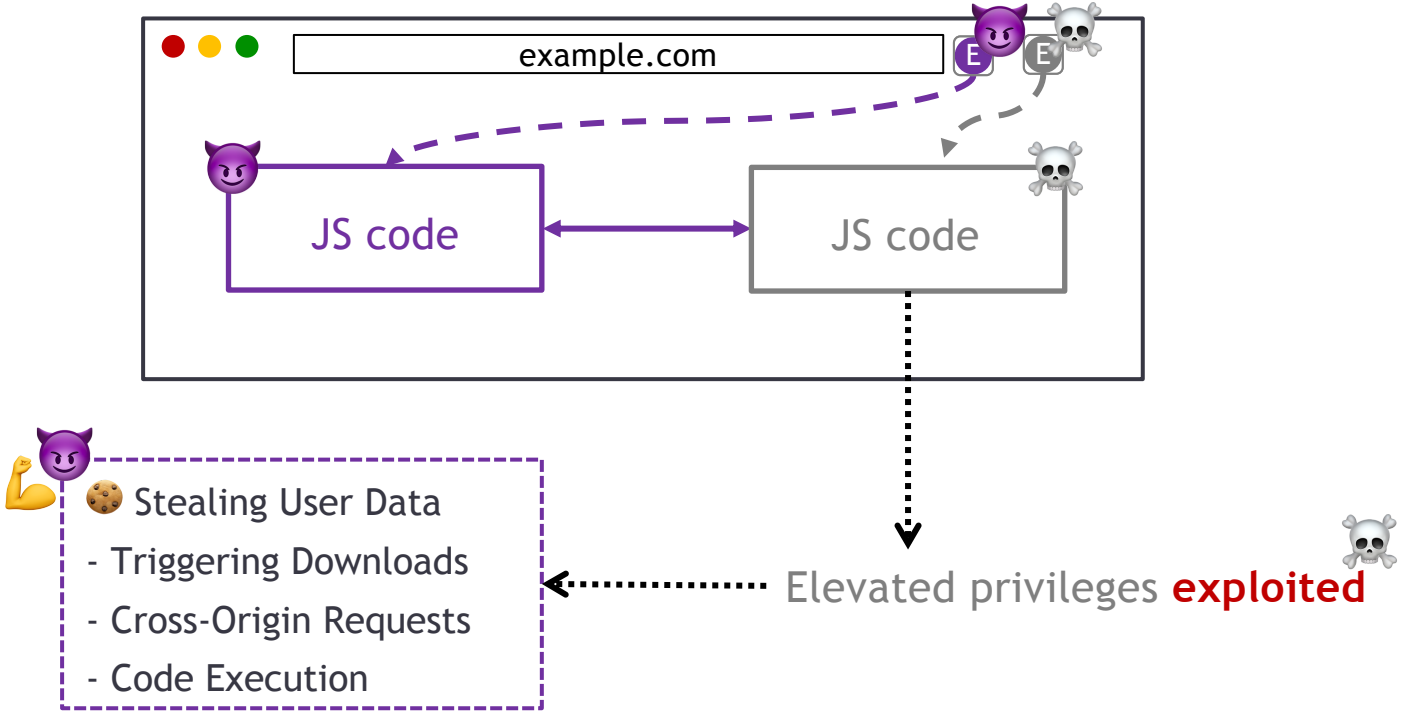
Analysis of Vulnerable Extensions: Confused Deputy

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



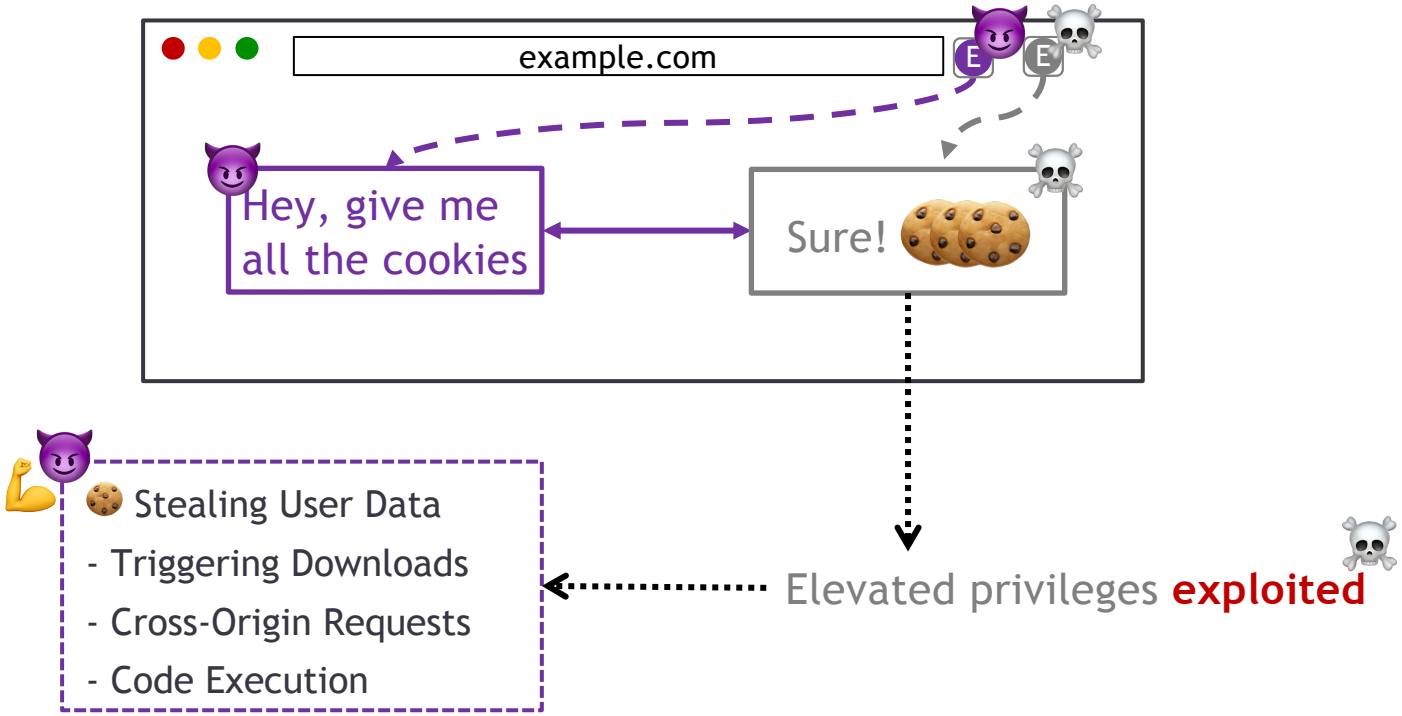
Analysis of Vulnerable Extensions: Confused Deputy

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



Analysis of Vulnerable Extensions: Confused Deputy

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



Exploiting Vulnerable Extensions: Confused Deputy

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

```
// Background page code of Extension B
chrome.runtime.onMessageExternal.addListener(
  function(request, sender, sendResponse) {
      chrome.bookmarks.getTree(function(data) {
          sendResponse(data);
      });
});
```

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

> 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

> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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

Content script

Background page
Service worker

Vulnerable extension

Detecting Vulnerable Extensions


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

Fass et al. 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

Malicious web page

Content script

Background page
Service worker

Vulnerable extension

Detecting Vulnerable Extensions


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

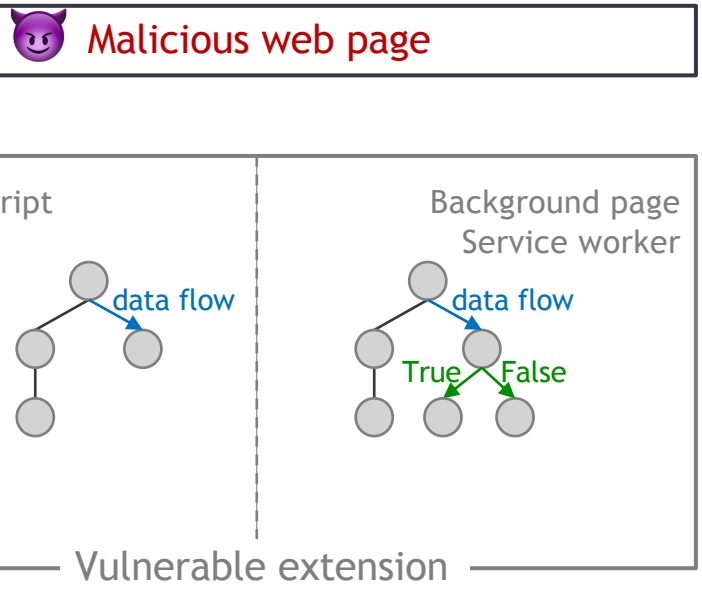
Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

CISPA Helmholtz Center for Information Security

Fass et al. 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 JS code abstraction

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

Detecting Vulnerable Extensions

Fass et al. CCS 2021

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY


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

Abstract

Introduction

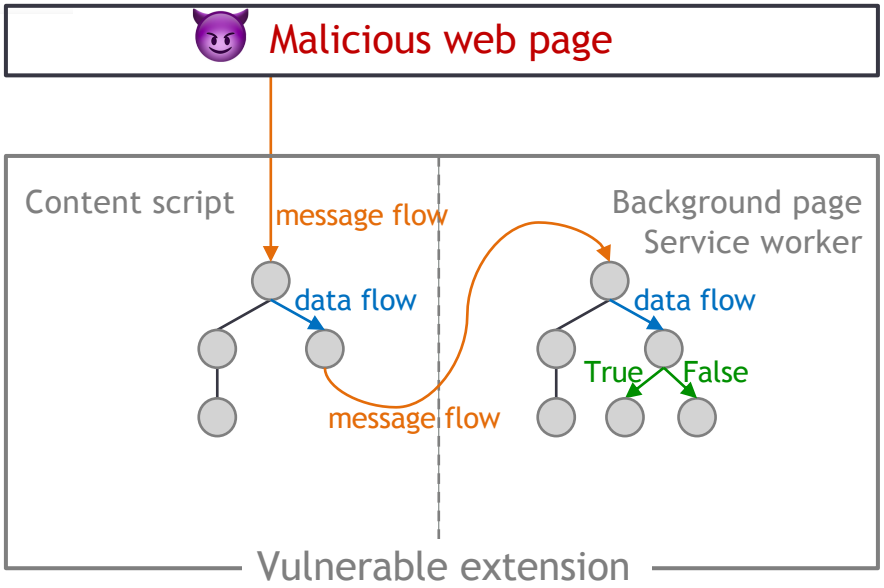
CCS Concepts

Keywords

ACM Reference Format

> 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

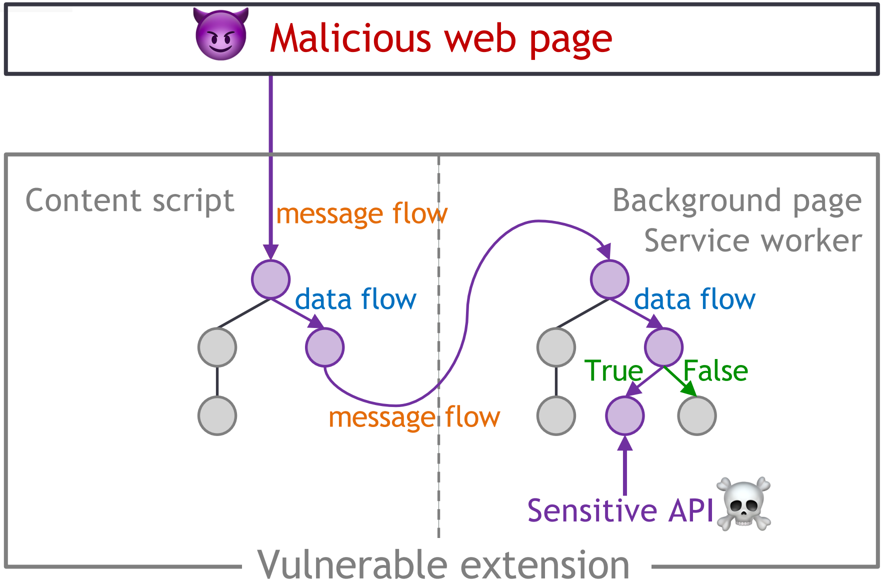

DoubleX: Statically Detecting Vulnerable Data Flows in Browser Extensions at Scale

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

CISPA Helmholtz Center for Information Security

> 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

Detecting Vulnerable Extensions

Fass et al. CCS 2021

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY


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

Aurore Fass, Dolière Francis Somé, Michael Backes, and Ben Stock

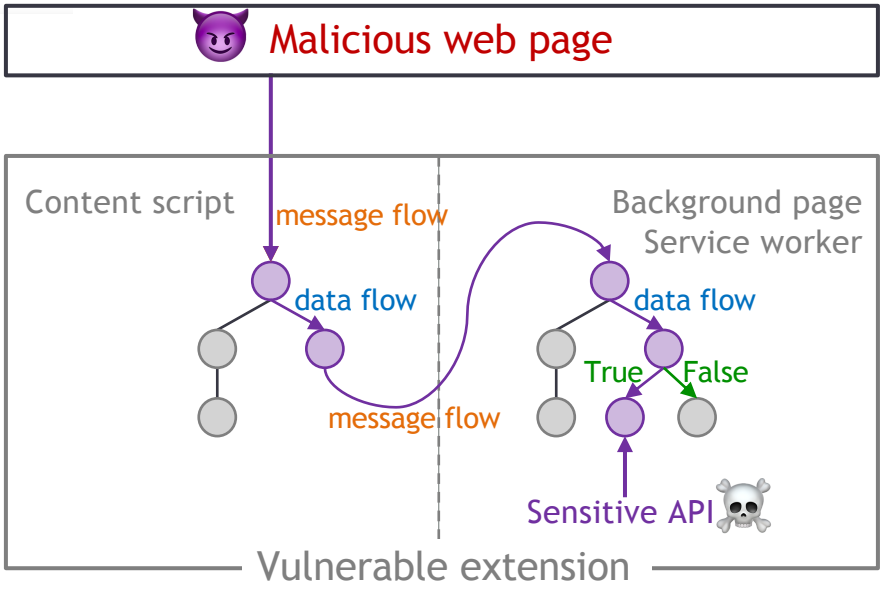
CISPA Helmholtz Center for Information Security

Abstract

Introduction

> 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)

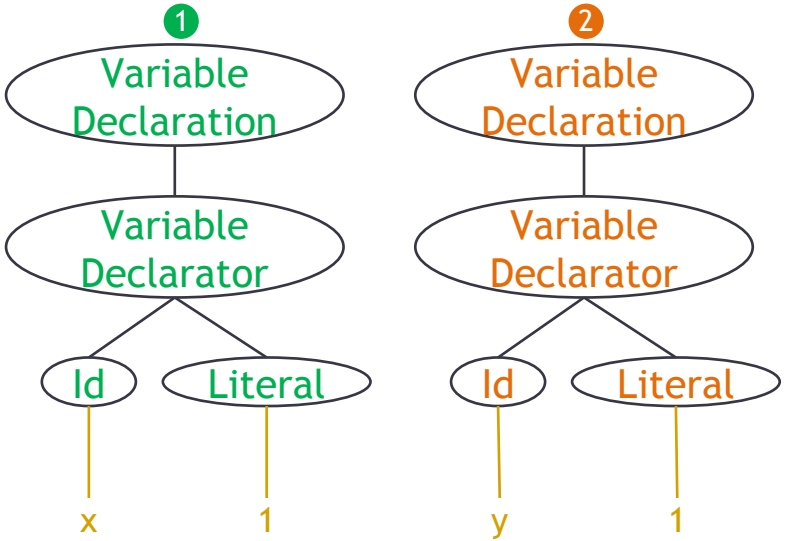
➢ Message interactions

Suspicious data flow tracking

➢ Detects any path between an attacker & sensitive APIs

Abstract Syntax Tree

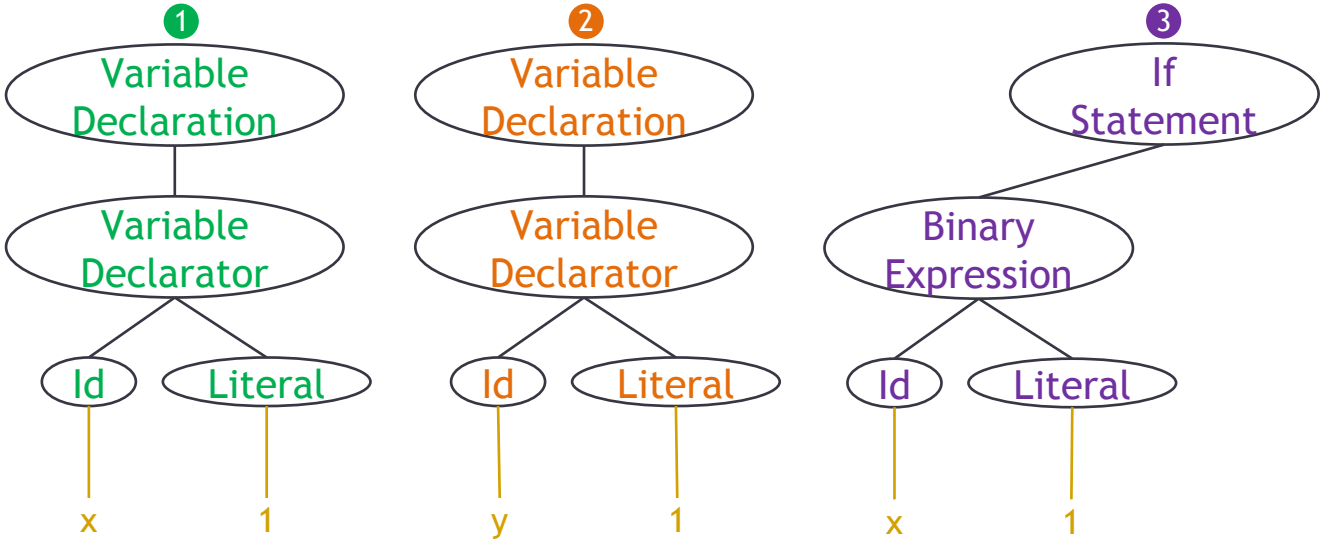
- AST: nesting of programming constructs




```
1 var x = 1;
2 var y = 1;
```

Abstract Syntax Tree

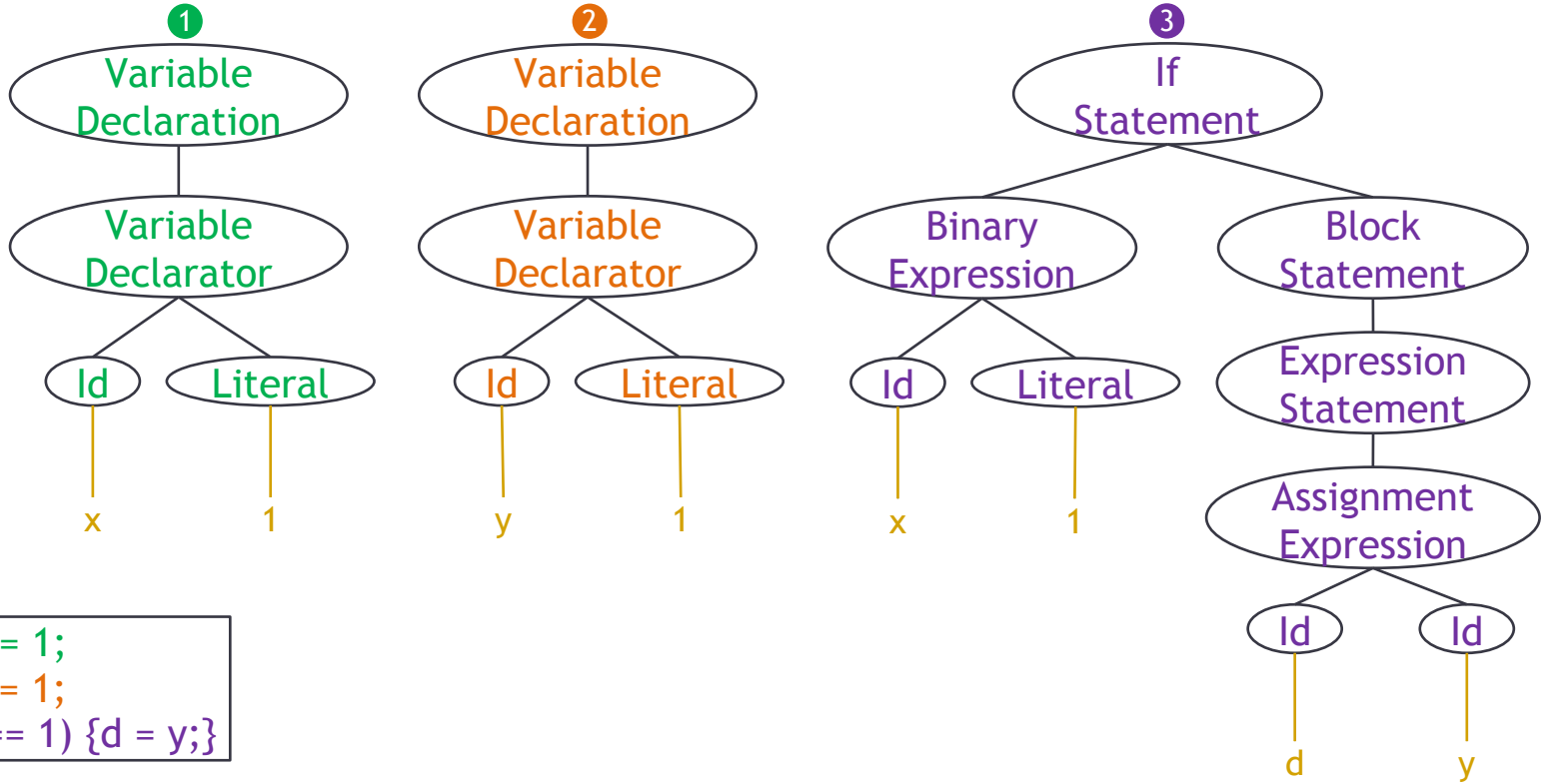
- AST: nesting of programming constructs




```
1 var x = 1;
2 var y = 1;
3 if (x == 1)
```

Abstract Syntax Tree

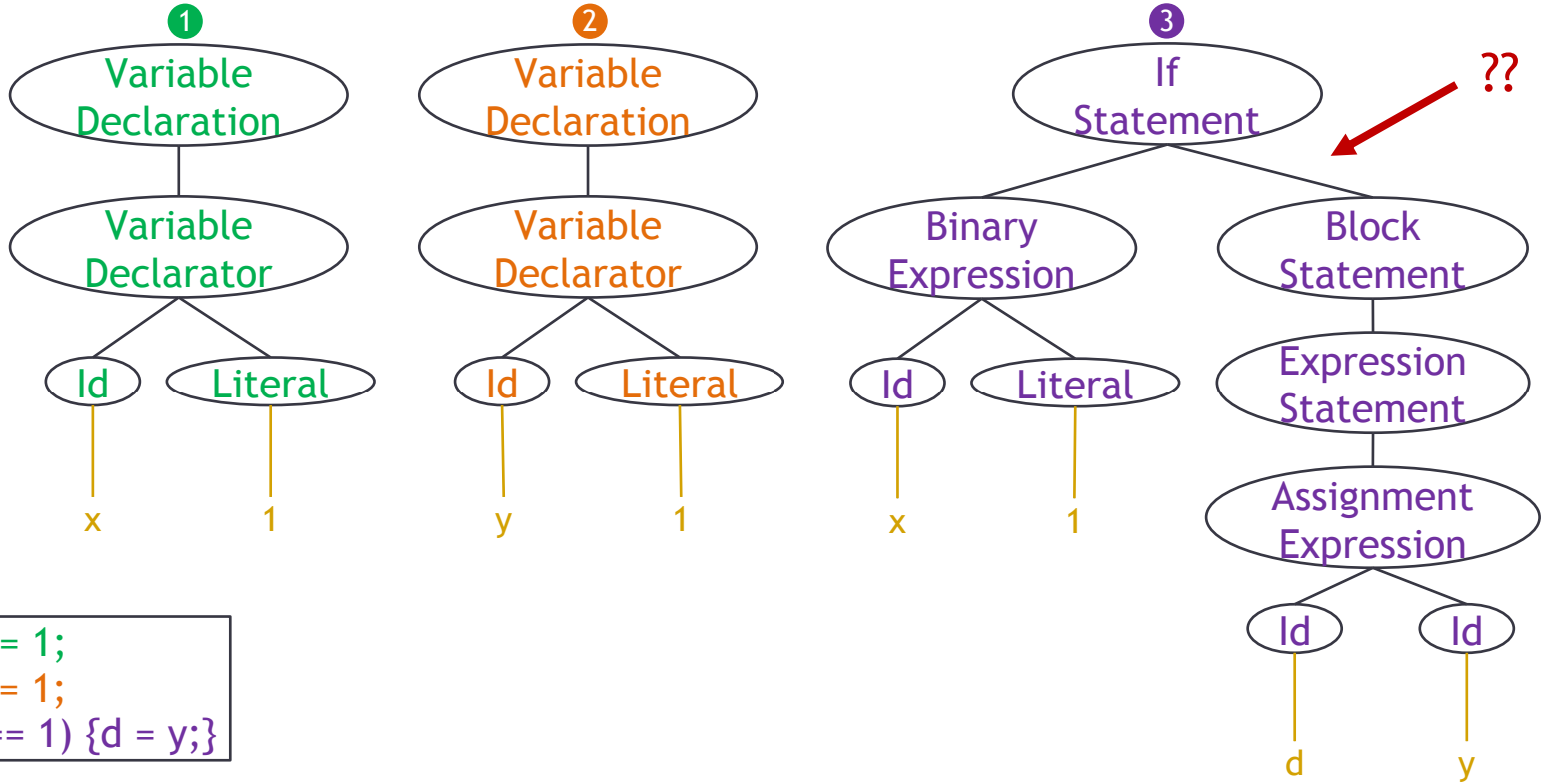
- AST: nesting of programming constructs




```
1 var x = 1;
2 var y = 1;
3 if (x == 1) {d = y;}
```

Abstract Syntax Tree

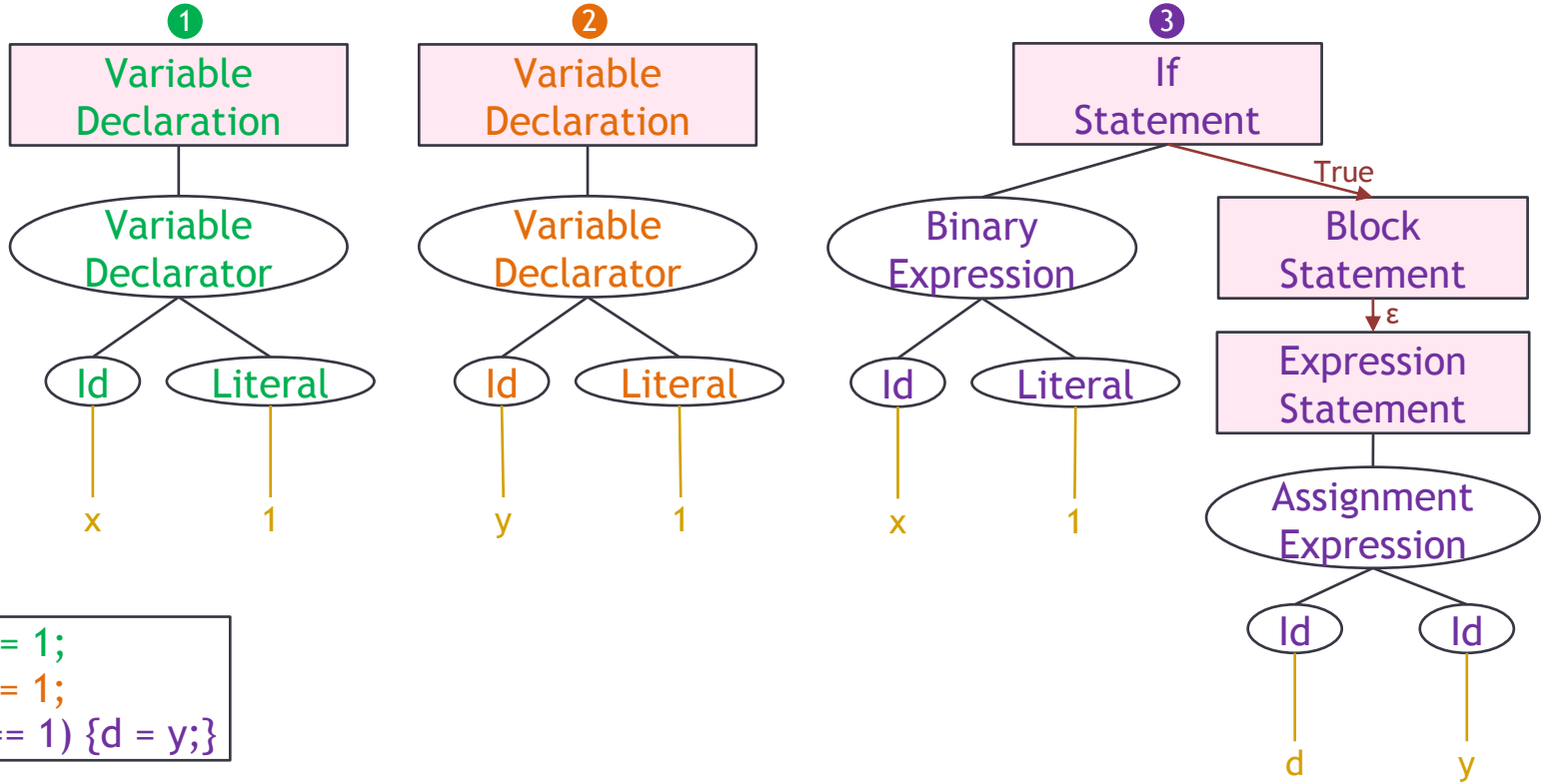
- AST: nesting of programming constructs




```
1 var x = 1;
2 var y = 1;
3 if (x == 1) {d = y;}
```

Control Flow Graph

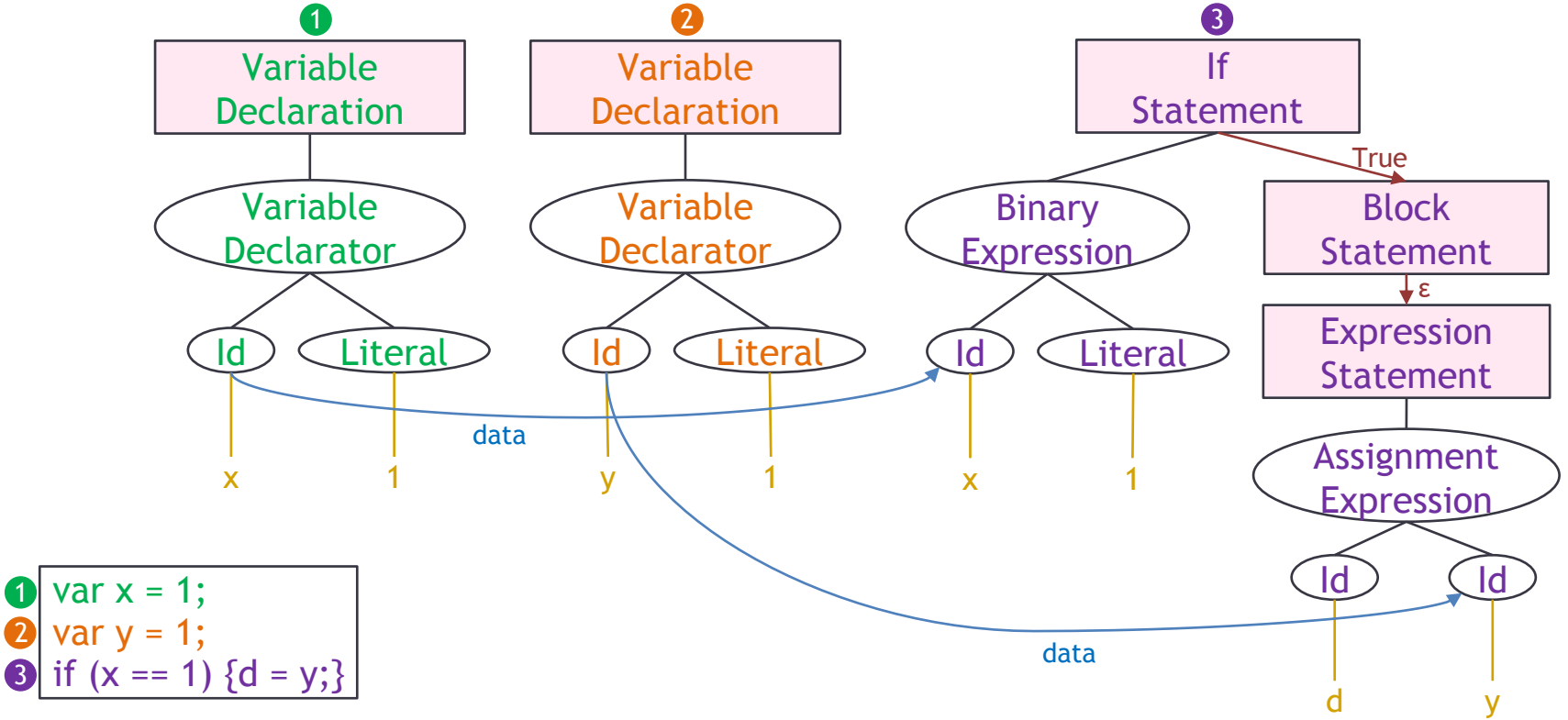
- CFG: execution path conditions




```
1 var x = 1;
2 var y = 1;
3 if (x == 1) {d = y;}
```

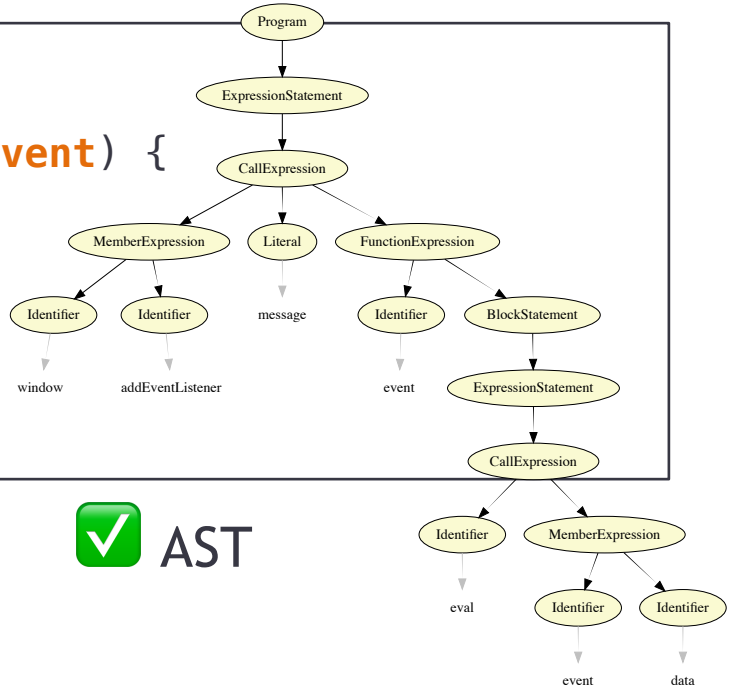
Program Dependence Graph

- PDG: variable dependency



Per-Component JS Code Abstraction

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

    eval(event.data);

})
```



Abstract code representation → ✅ AST

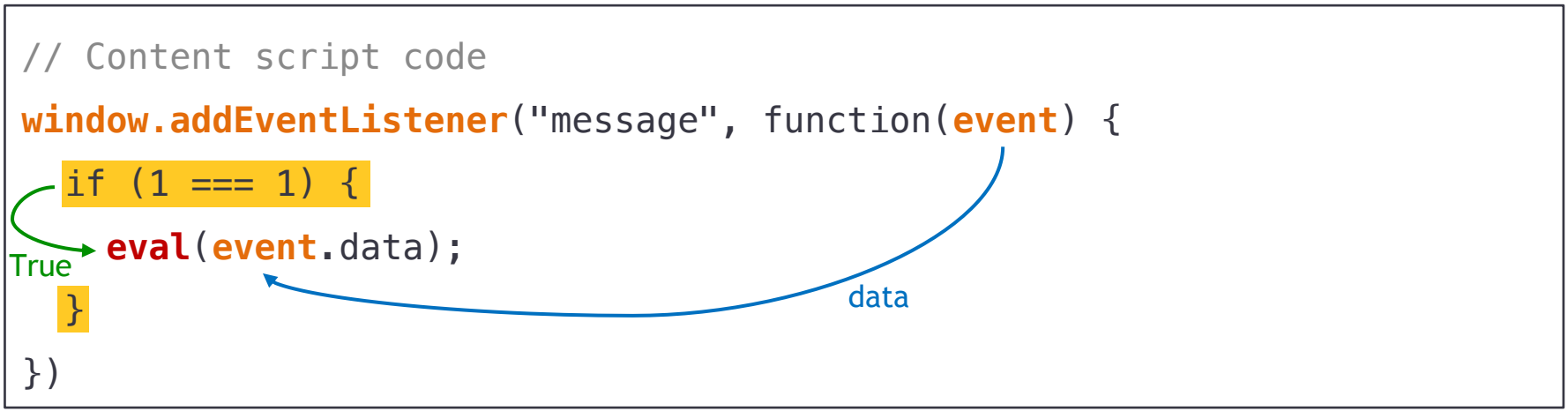
Per-Component JS Code Abstraction

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

    eval(event.data);

})
```

Abstract code representation → ✅ AST

– variable dependencies → ✅ data flow

Per-Component JS Code Abstraction



Abstract code representation → ✅ AST

– conditions → ✅ control flow

– variable dependencies → ✅ data flow

Per-Component JS Code Abstraction

```
// Content script code
window.addEventListener("message", function(event) {
  if (1 === 1) {
    window["e" + "val"](event.data);
  }
})
```

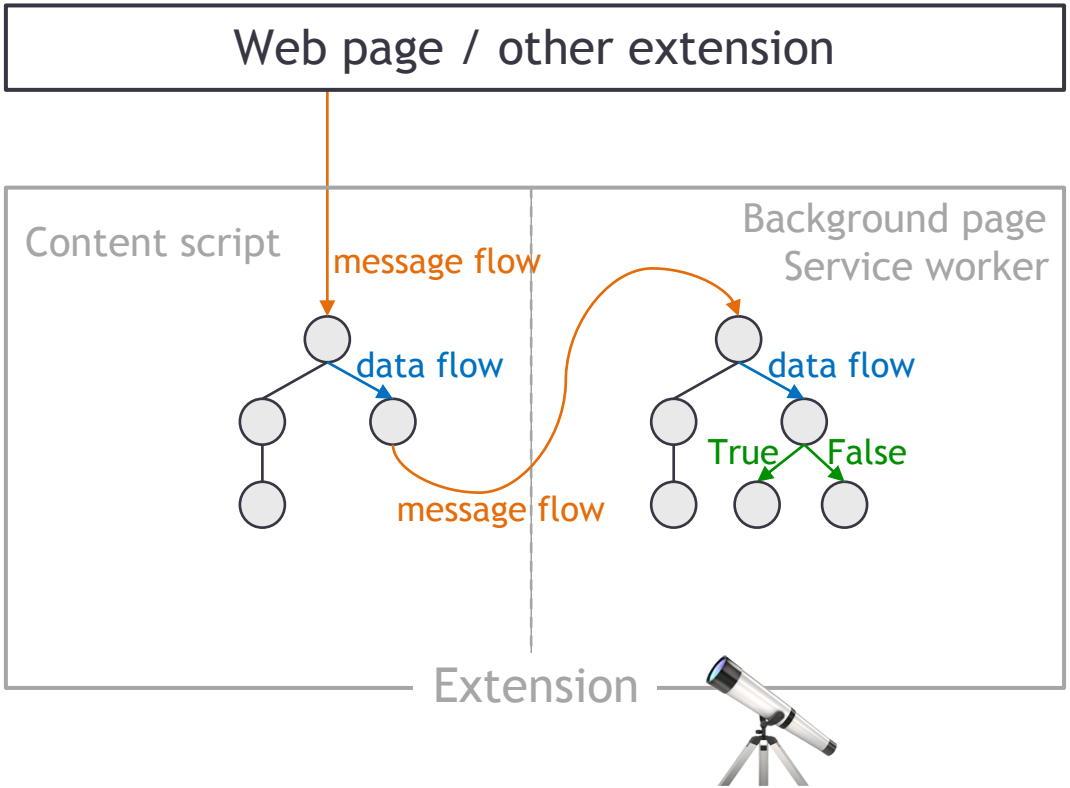
Abstract code representation → ✅ AST

– conditions → ✅ control flow

– variable dependencies → ✅ data flow

– variable values → ✅ pointer analysis

Detecting Vulnerable Extensions



→ DOUBLEX: detects suspicious data flows from and toward an extension privileged context

Per-component JS code abstraction

- AST
- Control flow
- Data flow
- Pointer analysis

Extension Dependence Graph (EDG)

➢ Message interactions

Suspicious data flow tracking

➢ Detects any path between an attacker & sensitive APIs

↓

Data flow report

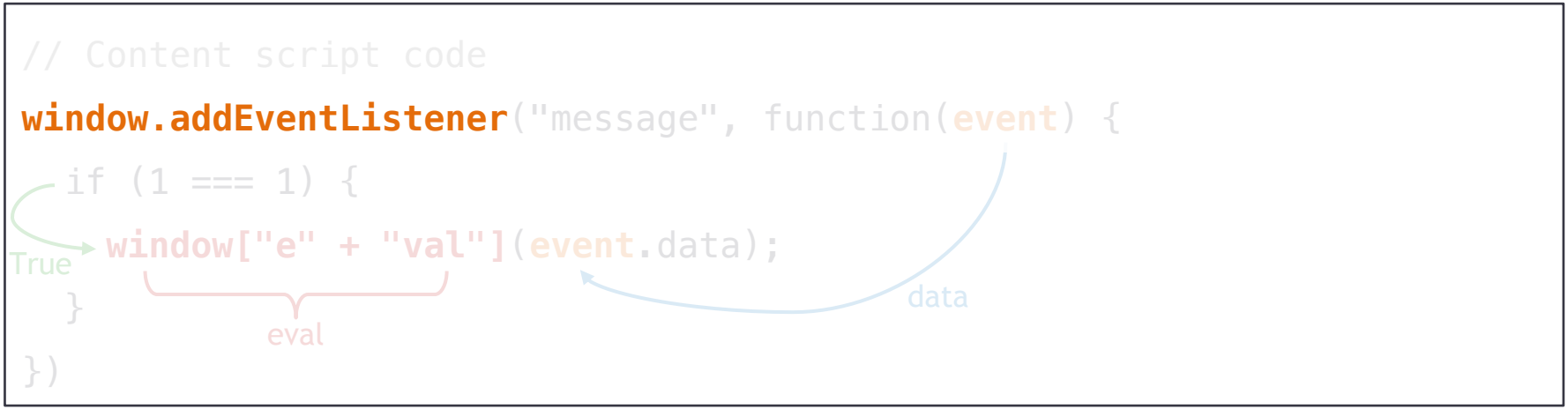
Extension Dependence Graph

```
// Content script code
window.addEventListener("message", function(event) {
  if (1 === 1) {
    window["e" + "val"](event.data);
  }
})
```

– external messages

– internal messages

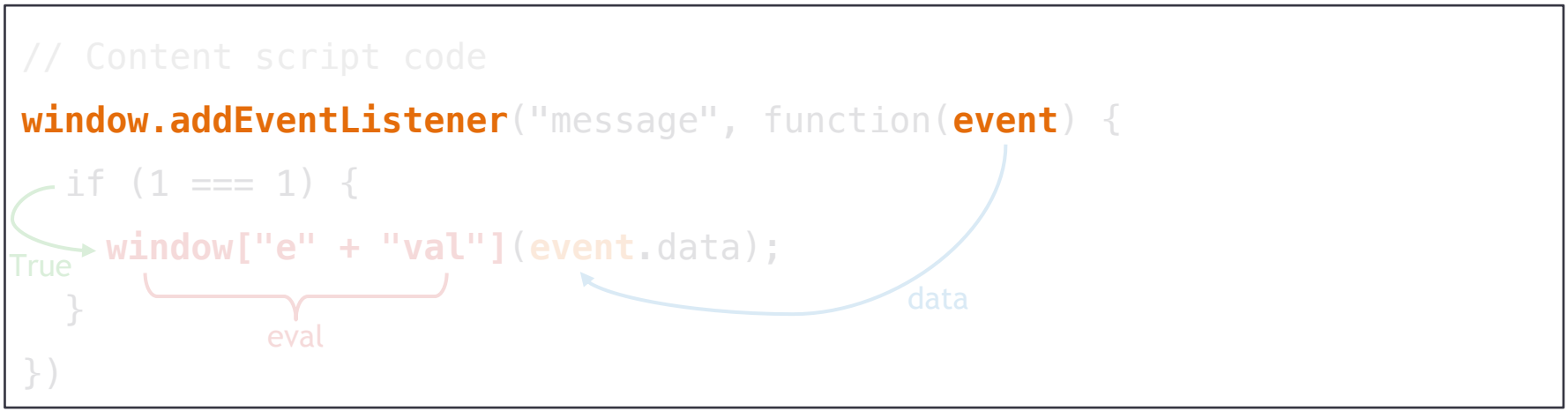
Extension Dependence Graph



– external messages

– internal messages

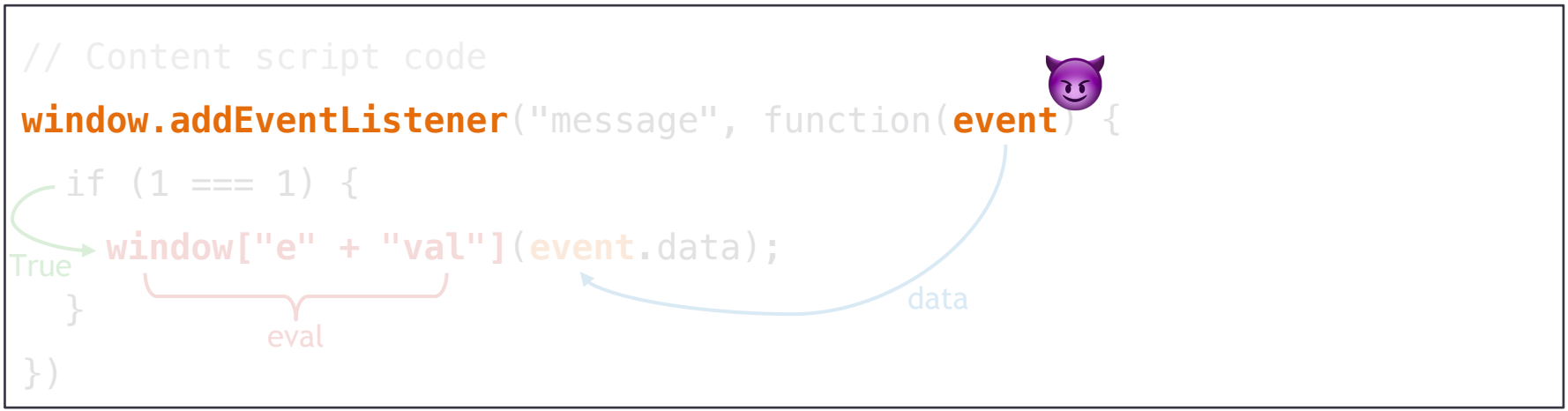
Extension Dependence Graph



– external messages ✅

– internal messages

Extension Dependence Graph



- external messages ✅
- internal messages

Extension Dependence Graph

```
// Content script code
chrome.runtime.sendMessage({toBP: mess});
```

- external messages ✅
- internal messages

Extension Dependence Graph

```
// Content script code
chrome.runtime.sendMessage({toBP: mess});
```

```
// Background page code
chrome.runtime.onMessage.addListener(function(request) {

})
```

- external messages ✅
- internal messages

Extension Dependence Graph

```
// Content script code
chrome.runtime.sendMessage({toBP: mess});
```

```
// Background page code
chrome.runtime.onMessage.addListener(function(request) {

})
```

- external messages ✅
- internal messages

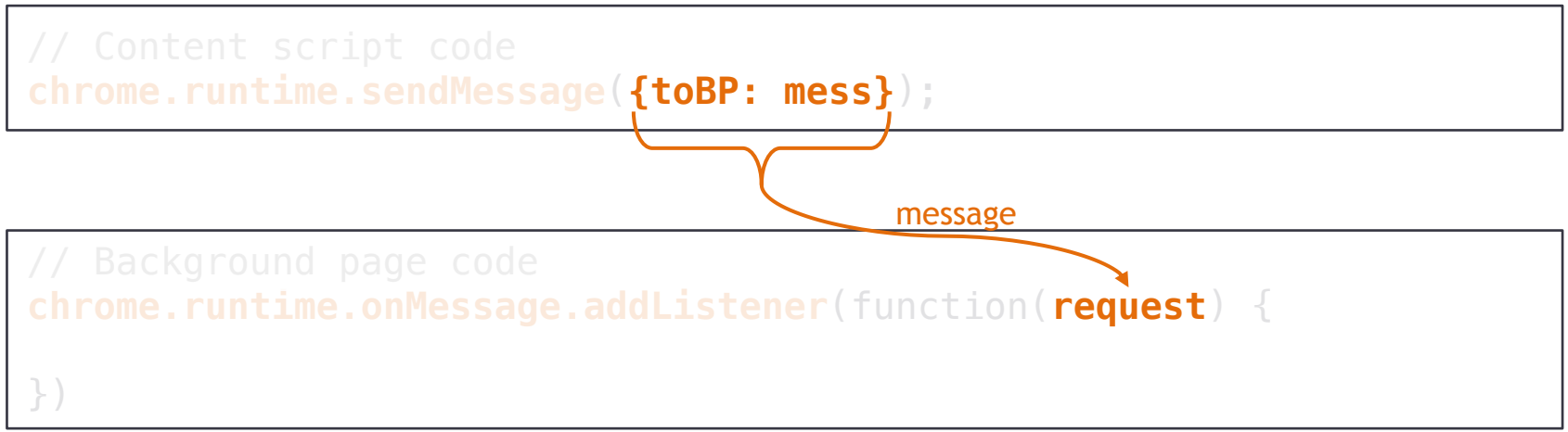
Extension Dependence Graph

```
// Content script code
chrome.runtime.sendMessage({toBP: mess});
```

```
// Background page code
chrome.runtime.onMessage.addListener(function(request) {

})
```

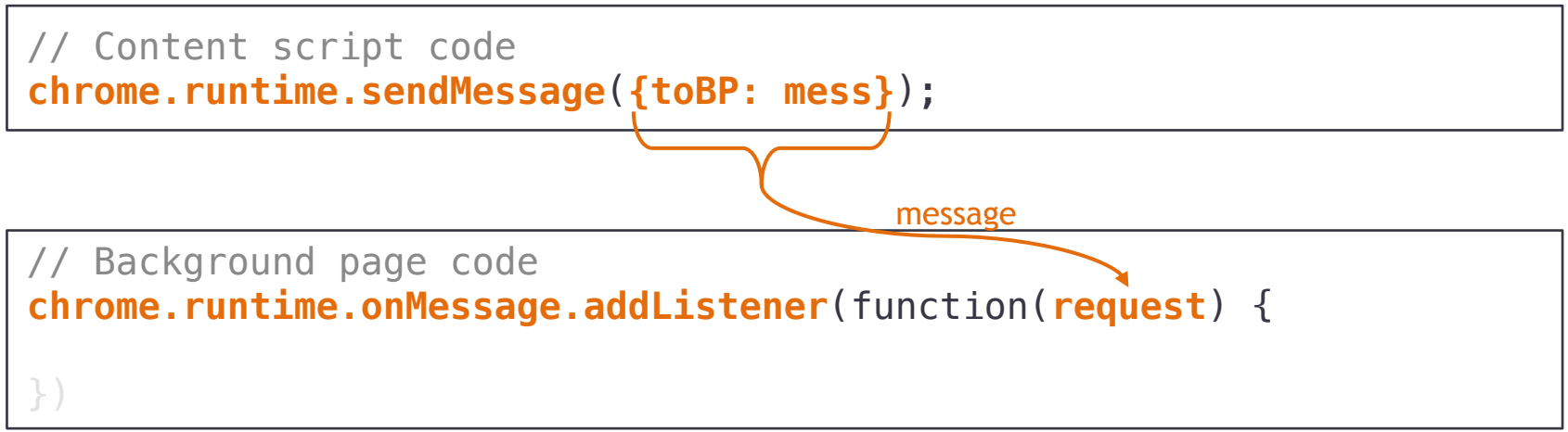
- external messages ✅
- internal messages

Extension Dependence Graph



- external messages ✅
- internal messages ✅

Extension Dependence Graph

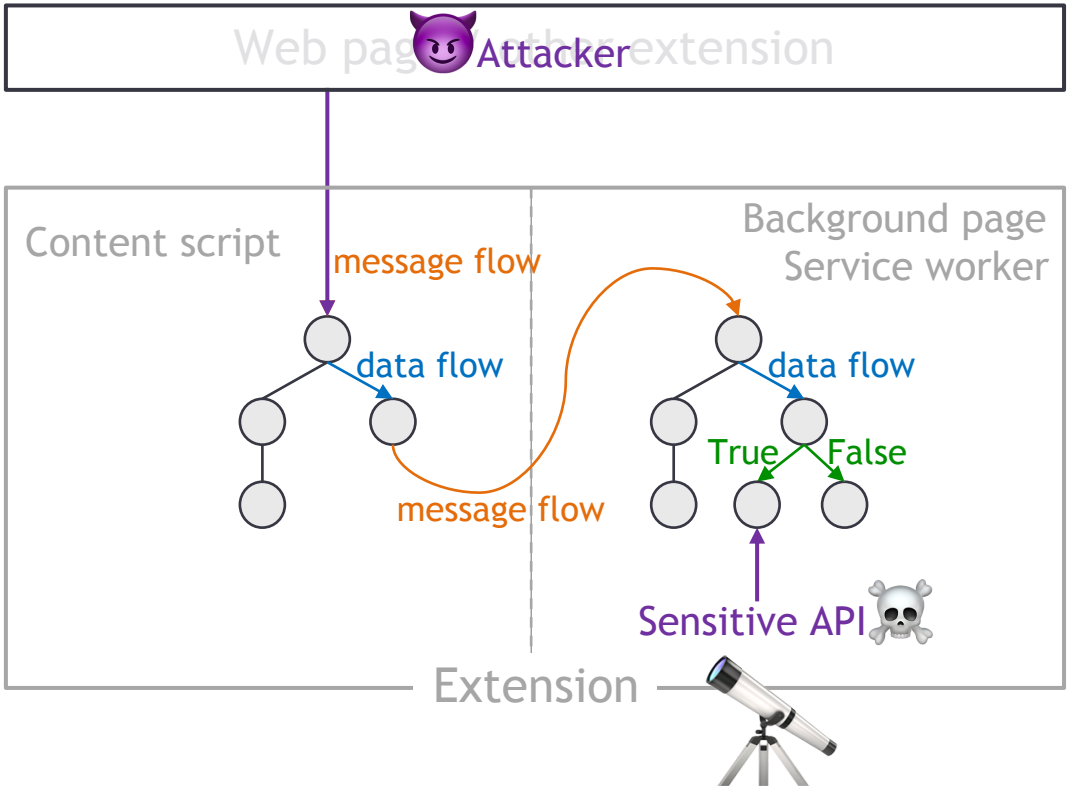


– external messages ✅

– internal messages ✅

➢ Models message interaction within and outside of an extension

Detecting Vulnerable Extensions



Per-component JS code abstraction

- AST
- Control flow
- Data flow
- Pointer analysis

Extension Dependence Graph (EDG)

➢ Message interactions

Suspicious data flow tracking

➢ Detects any path between an attacker & sensitive APIs

↓

Data flow report

→ DOUBLEX: detects suspicious data flows from and toward an extension privileged context

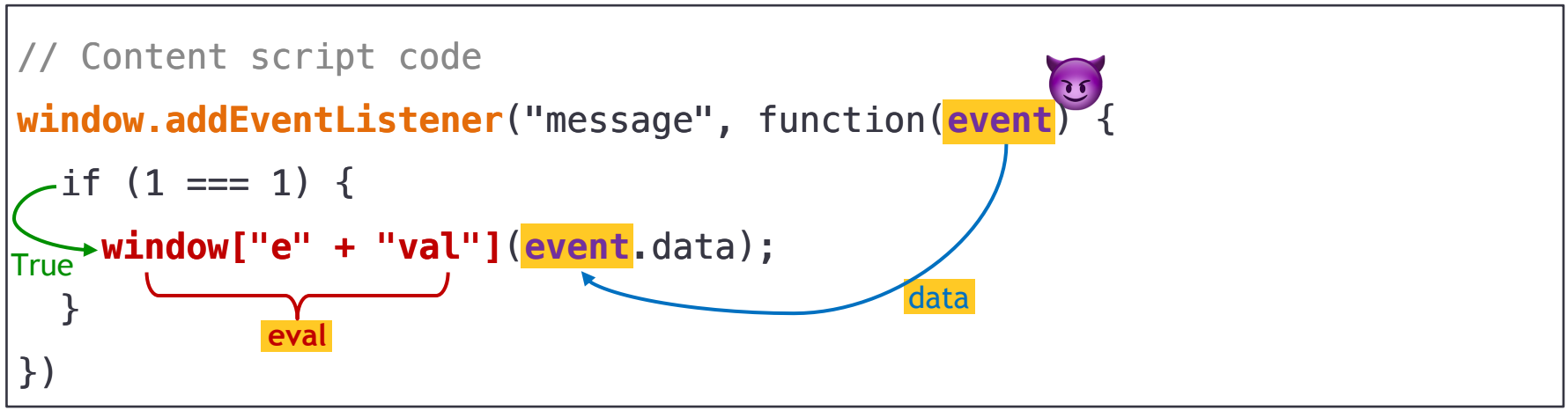
Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5    }
6  })
```

Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5    }
6  })
```

Suspicious Data Flow Tracking



Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5    }
6  })
```

True

eval

data

Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5    }
6  })
```

True

eval

data


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

Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5    }
6  })
```

True

eval

data


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

Suspicious Data Flow Tracking

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    if (1 === 1) {
4      window["e" + "val"](event.data);
5
6
7
8
9    }
10 })
```

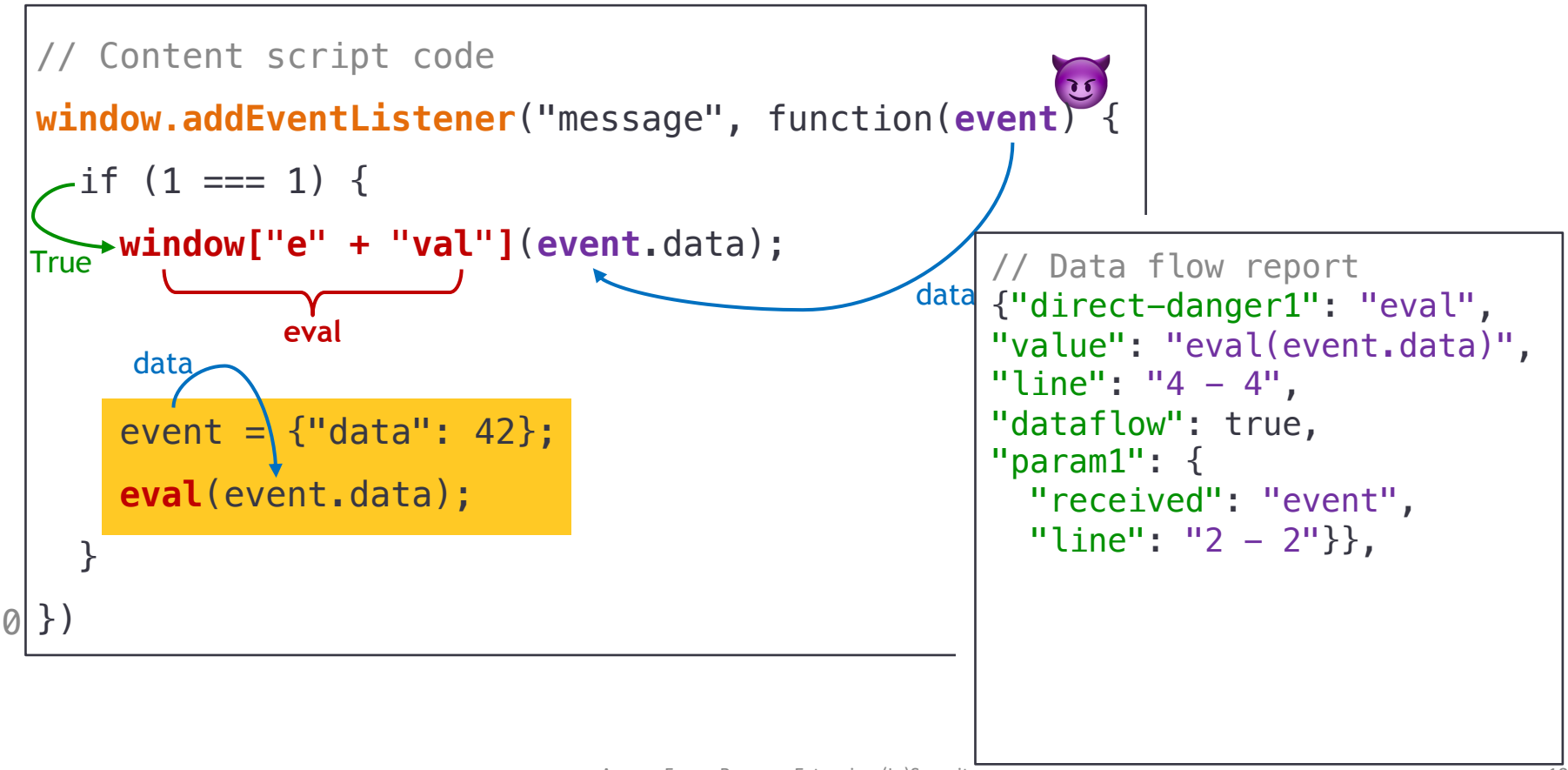
True

eval

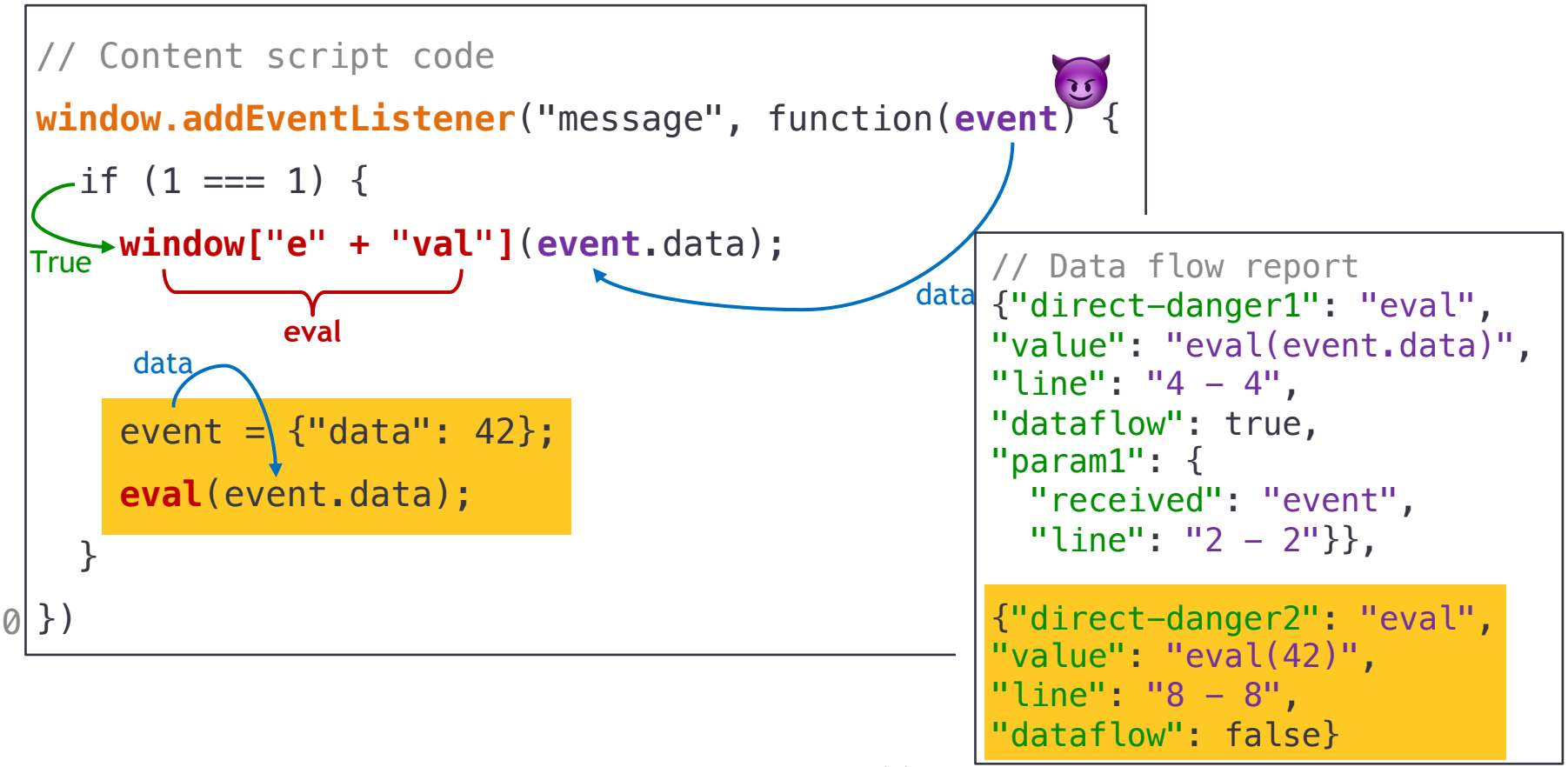
data

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

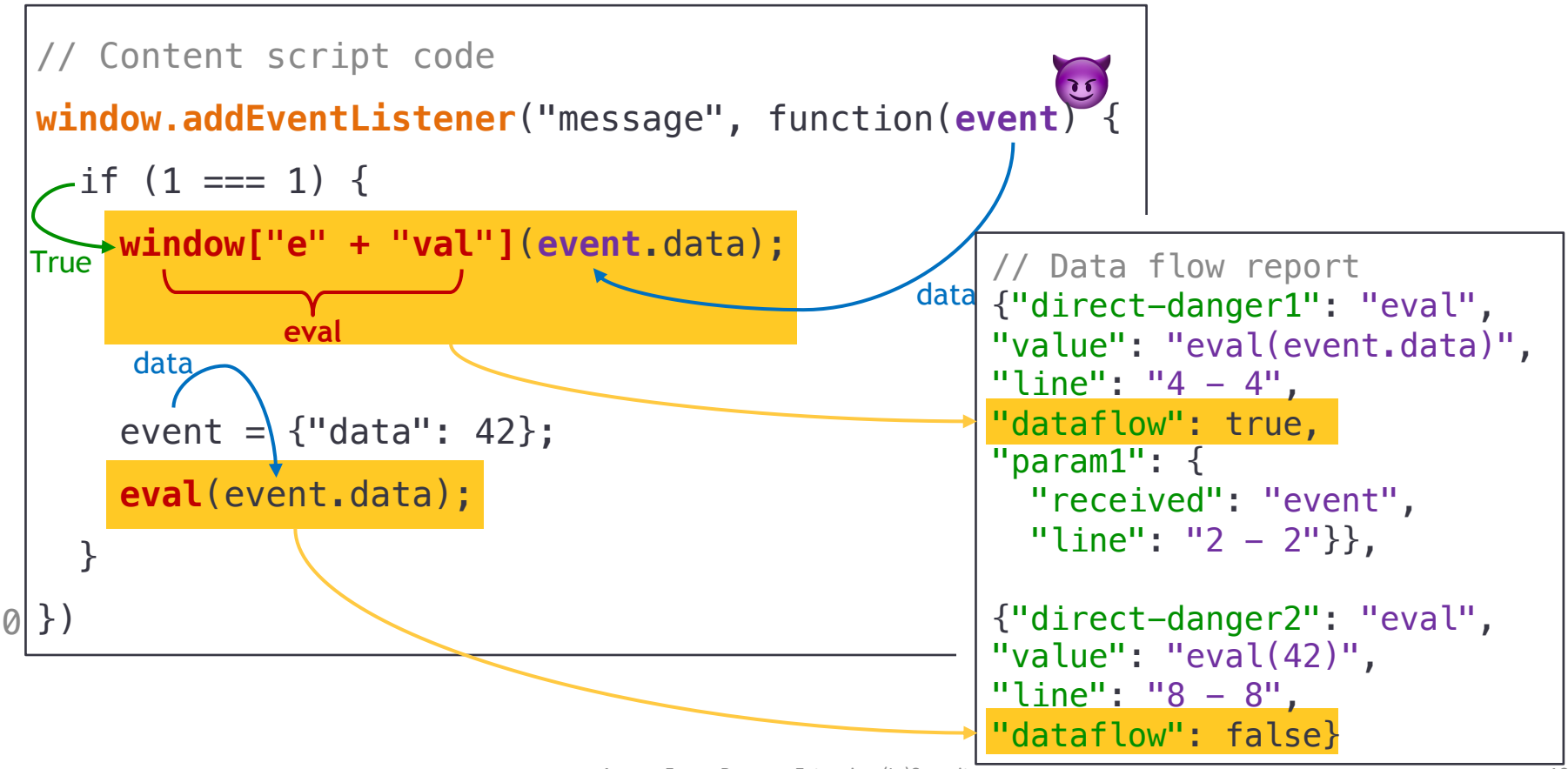
Suspicious Data Flow Tracking



Suspicious Data Flow Tracking



Suspicious Data Flow Tracking



Suspicious Data Flow Tracking

```
1  // Content script code
2
3
4
5
6
7  // Background page code
8
9
10
```

Suspicious Data Flow Tracking

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

```
7  // Background page code
8
9
10
```

Suspicious Data Flow Tracking

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

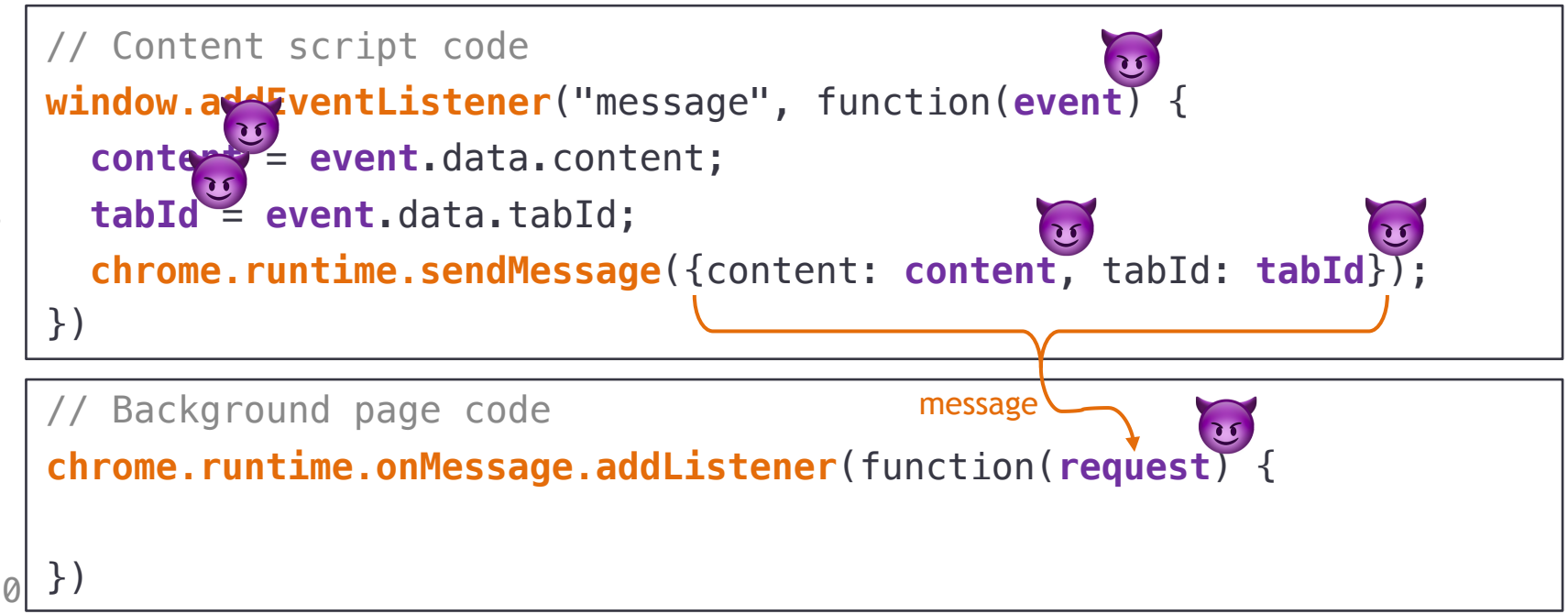
```
7  // Background page code
8
9
10
```

Suspicious Data Flow Tracking

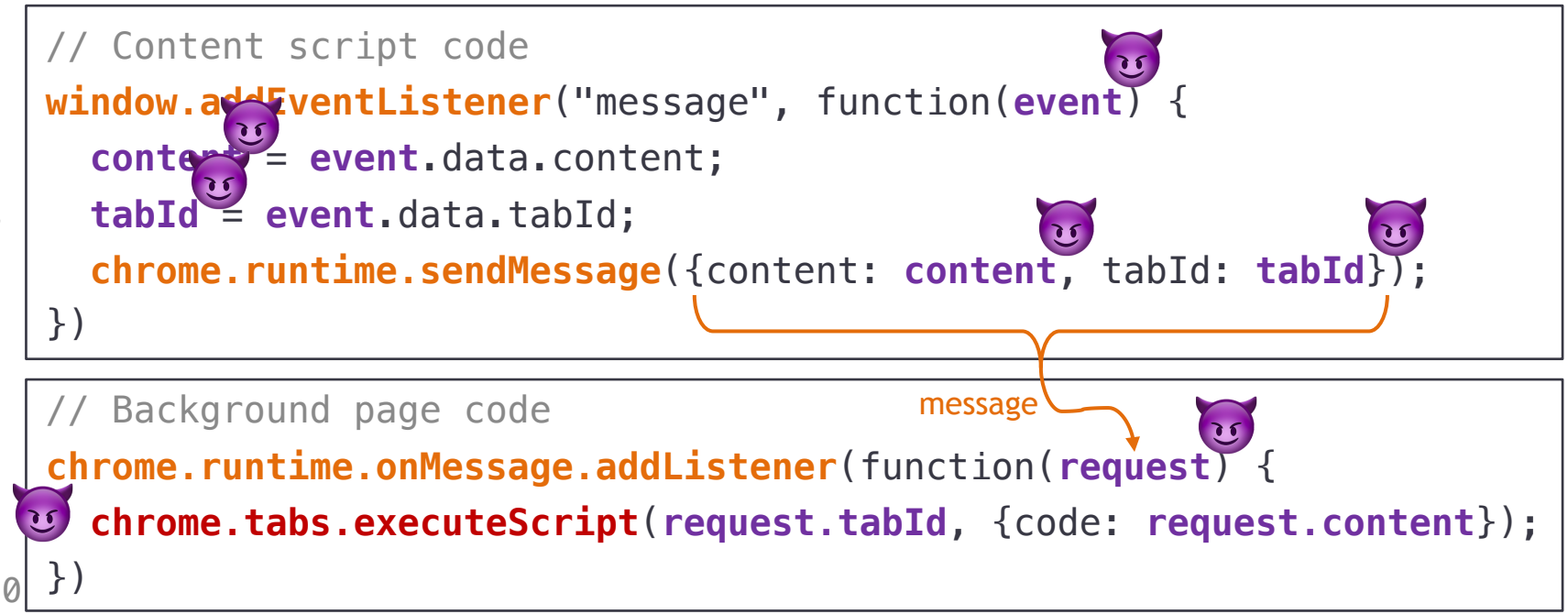
```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    content = event.data.content;
4    tabId = event.data.tabId;
5    chrome.runtime.sendMessage({content: content, tabId: tabId});
6  })
```

```
7  // Background page code
8
9
10
```

Suspicious Data Flow Tracking



Suspicious Data Flow Tracking



Suspicious Data Flow Tracking

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    content = event.data.content;
4    tabId = event.data.tabId;
5    chrome.runtime.sendMessage({content: content, tabId: tabId});
6  })
7  // Background page code
8  chrome.runtime.onMessage.addListener(function(request) {
9    chrome.tabs.executeScript(request.tabId, {code: request.content});
10 })
```

message

```
// Data flow report
{"direct-danger1": "tabs.executeScript",
"line": "9 - 9",
"dataflow": true,
...
"param1": {
  "received": "event",
  "line": "2 - 2"}}
```

Suspicious Data Flow Tracking

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

```
1  // Content script code
2  window.addEventListener("message", function(event) {
3    content = event.data.content;
4    tabId = event.data.tabId;
5    chrome.runtime.sendMessage({content: content, tabId: tabId});
6  })
```

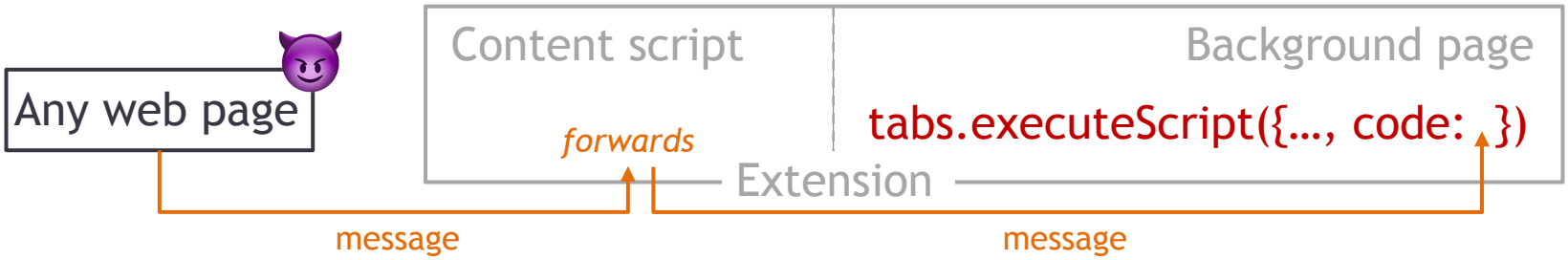
message

```
7  // Background page code
8  chrome.runtime.onMessage.addListener(function(request) {
9    chrome.tabs.executeScript(request.tabId, {code: request.content});
10 })
```

```
// Data flow report
{"direct-danger1": "tabs.executeScript",
"line": "9 - 9",
"dataflow": true,
...
"param1": {
  "received": "event",
  "line": "2 - 2"}}
```

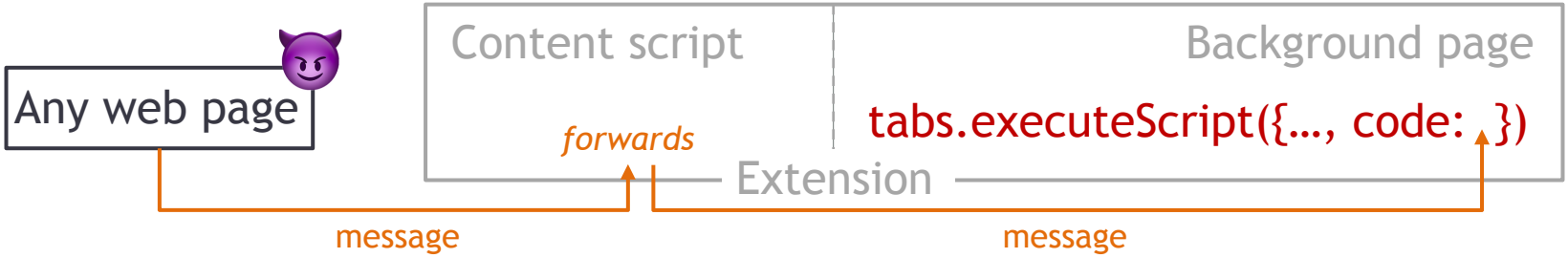
Case Studies of Vulnerable Chrome Extensions

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

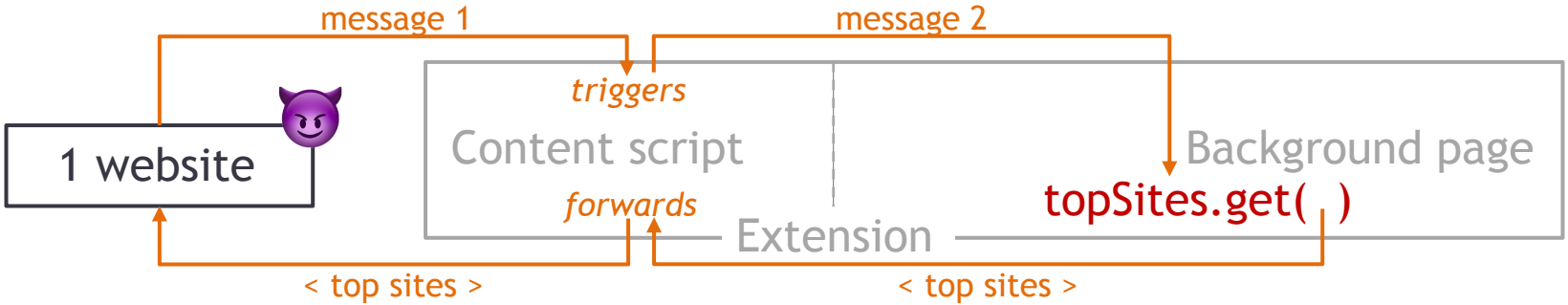


Case Studies of Vulnerable Chrome Extensions

- 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!)

[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

Takeaways — Browser Extension (In)Security

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Browser Extensions are Popular

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 JS, HTML, or CSS files, defined in a `manifest.json`
- **145k** Chrome extensions totaling over **1.6B** active users

Security-Noteworthy Extensions (SNE)

- **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

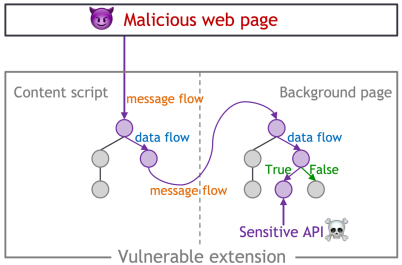
What is in the Chrome Web Store?

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%**

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