

Browser Extension (In)Security

Aurore Fass

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

Dr.-Ing. Aurore (/ɔʁɔʁ/) FASS

🇫🇷 Graduated from TELECOM Nancy (FR, 2017)



🇩🇪 PhD Student + Postdoc at CISPA (DE, 2017–21)



🇺🇸 Visiting Assistant Professor at Stanford (US, 2021–23)



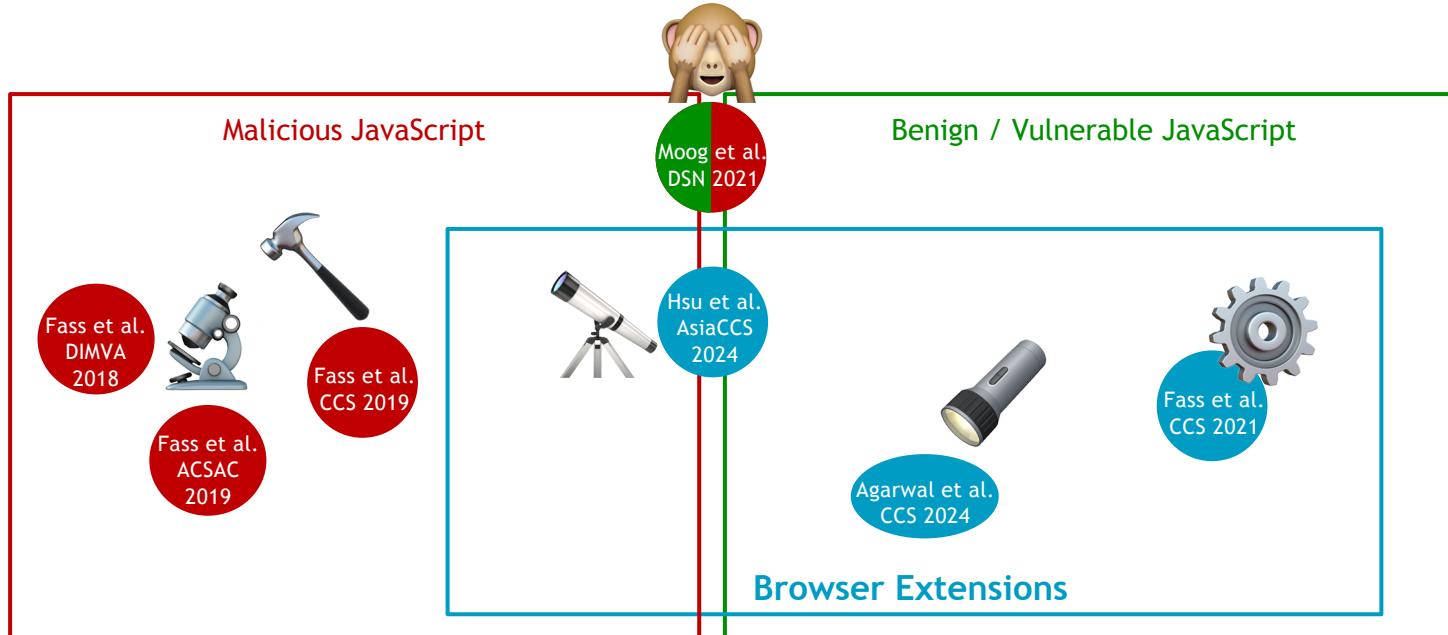
🇩🇪 Tenure-Track Faculty at CISPA (DE, 2023–)



CISPA - Helmholtz Center for Information Security



Research Work: Web Security & Privacy



Apruzzese et al.
AISeC 2024

Machine Learning



Ruth et al.
IMC 2022

Internet Measurements

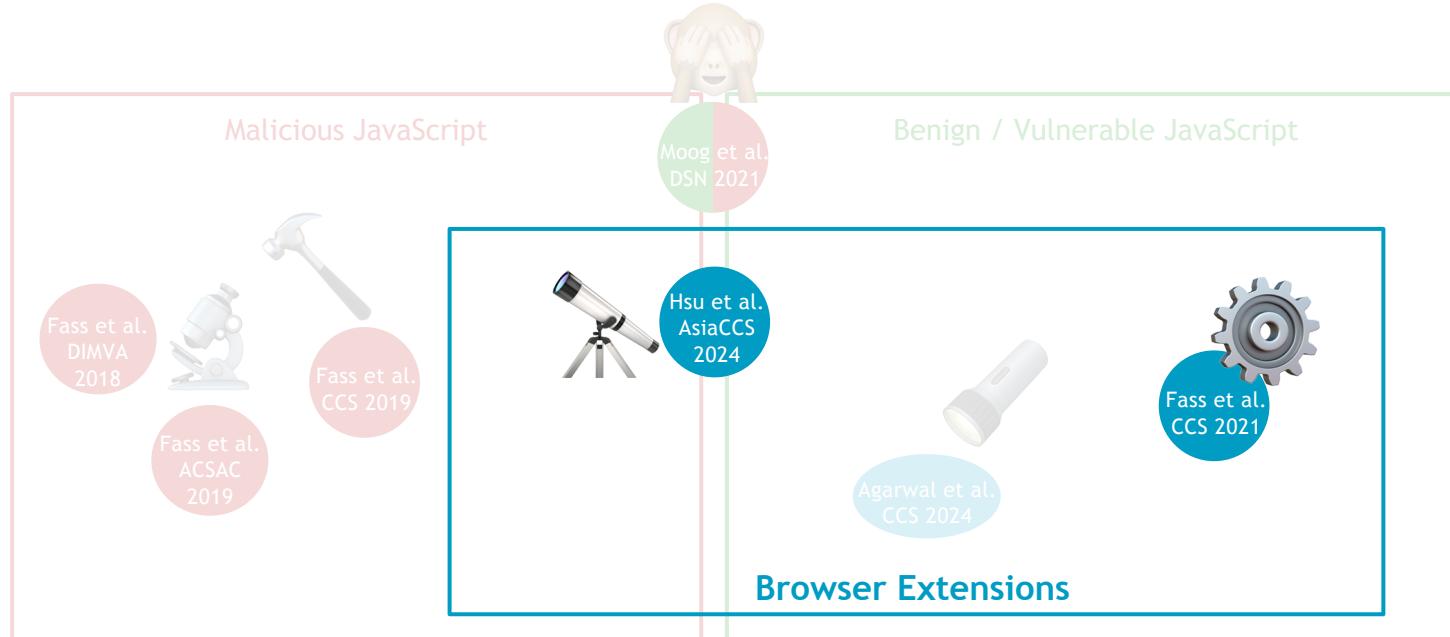
Izhikevich et al.
IMC 2023



Software Engineering

Troppmann et al.
ASE 2024

Research Work: Web Security & Privacy



Apruzzese et al.
AISeC 2024

Machine Learning

Troppmann et al.
ASE 2024

Software Engineering

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

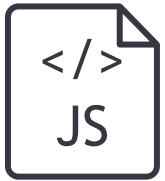
■ Investigating Security-Noteworthy Extensions (SNE)

- SNE definition
- Extension analysis & SNE detection

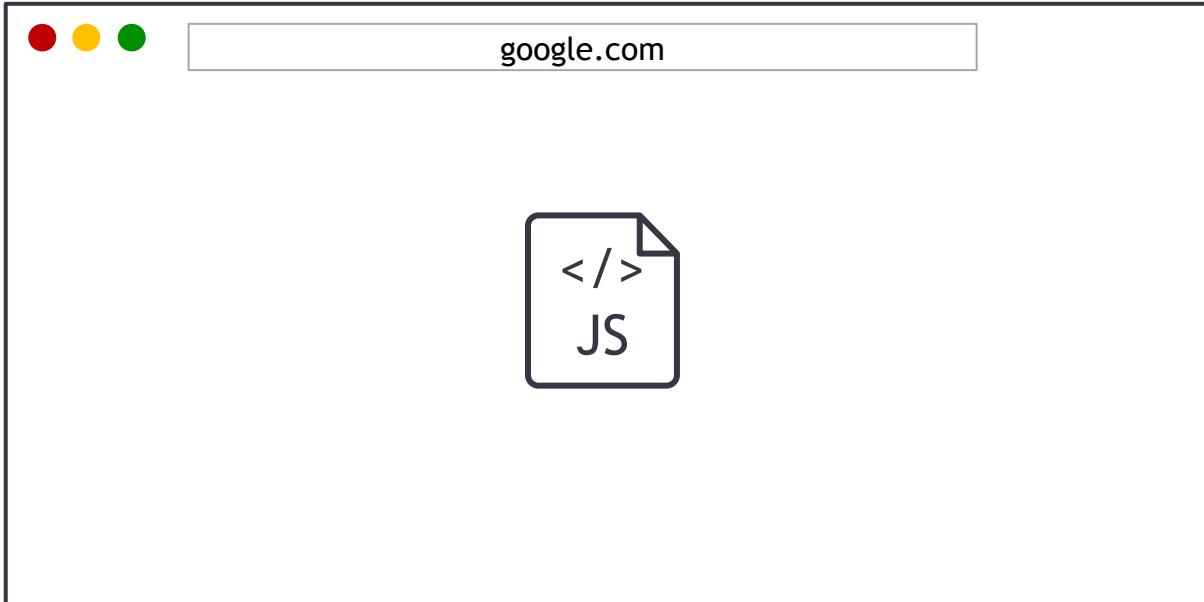
■ Detecting Vulnerable Extensions

- Threat models & examples
- Static analysis of extensions (JavaScript) & examples
- Case studies, results, and potential defense strategies

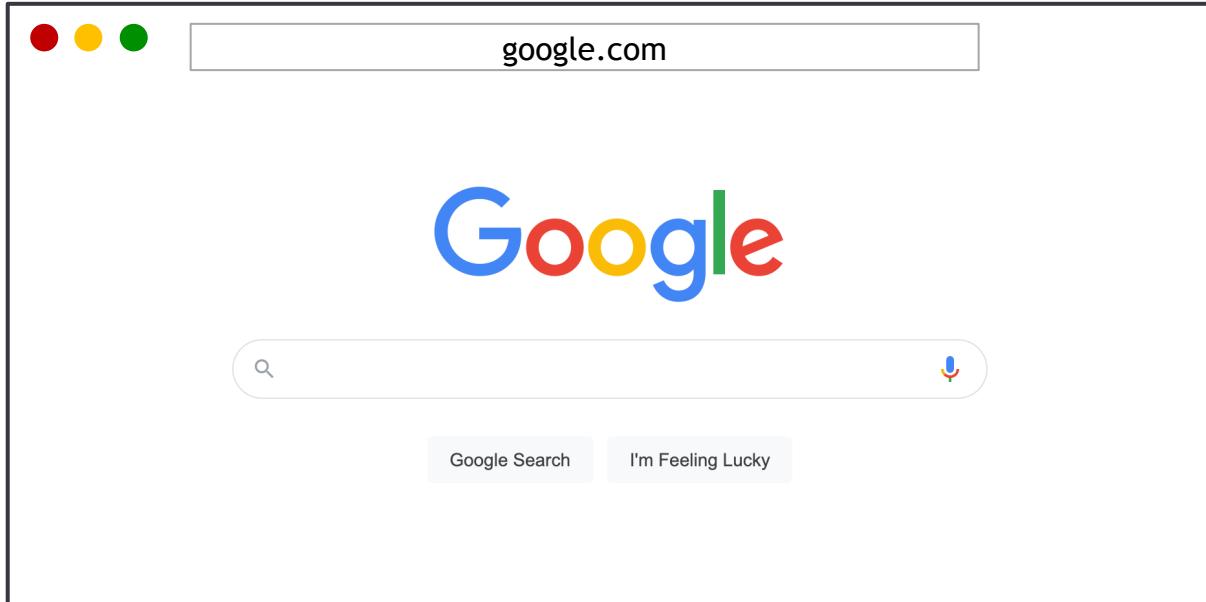
Background – JavaScript



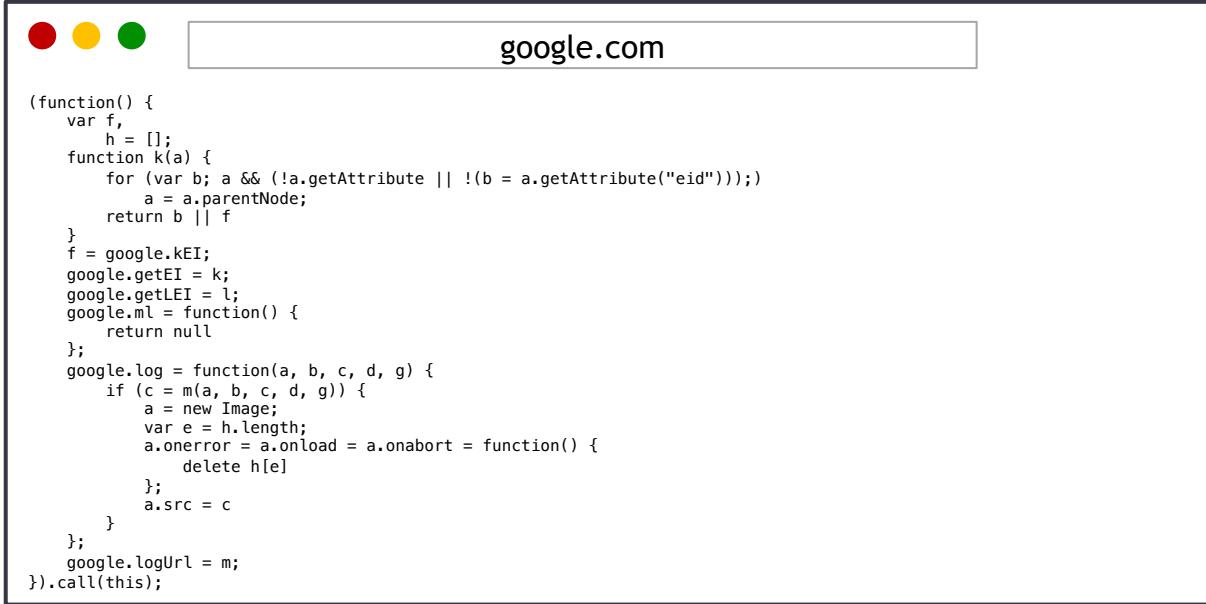
Background – JavaScript



Background – JavaScript



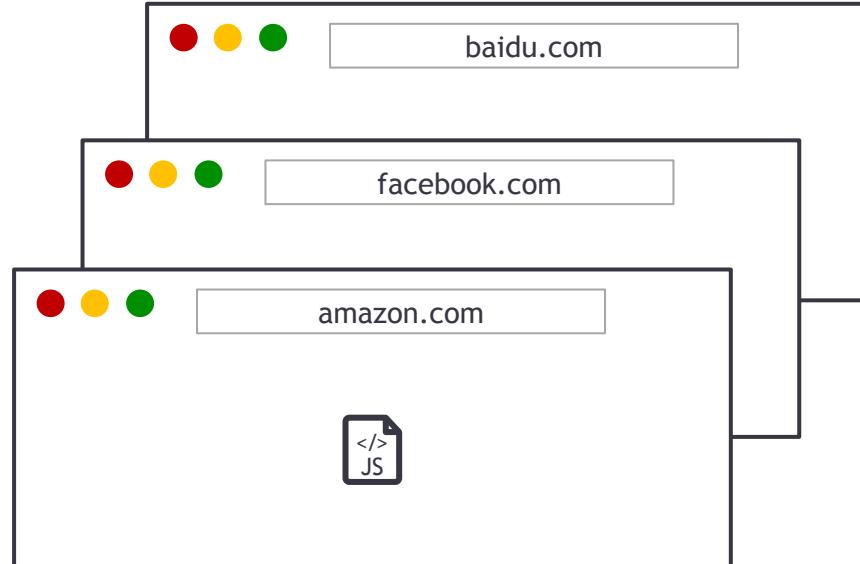
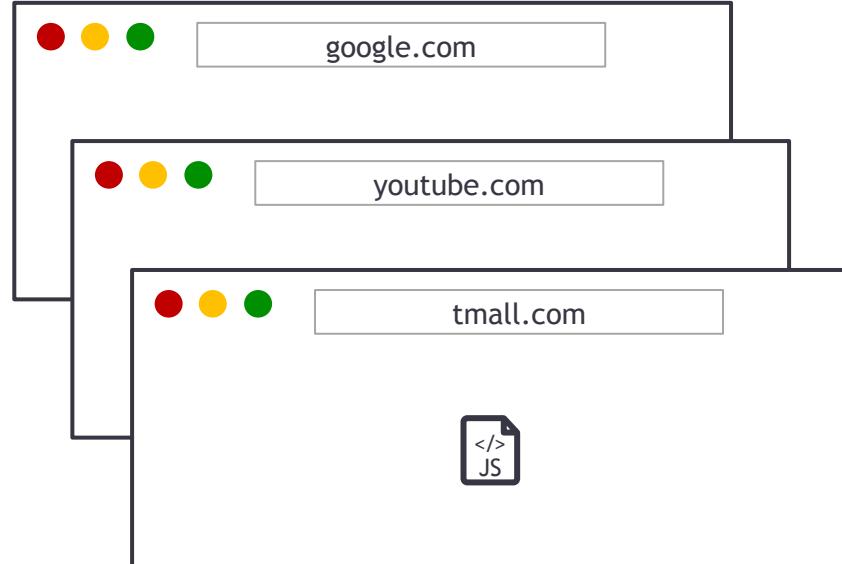
Background – JavaScript



The screenshot shows a browser window with three colored window controls (red, yellow, green) at the top left. The address bar contains the URL "google.com". The main content area of the browser displays a large amount of JavaScript code. The code appears to be a malicious script, likely a browser extension or exploit, which manipulates the Google homepage to perform unwanted actions.

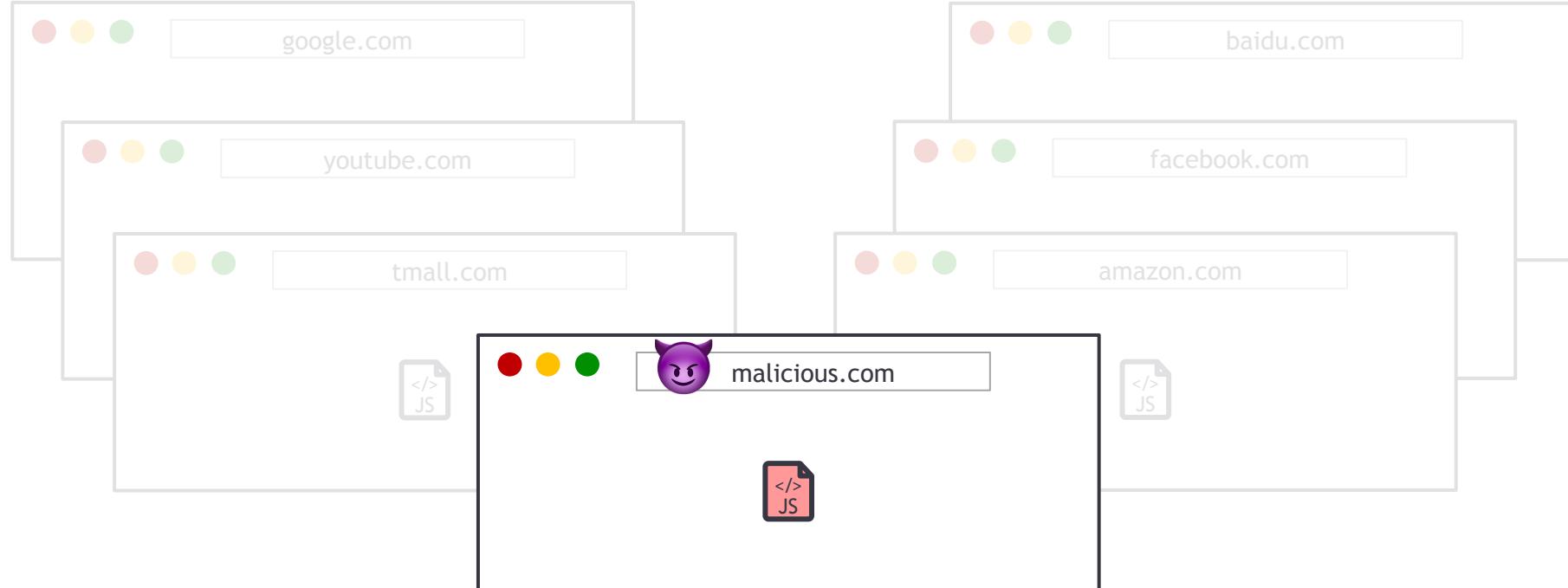
```
(function() {
    var f,
        h = [];
    function k(a) {
        for (var b; a && (!a.getAttribute || !(b = a.getAttribute("eid")));)
            a = a.parentNode;
        return b || f
    }
    f = google.kEI;
    google.getEI = k;
    google.getLEI = l;
    google.ml = function() {
        return null
    };
    google.log = function(a, b, c, d, g) {
        if (c = m(a, b, c, d, g)) {
            a = new Image;
            var e = h.length;
            a.onerror = a.onload = a.onabort = function() {
                delete h[e]
            };
            a.src = c
        }
    };
    google.logUrl = m;
}).call(this);
```

Background – JavaScript



JavaScript usage: > 99% websites

Background – JavaScript



JavaScript usage: > 99% websites

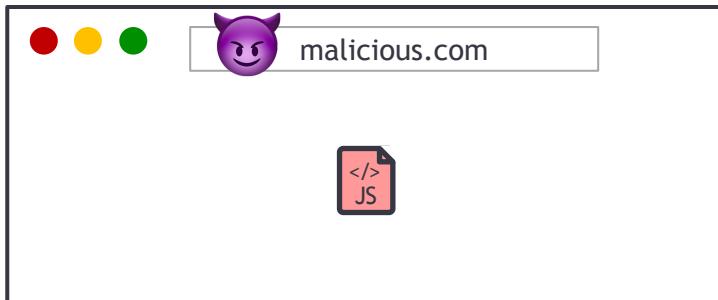
Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims
 - e.g., exploiting vulnerabilities,
stealing sensitive user data

Background – JavaScript: Basis for Attacks

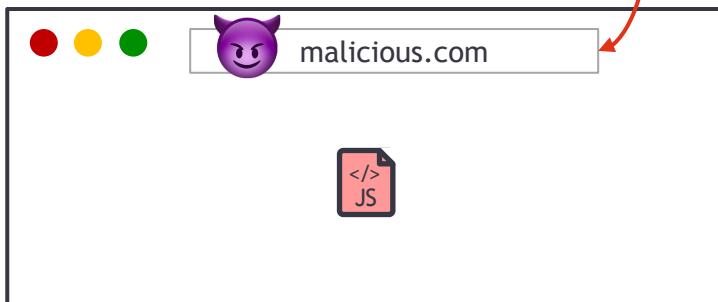
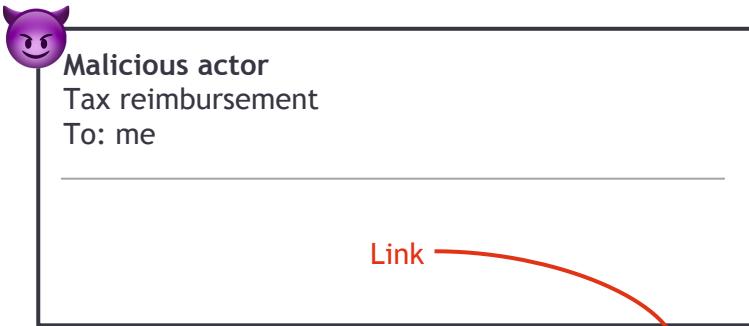
Malicious JavaScript

- Designed by malicious actors
- Aim: harming victims
 - e.g., exploiting vulnerabilities, stealing sensitive user data



Background – JavaScript: Basis for Attacks

Malicious JavaScript



Background – JavaScript: Basis for Attacks

Malicious JavaScript

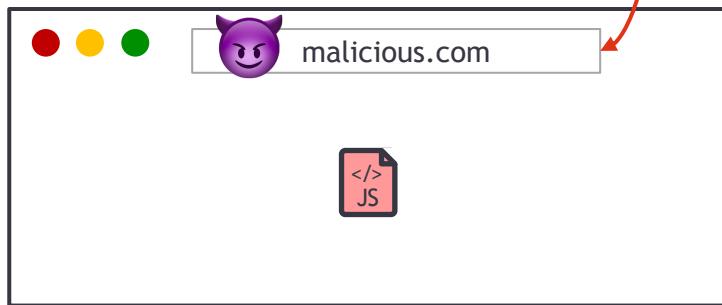
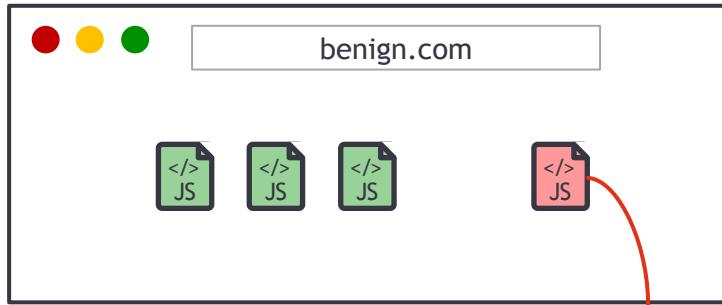


Malicious actor
Tax reimbursement
To: me



Background – JavaScript: Basis for Attacks

Malicious JavaScript



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



Malicious JavaScript

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

-but-buggy Benign JavaScript

- Designed by well-intentioned developers
- ... but contains some vulnerabilities
 - which 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

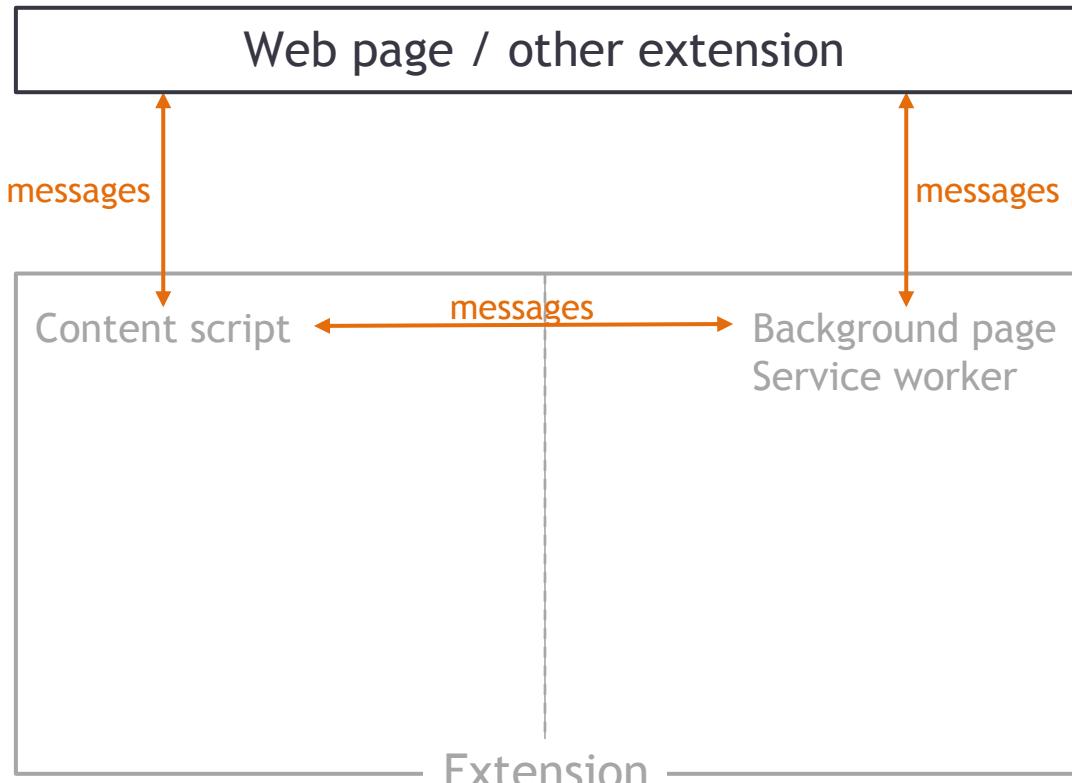
- https://developer.chrome.com/docs/extensions/mv3/declare_permissions/

- <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, ...)
 - ...

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

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 😈

→ 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?

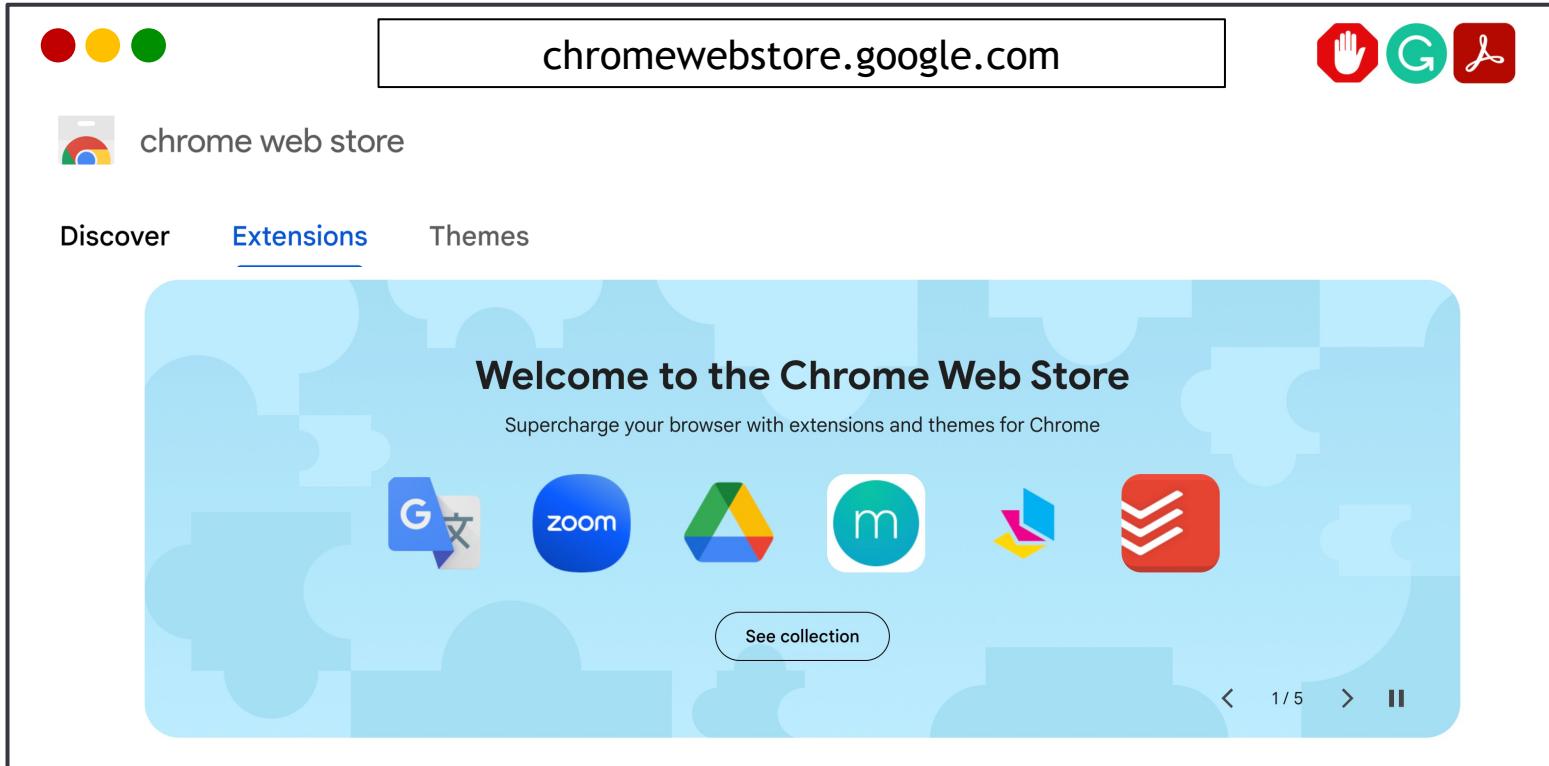


> What is in the Chrome Web Store?



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

The screenshot shows the homepage of chrome-stats.com. At the top, there's a navigation bar with three colored dots (red, yellow, green) on the left and a search bar containing "chrome-stats.com" on the right. Below the navigation bar is a banner with the text "Compare and analyze Chrome extensions" and "All-in-one platform for competitor research, risk analysis, and growth tracking". A search bar is also present on the banner. To the left of the main content area is a sidebar with the "Chrome-Stats" logo and a "Premium" badge. It includes sections for "Recently viewed" (Spotify™ & Deezer™ Music Downloader, GS Auto Clicker:Free Download 2021, Fraud Risk Scoring, Autoskip for Youtube, Maxi Refresher) and "Stats & analysis tools" (Chrome extension statistics, Extension explorer, Keyword explorer, Publisher explorer, Advanced search, Raw data download, Chrome-Stats extension). The main content area features two large numbers: "127862" under "Extensions" and "27638" under "Themes". Below these numbers is a line chart titled "Chrome Web Store stats" showing the number of extensions and themes over time from March 2021 to August 2021. The chart has two data series: "Extensions" (blue line with dots) and "Themes" (red line with dots). The "Extensions" series remains relatively stable around 127,000, while the "Themes" series shows a significant dip starting in late April 2021. At the bottom of the chart is a link "Explore more Chrome extension statistics". A footer note at the bottom of the page states: "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."

chrome-stats.com

Compare and analyze Chrome extensions

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

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

127862

Extensions

27638

Themes

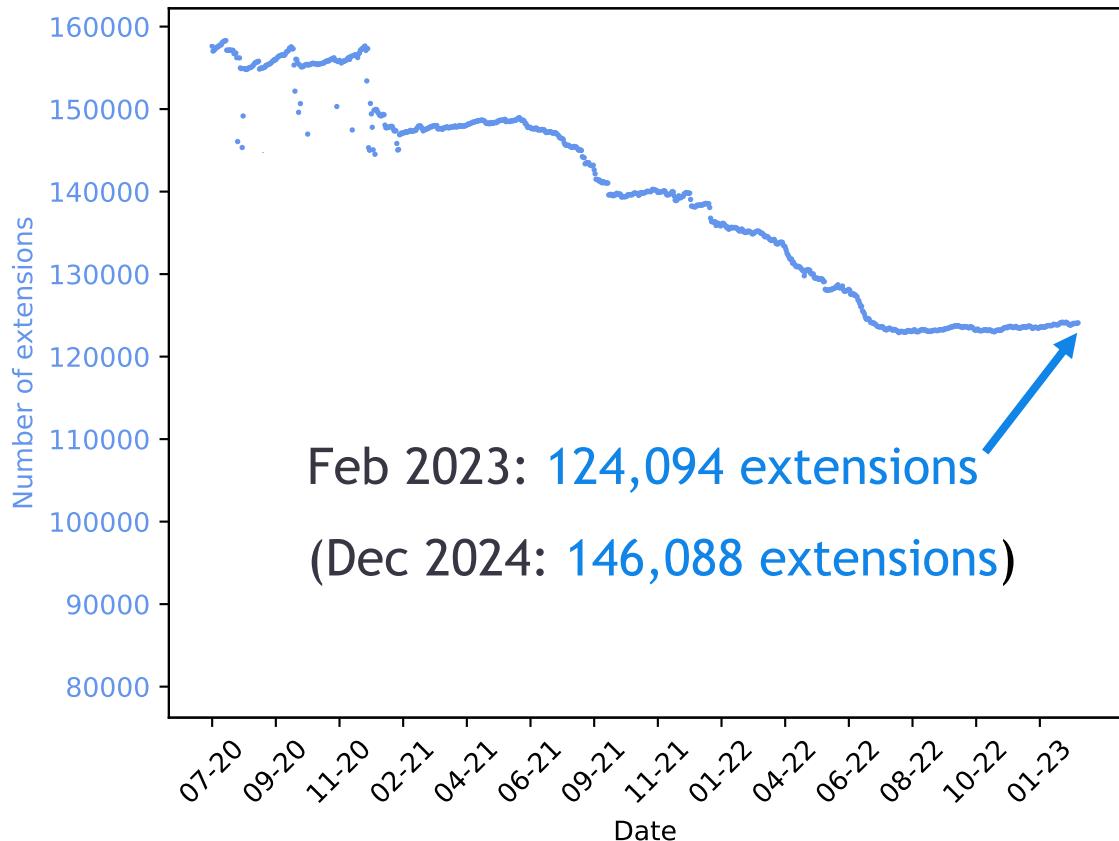
Chrome Web Store stats

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.

Number of Extensions in the Chrome Web Store

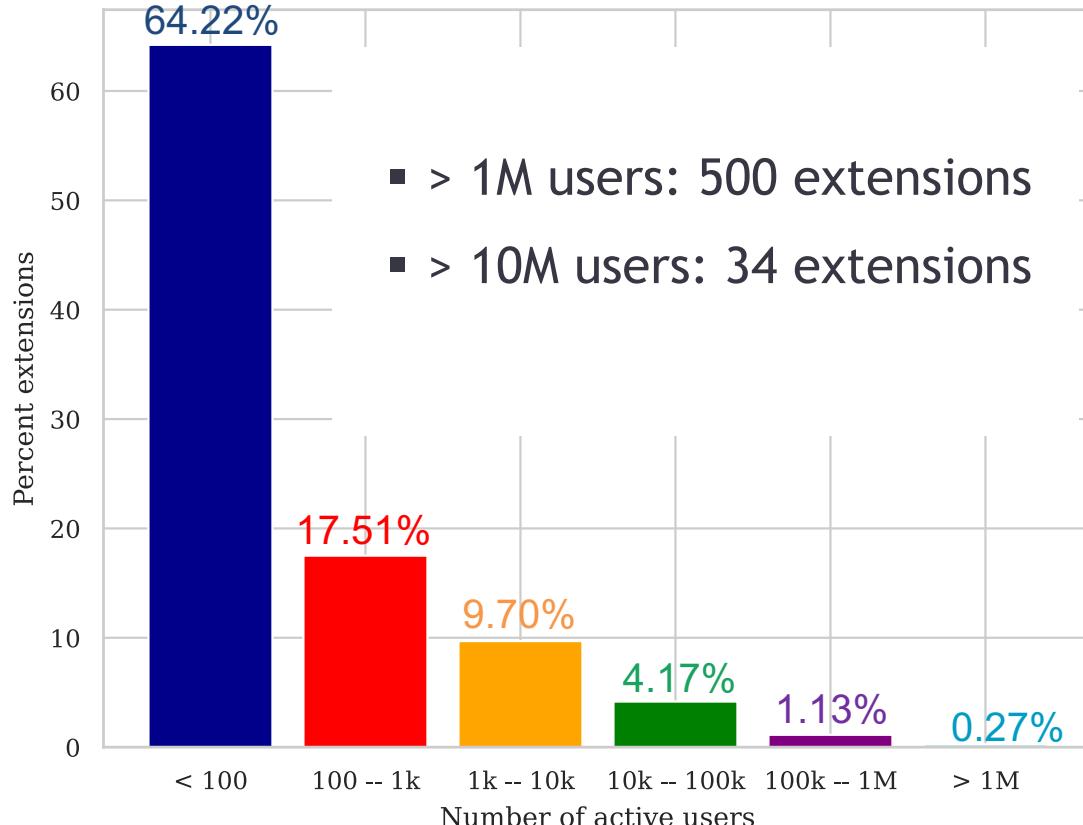


Every month:

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

➤ Analyses on the CWS
should be run regularly

Breakdown of Extension Users



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”

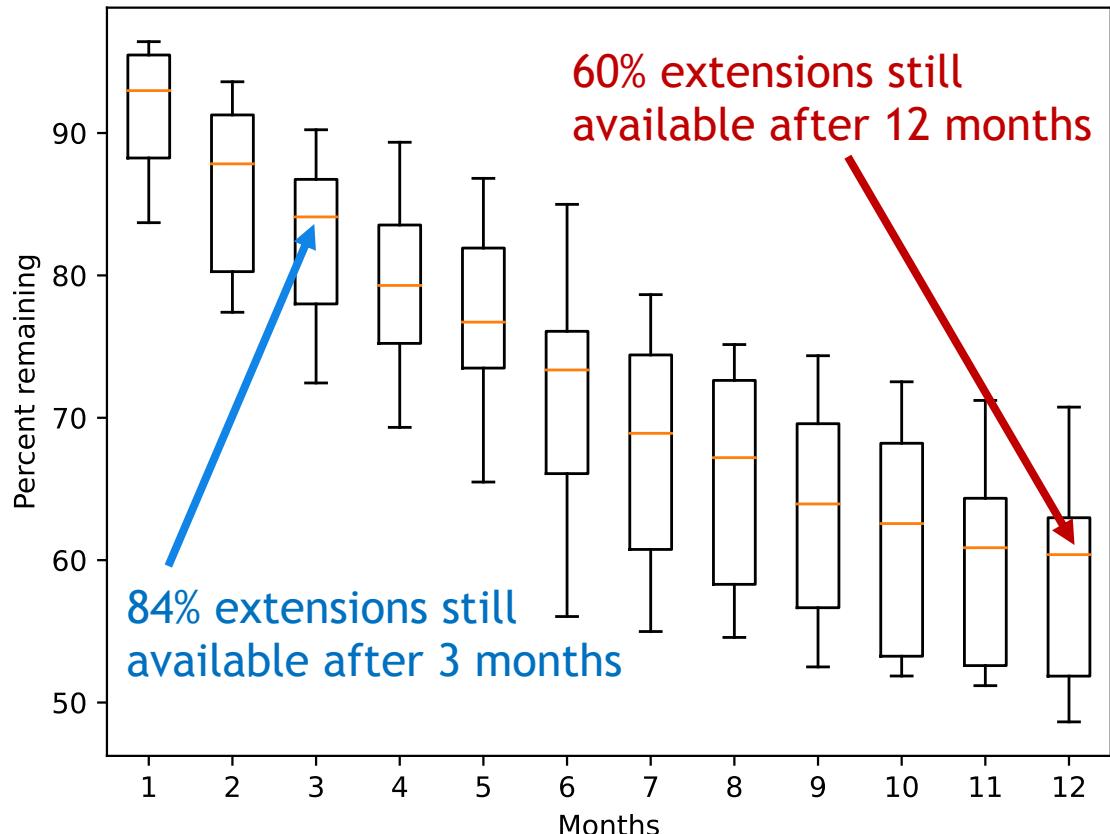
~ Chrome Web Store Developer Support

Life Cycle of Extensions

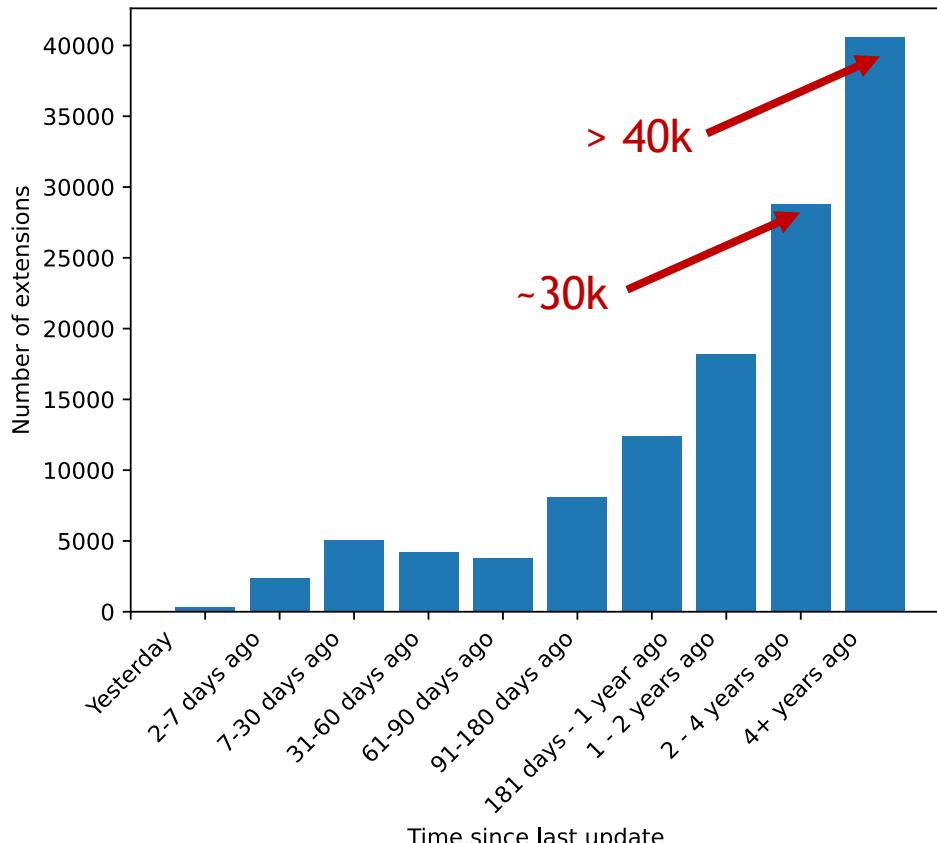
Methodology:

- 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

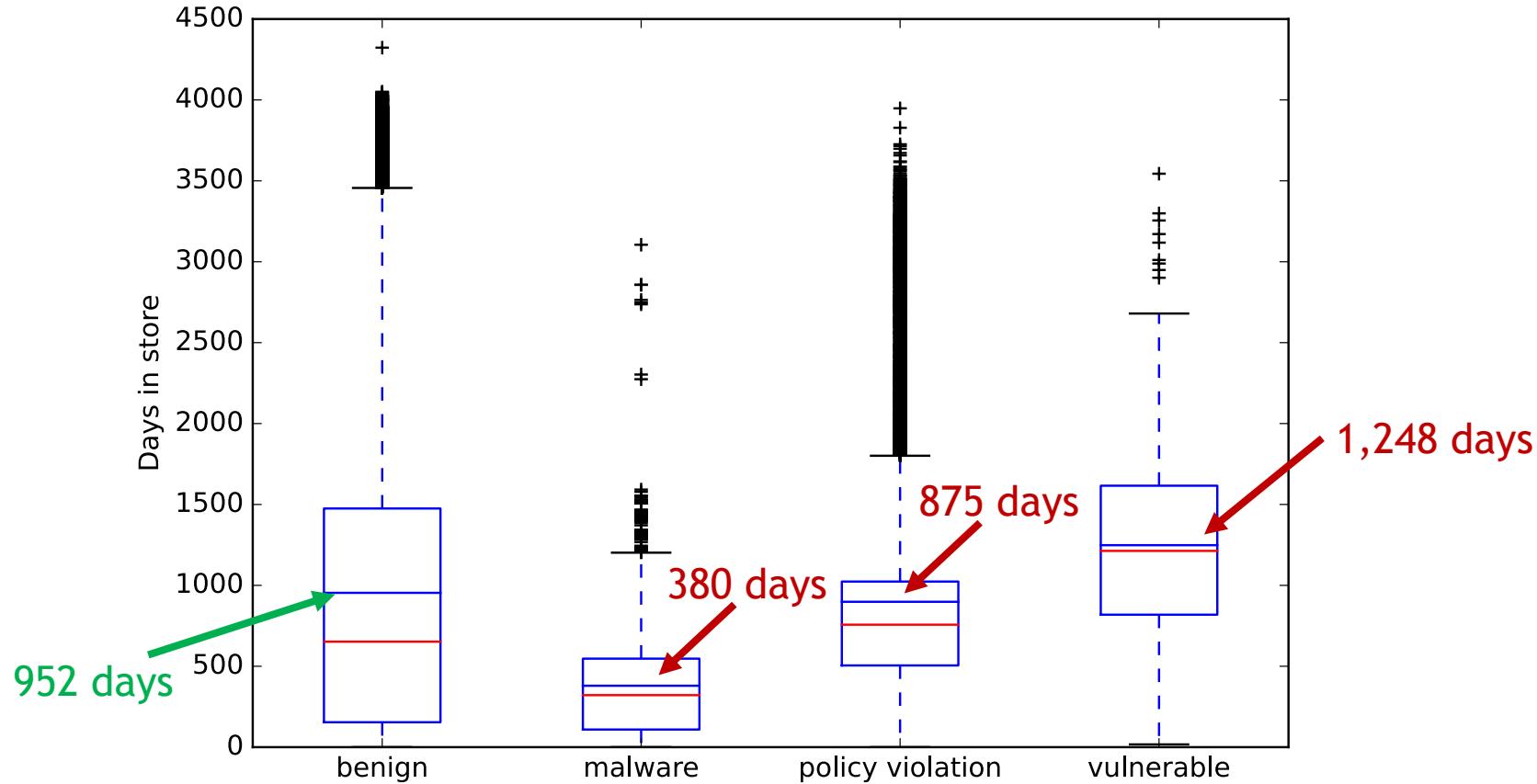
The screenshot shows a web browser window with the address bar containing "chrome-stats.com". The main content area is titled "Advanced search". A red oval highlights the search query input field, which contains "obsoleteReason" followed by a dropdown arrow, an equals sign, and the word "malware". Below this is a blue "Search" button. To the right of the search bar are buttons for "Export", "Saved query", "Visible columns", and "+ Add condition". The results table has a header row with columns: logo, name, userCount, author, ratingValue, ratingCount, obsoleteReason, lastUpdate, and creationDate. The table lists six extension entries, each with a small icon, the extension name, its user count, author, rating, and last update information. The first entry, "Video downloader for Instagram™", is highlighted with a light gray background.

| logo | name | userCount | author | ratingValue | ratingCount | obsoleteReason | lastUpdate | creationDate |
|------|--|-----------|--|-------------|-------------|----------------|------------|--------------|
| ⚠️ | Video downloader for Instagram™ | 100 000 | 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 | 9 000 | 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 | 50 000 | https://snake.9834722.xyz | 4.19 | 52 | malware | 2024-02-27 | 2021-10-04 |
| ⚠️ | Settings for Chrome | 600 000 | Chrome Settings | 3.75 | 4 | malware | 2024-02-27 | 2022-06-24 |

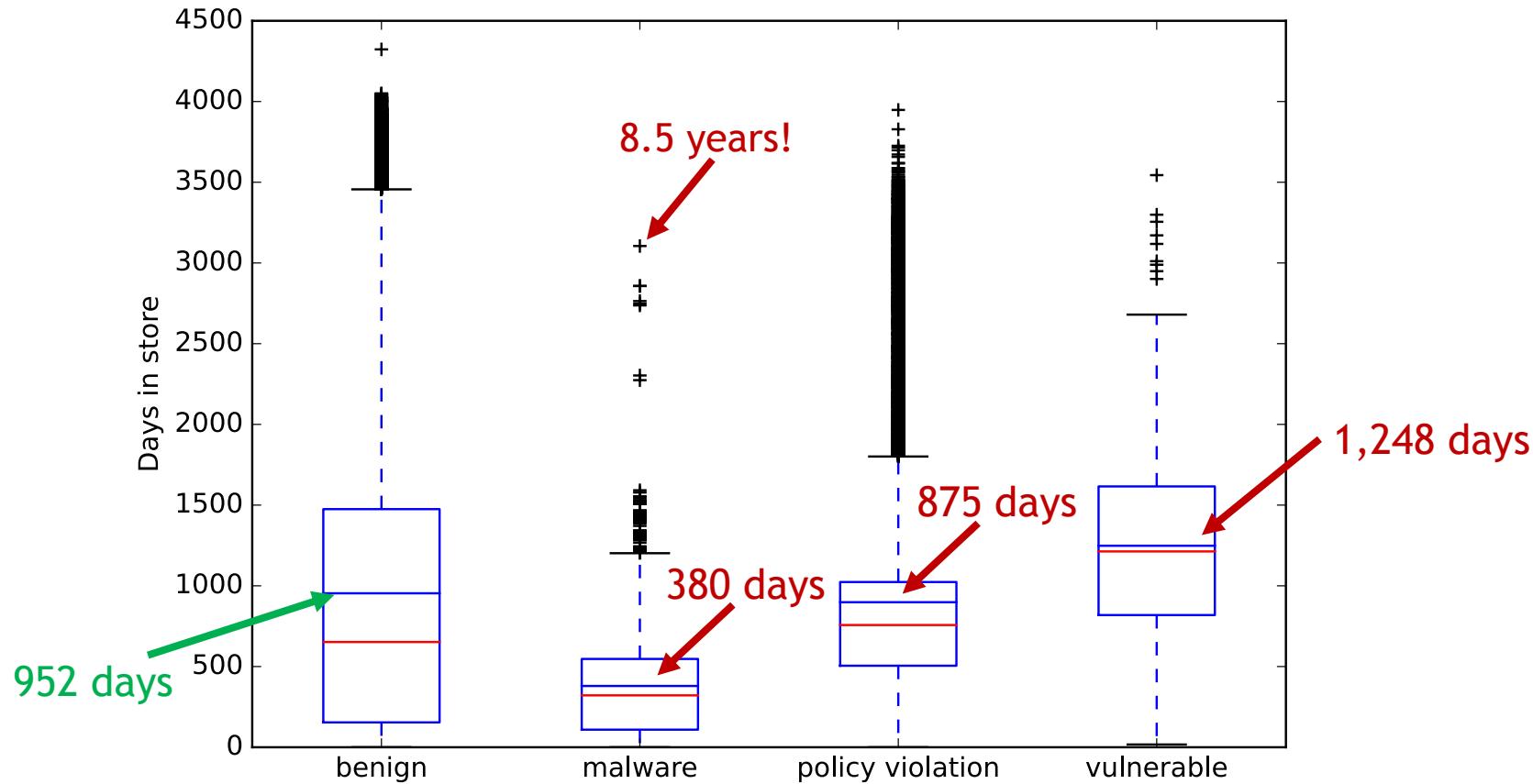
Browser Extension Collection: Chrome-Stats

| 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 |

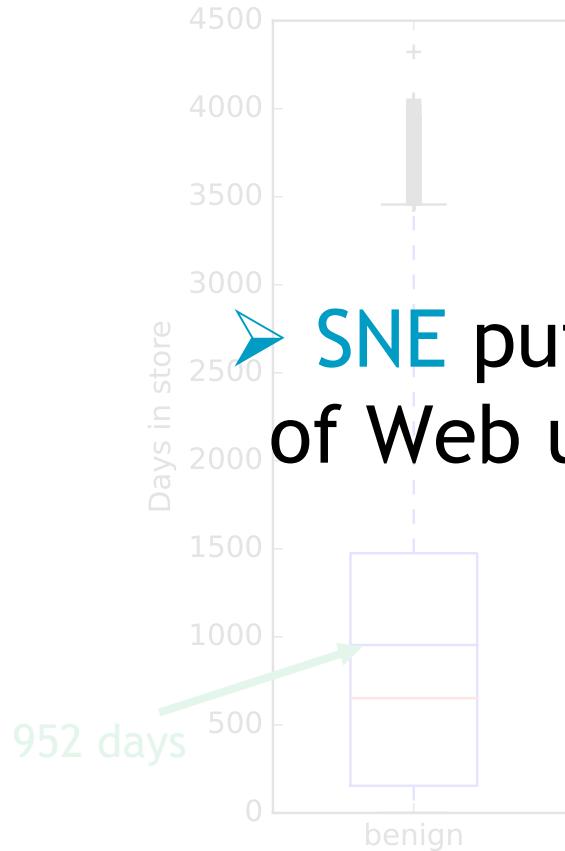
Number of Days in the CWS



Number of Days in the CWS



Number of Days in the CWS

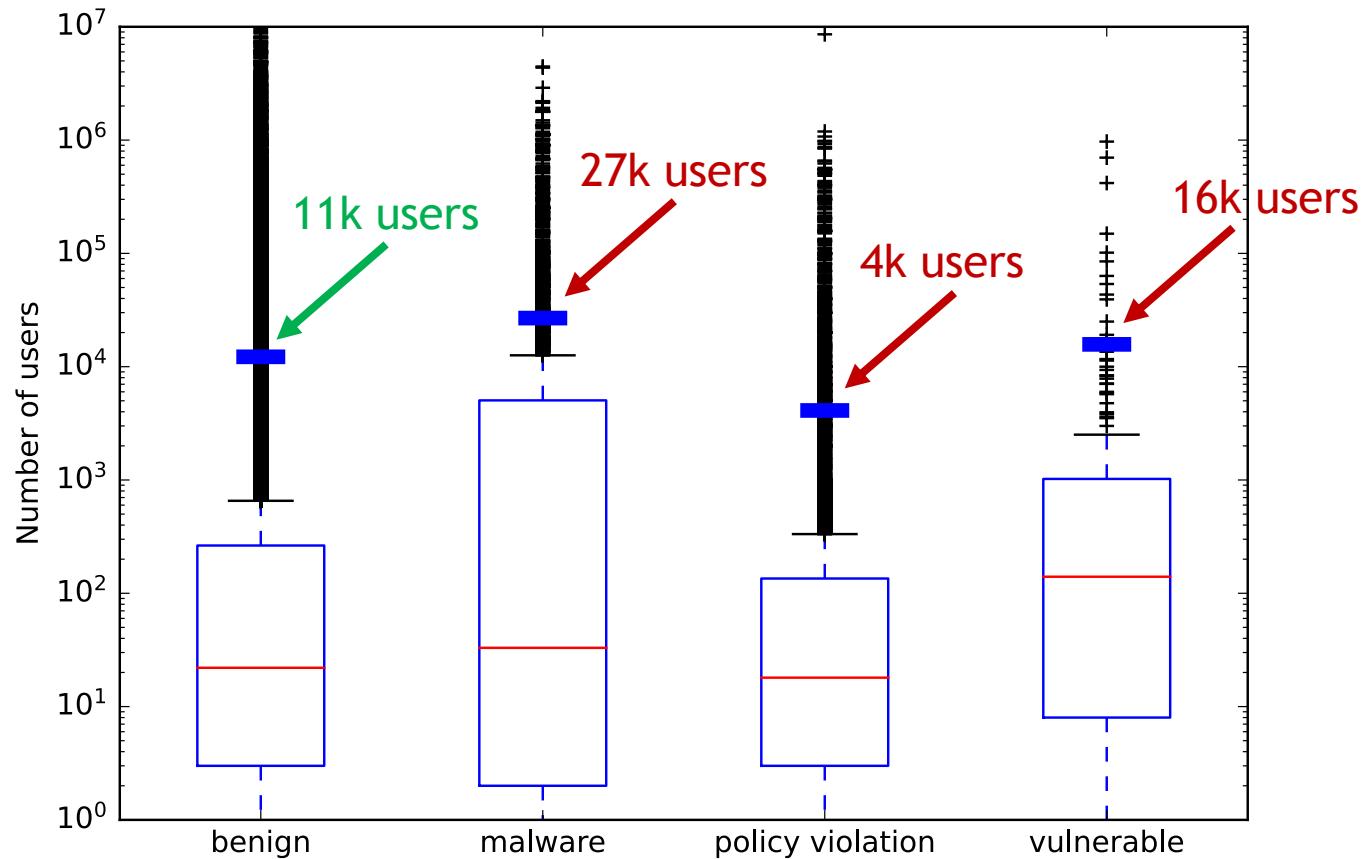


➤ SNE put the security & privacy
of Web users **at risk *for years***

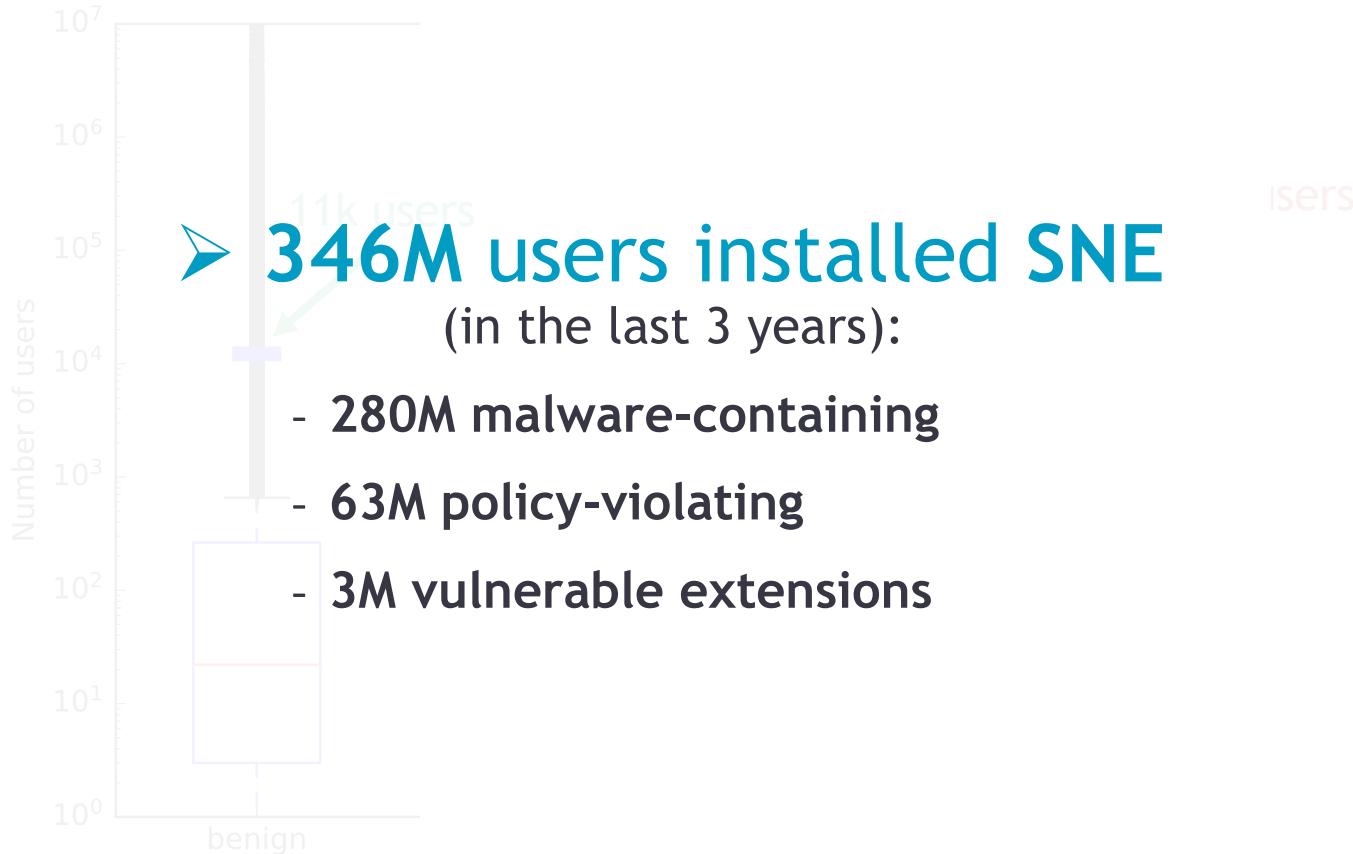
,248 days

952 days

Number of Users



Number of Users



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.

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

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*)



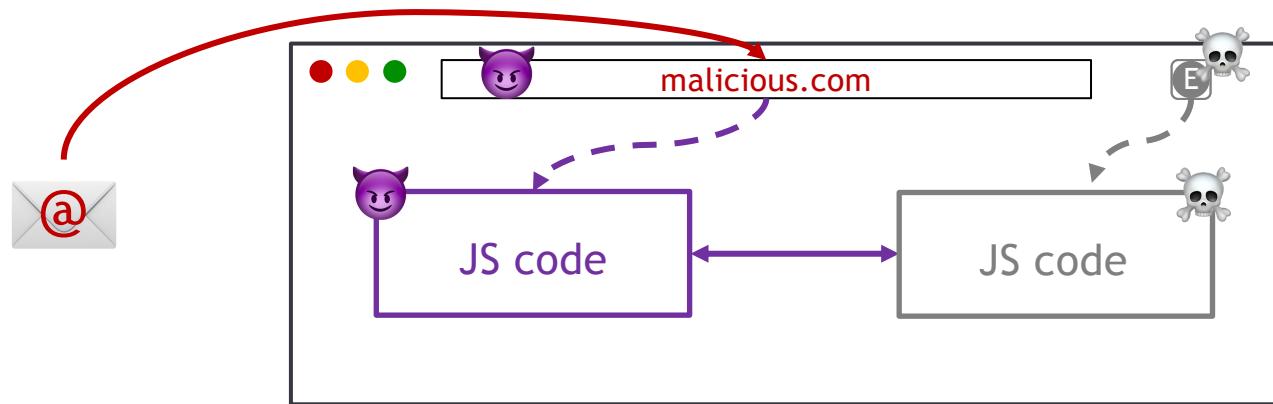
Analysis of Vulnerable Extensions: Web Attacker

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



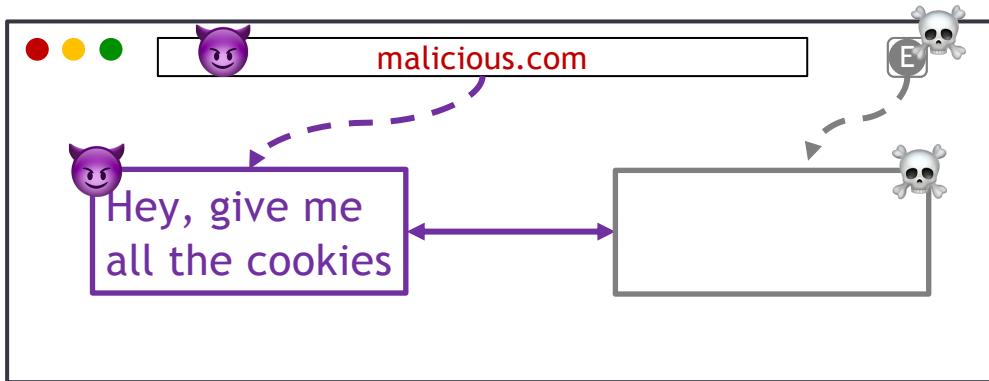
Analysis of Vulnerable Extensions: Web Attacker

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



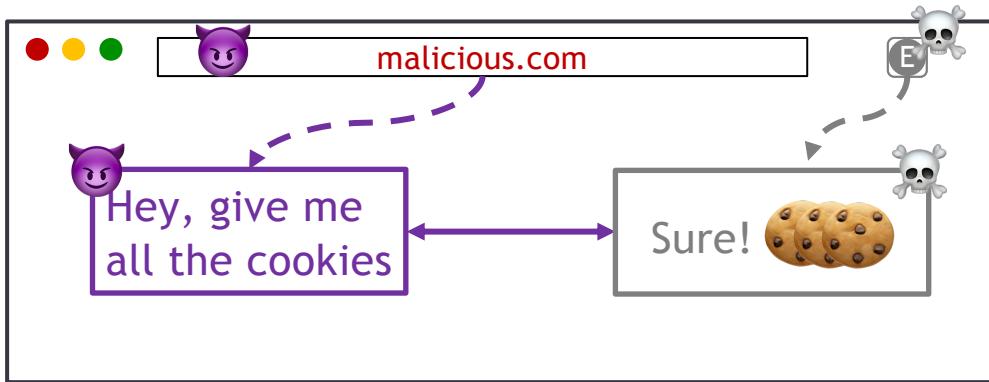
Analysis of Vulnerable Extensions: Web Attacker

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



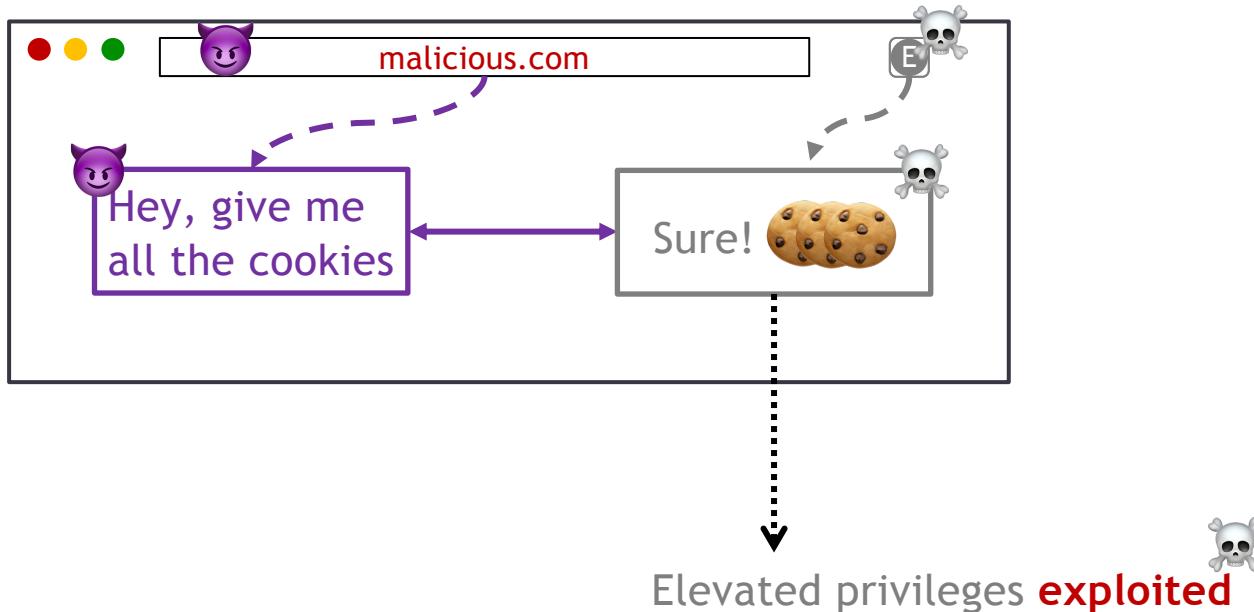
Analysis of Vulnerable Extensions: Web Attacker

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



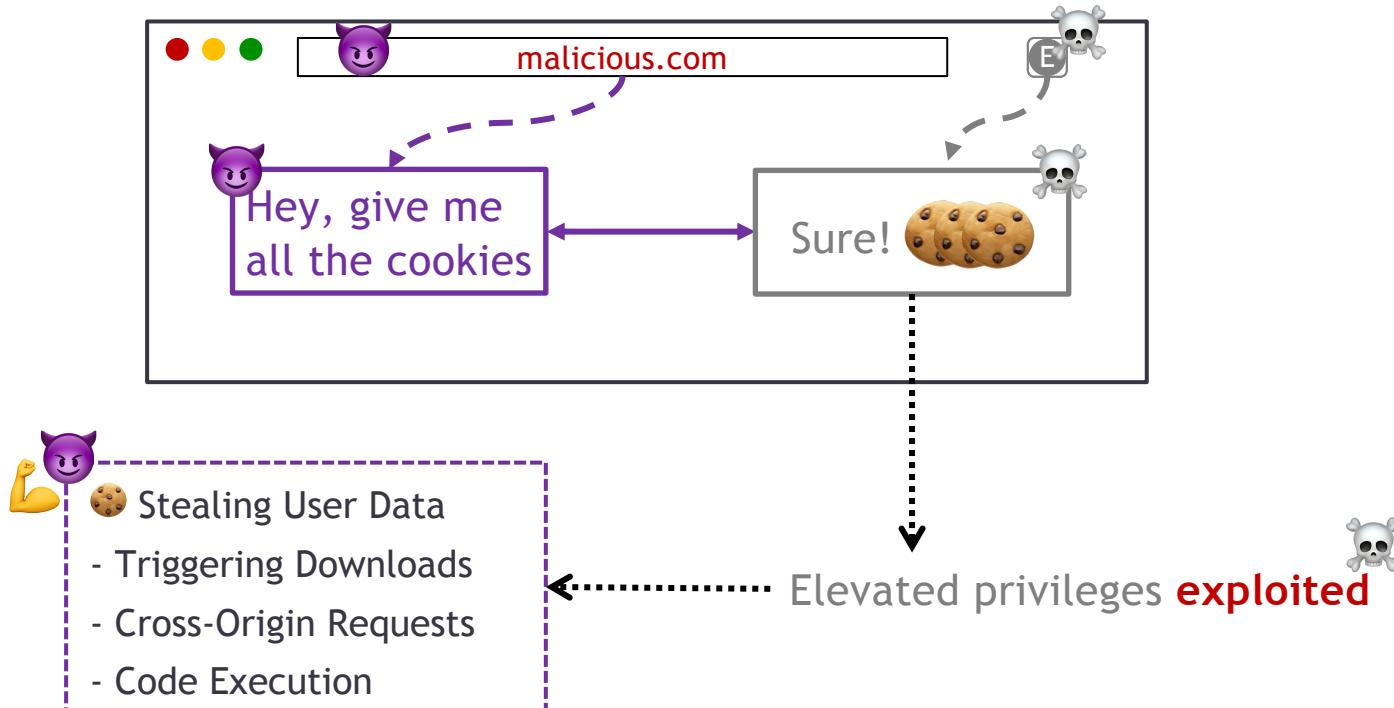
Analysis of Vulnerable Extensions: Web Attacker

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*)

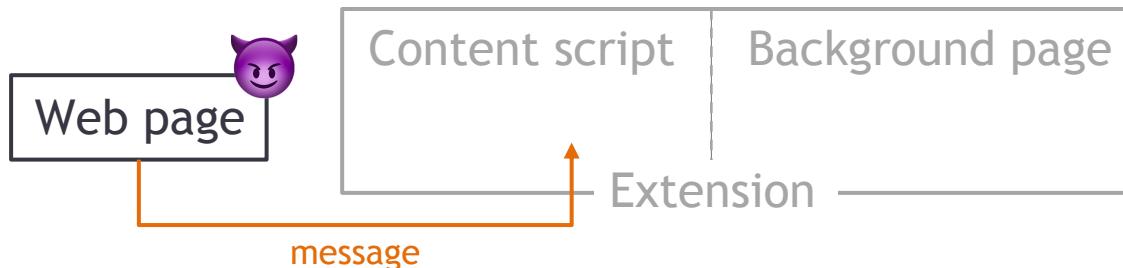


Background – postMessage

- 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

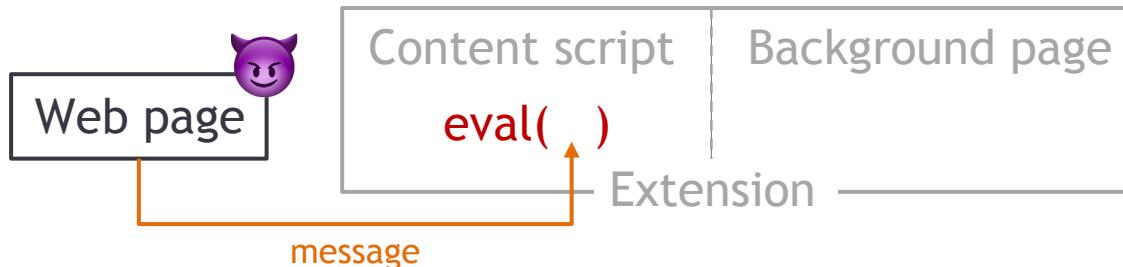
Simplified Example of a Vulnerability

```
// 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

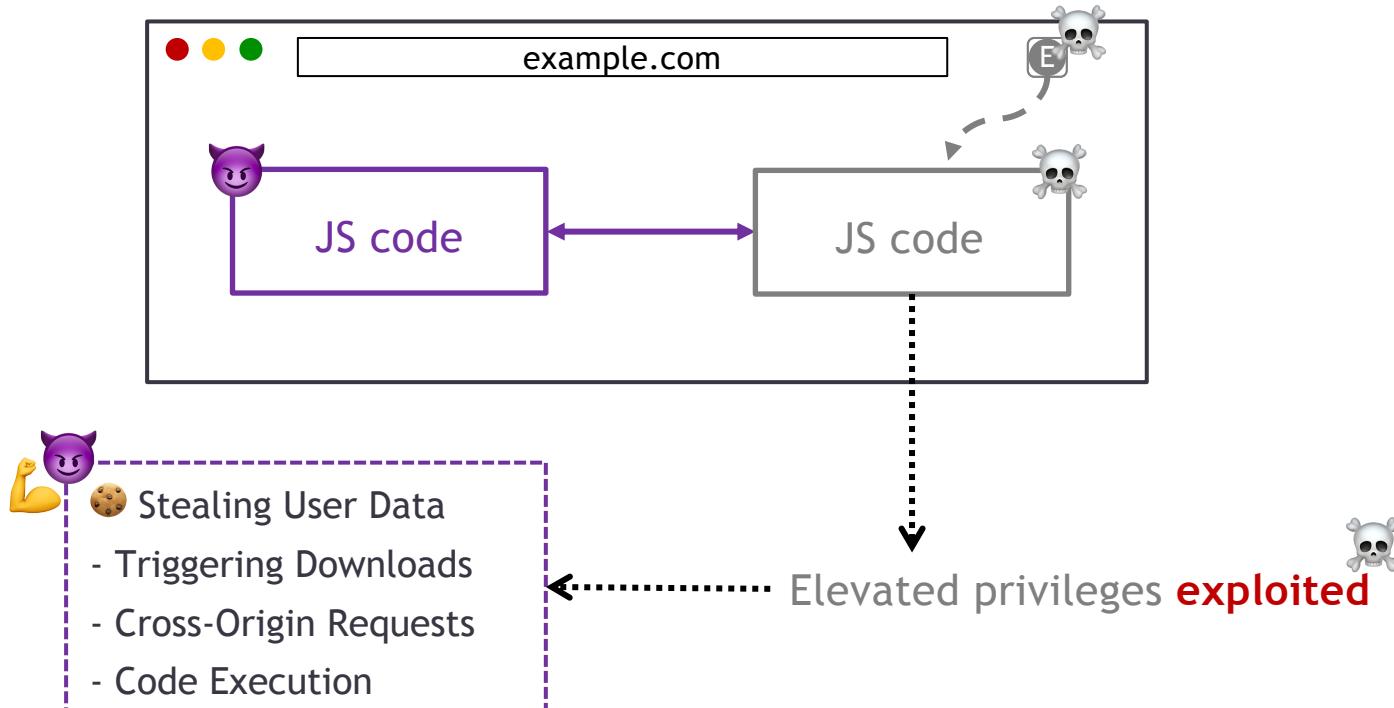
developer.chrome.com indique

1

OK

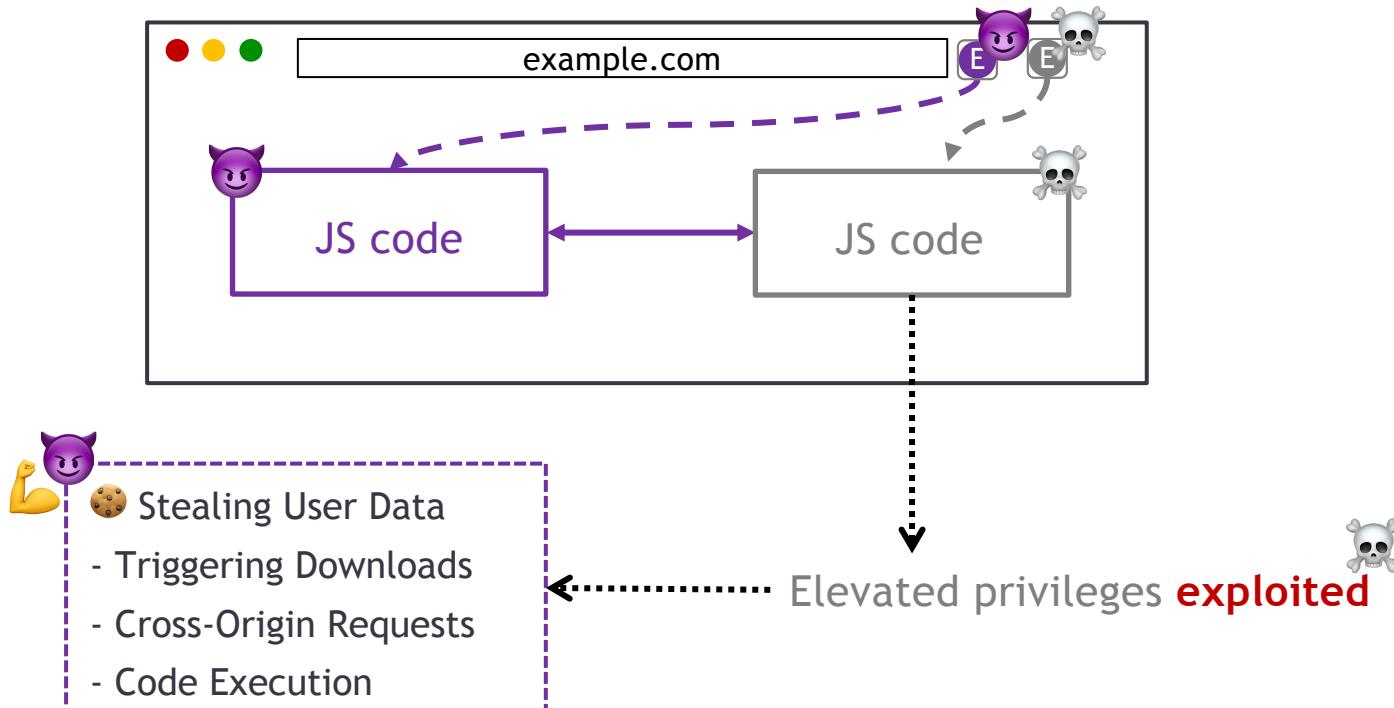
Analysis of Vulnerable Extensions: Confused Deputy

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



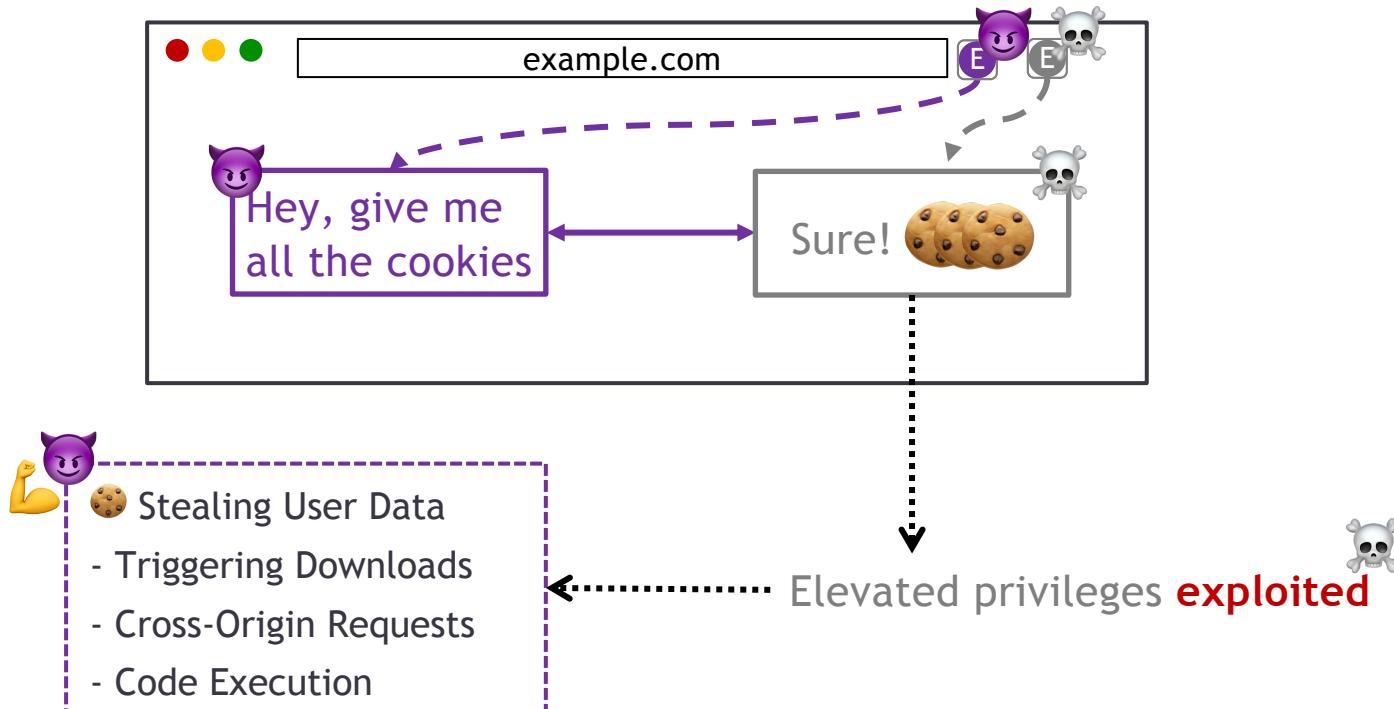
Analysis of Vulnerable Extensions: Confused Deputy

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

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



Detecting Vulnerable Extensions



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



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



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



Malicious web page



Detecting Vulnerable Extensions



CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

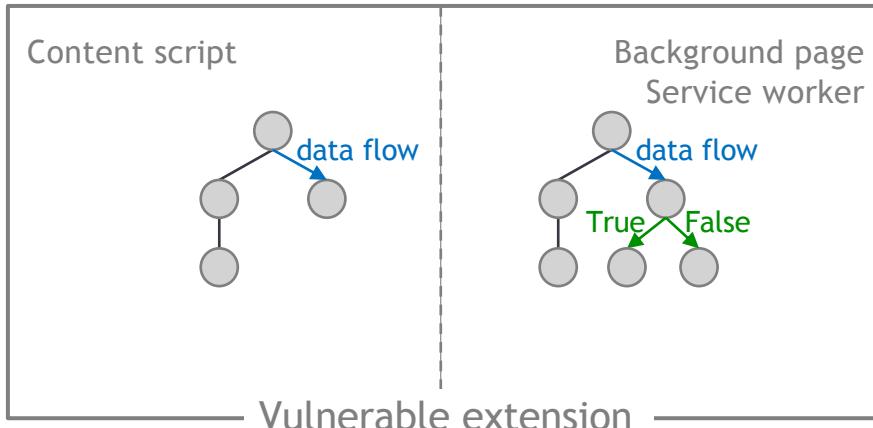


> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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



Malicious web page



Per-component JS code abstraction

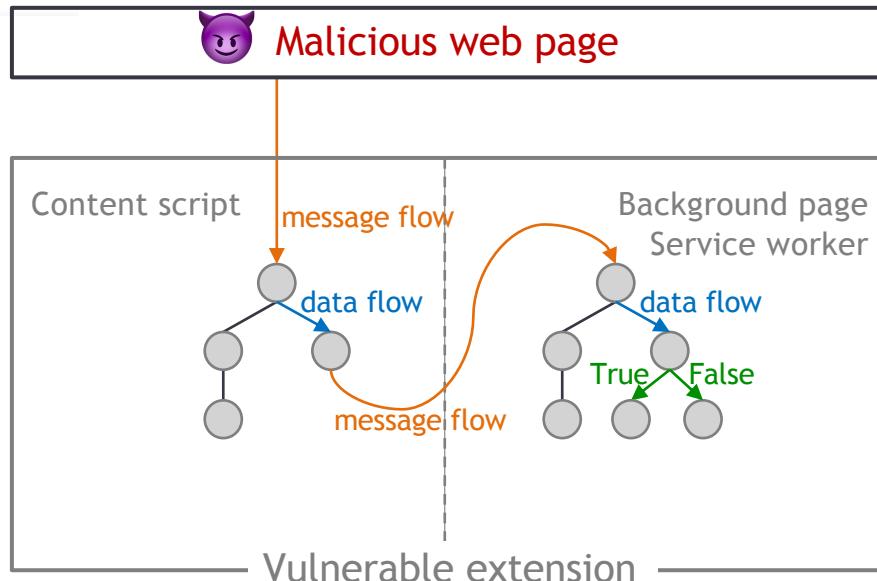
- AST (Abstract Syntax Tree)
- Control flow
- Data flow
- Pointer analysis

Detecting Vulnerable Extensions



> DOUBLEX: Statically Detecting Vulnerable Data Flows in Browser Extensions

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



Per-component JS code abstraction

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

Extension Dependence Graph (EDG)

- Message interactions

Detecting Vulnerable Extensions

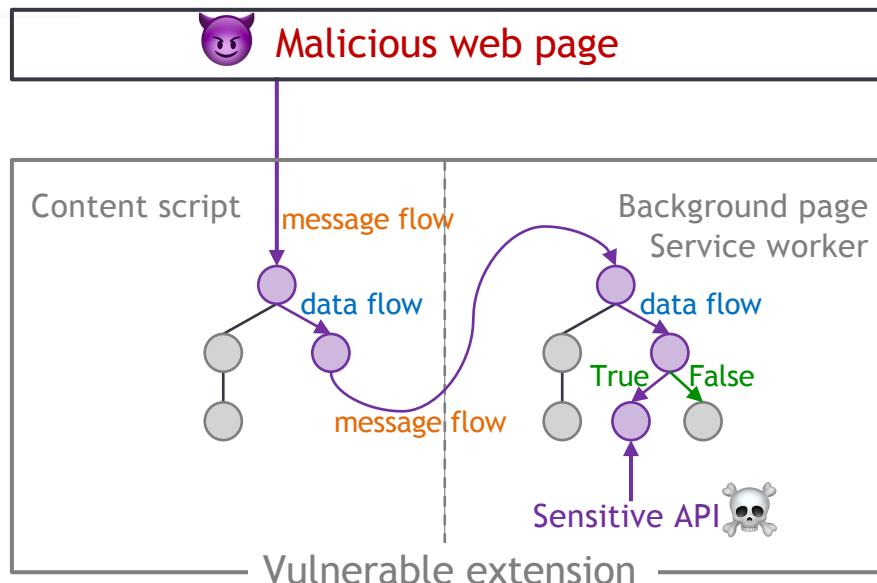


CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY



> 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

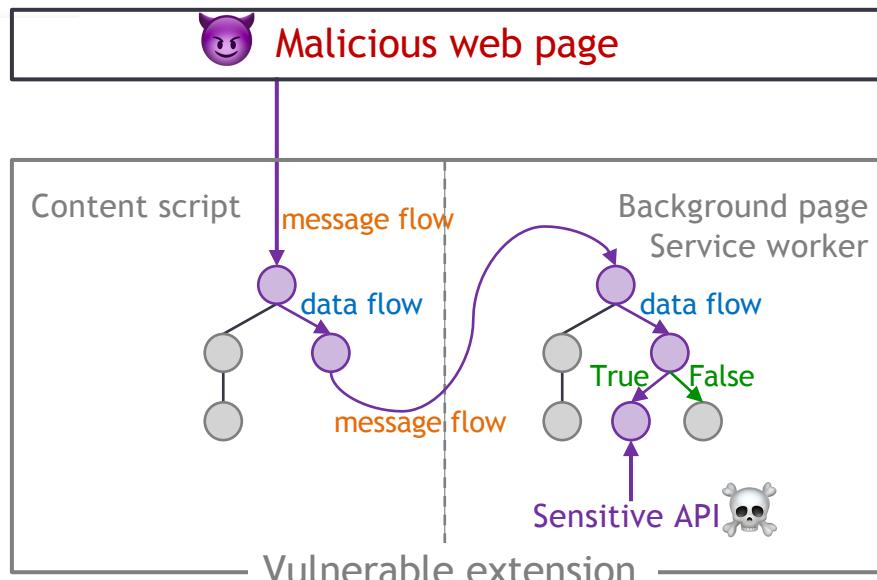


CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY



> 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)

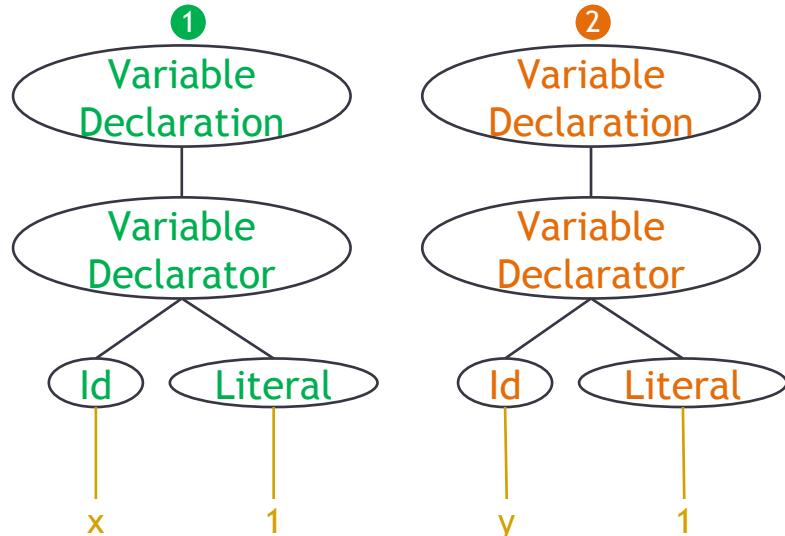
- Message interactions

Suspicious data flow tracking

- Detects any path between an attacker & sensitive APIs

Abstract Syntax Tree

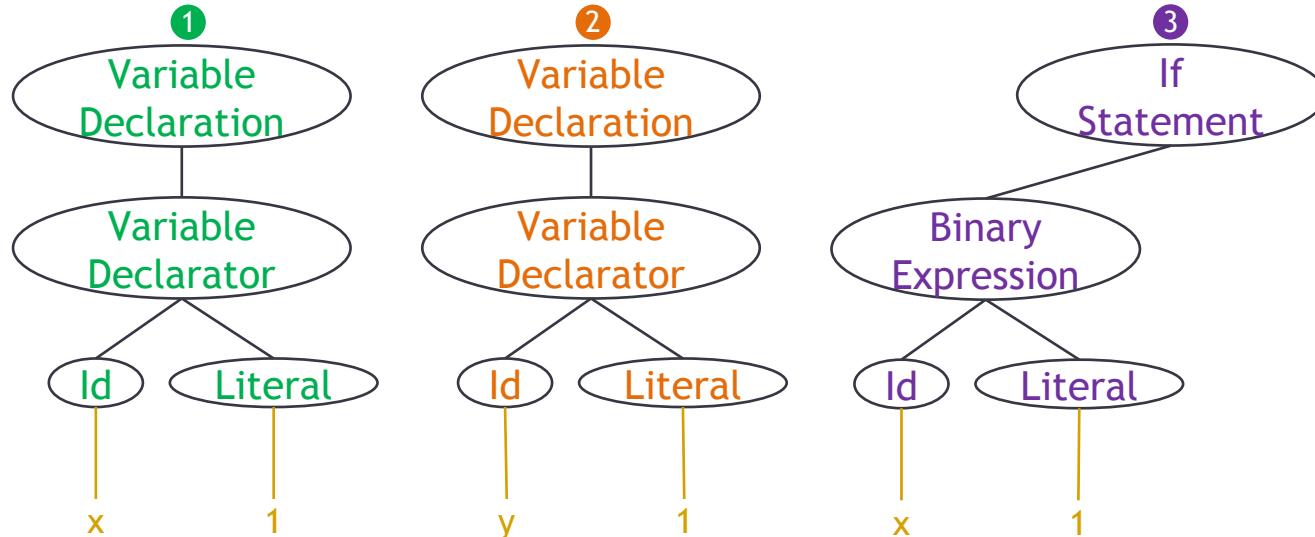
- AST: nesting of programming constructs



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

Abstract Syntax Tree

- AST: nesting of programming constructs

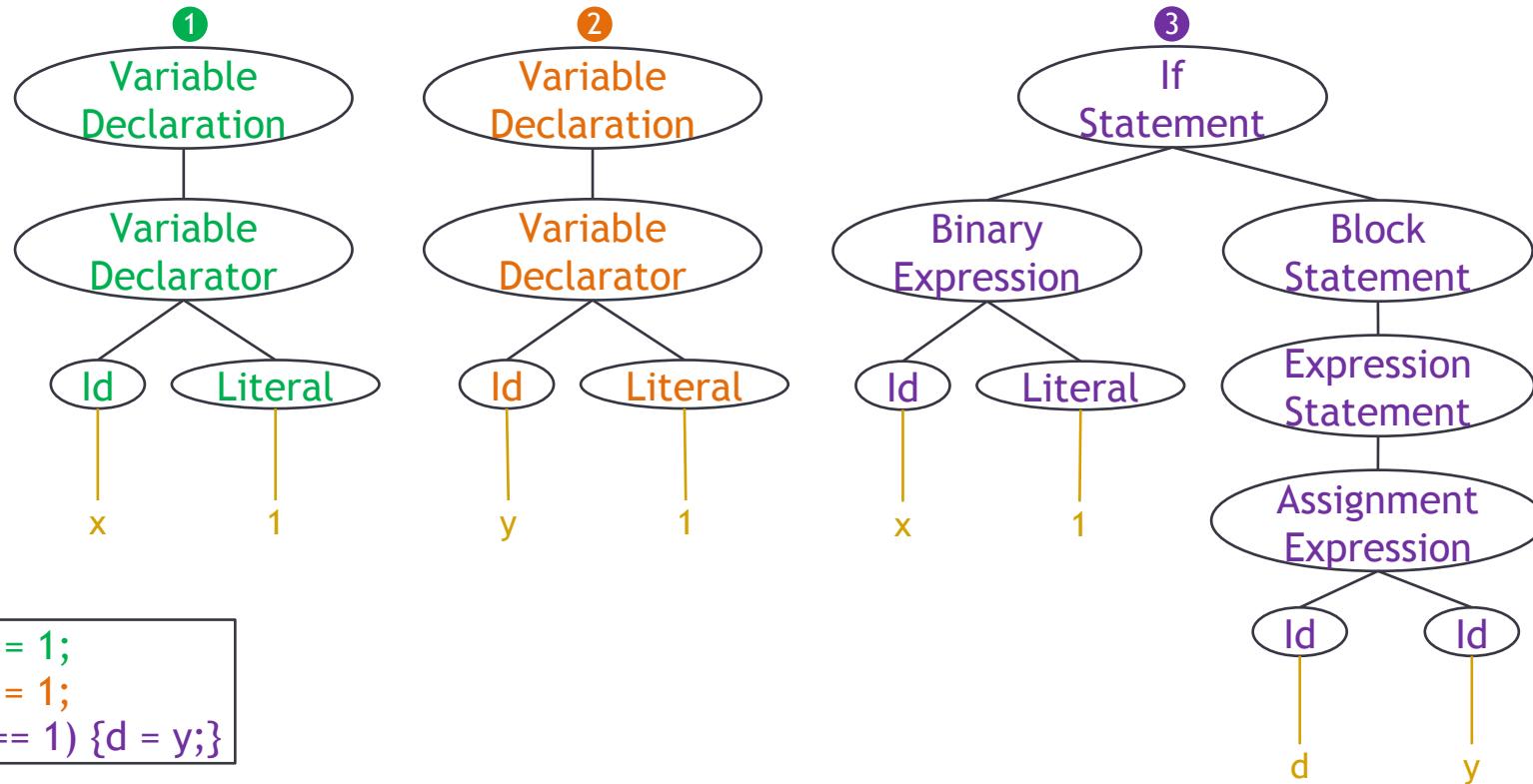


```

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

Abstract Syntax Tree

- AST: nesting of programming constructs

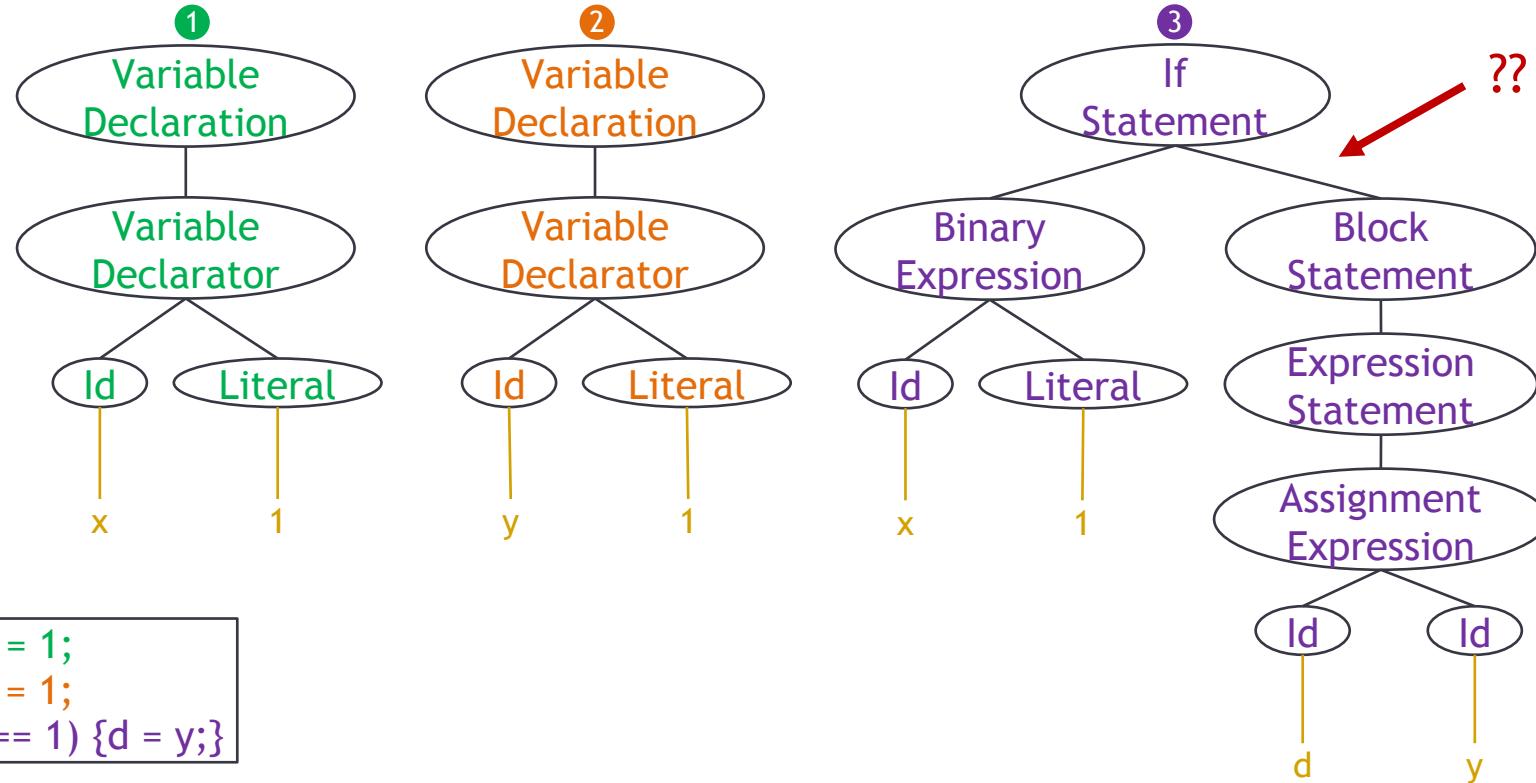


```

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

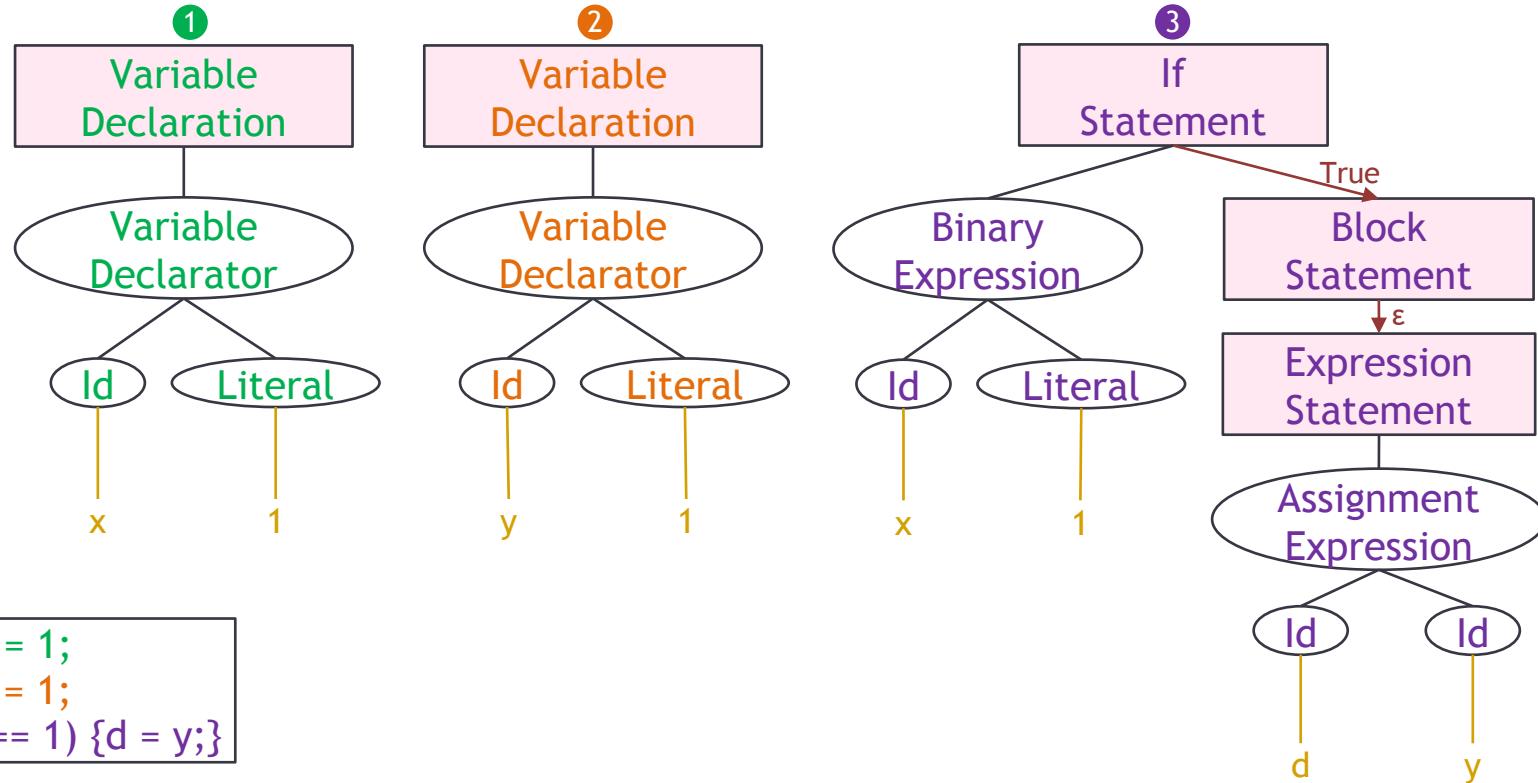
Abstract Syntax Tree

- AST: nesting of programming constructs



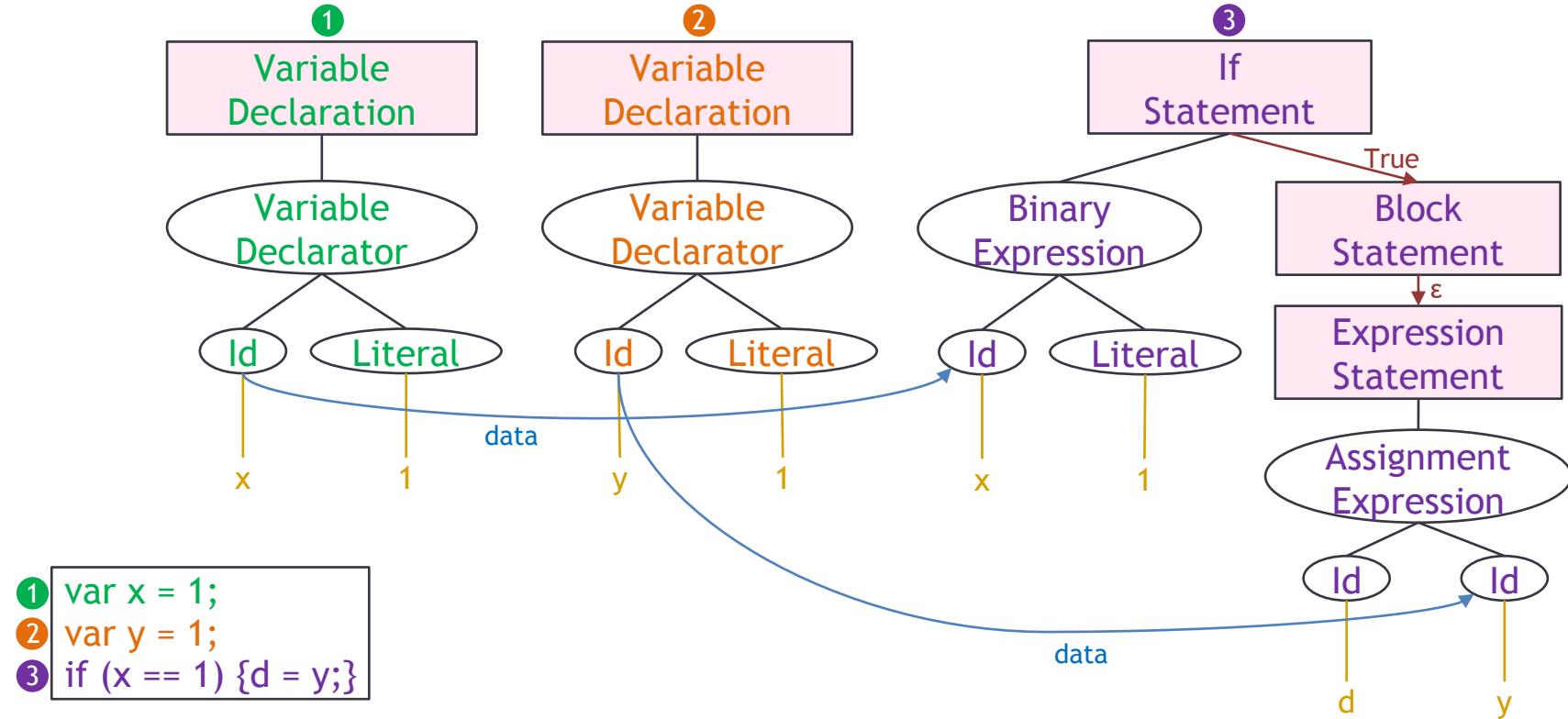
Control Flow Graph

➤ CFG: execution path conditions



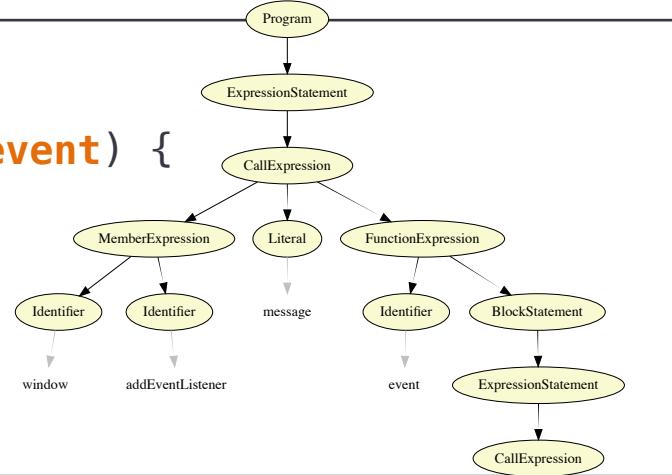
Program Dependence Graph

➤ PDG: variable dependency



Per-Component JS Code Abstraction

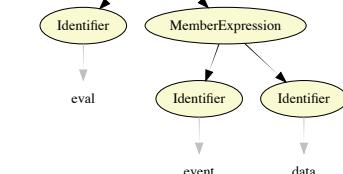
```
// Content script code  
window.addEventListener("message", function(event) {  
    eval(event.data);  
})
```



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

```
// Content script code
window.addEventListener("message", function(event) {
    if (1 === 1) {
        eval(event.data);
    }
})
```

The diagram illustrates data flow analysis on the provided JavaScript code. A blue curved arrow labeled 'data' points from the variable 'event.data' in the code to the argument of the 'eval' function. A green curved arrow labeled 'True' points from the condition '1 === 1' in the 'if' block to the start of the code block. The code block itself is highlighted with a yellow box.

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



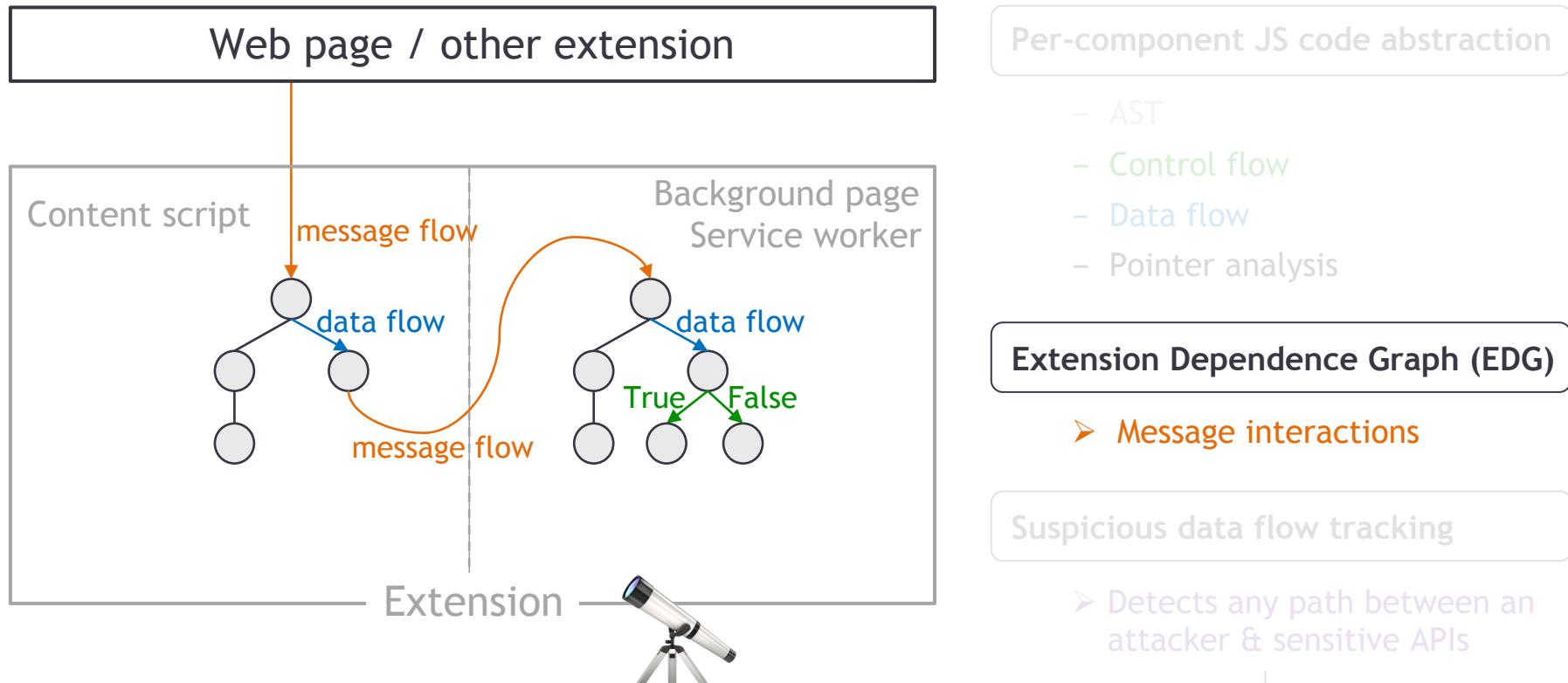
data flow

– 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) {
        True
        window["e" + "val"](event.data);
    }
})
```



- external messages
- internal messages

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

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



- external messages 
- internal messages

Extension Dependence Graph

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

The diagram shows a snippet of Content Script code. A purple devil icon is positioned above the word 'event'. A green curved arrow labeled 'True' points from the condition '1 === 1' to the 'eval' expression 'window["e" + "val"]'. A blue curved arrow labeled 'data' points from the variable 'event.data' to the same 'eval' expression. This indicates that the value of 'event.data' is being injected into the string 'e' + 'val'.

- 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});
```

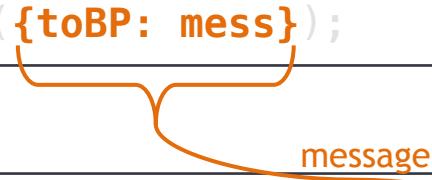
```
// Background page code
chrome.runtime.onMessage.addListener(function(request) {
})
```

- external messages 
- internal messages

Extension Dependence Graph

```
// 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});
```

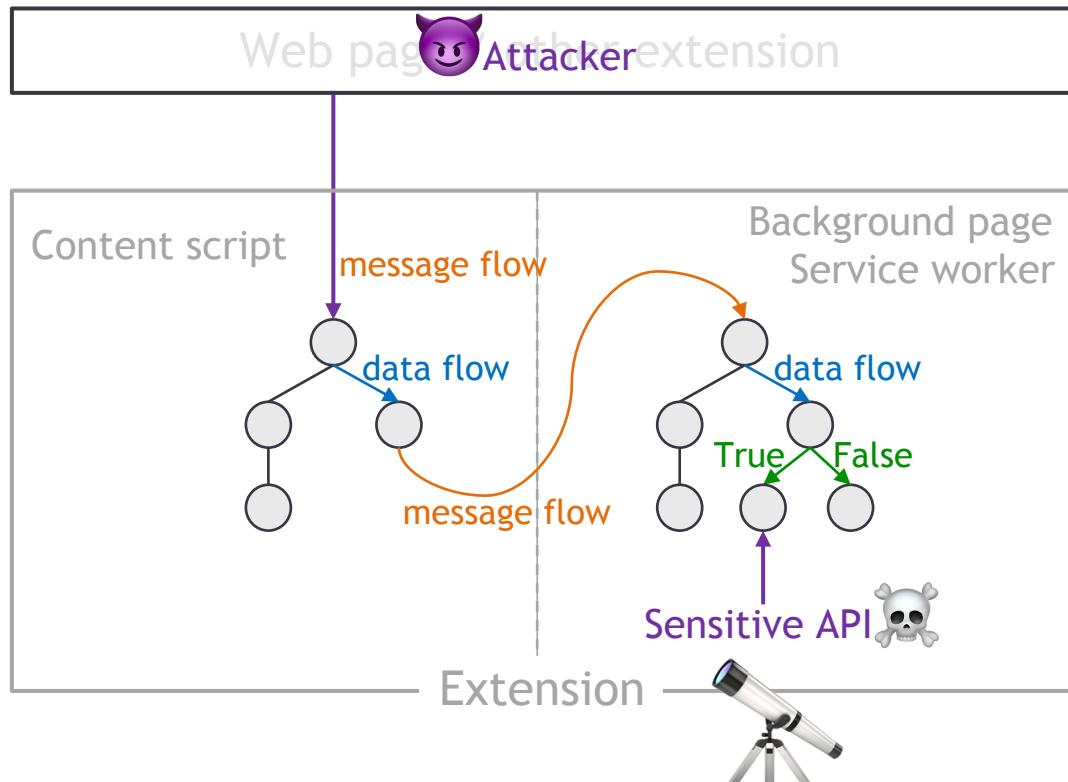
```
// Background page code  
chrome.runtime.onMessage.addListener(function(request) {  
})
```

message

- external messages 
- internal messages 

➤ Models message interaction within and outside of an extension

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

Suspicious Data Flow Tracking

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

The diagram illustrates the flow of data in the provided JavaScript code. A blue curved arrow labeled 'data' points from the variable `event.data` in line 4 to the argument of the `window["e" + "val"]` call in line 4. A red bracket labeled 'eval' spans the string concatenation `"e" + "val"`. A green curved arrow labeled 'True' points from the condition `1 === 1` in line 3 to the opening brace of the if-block in line 4.

Suspicious Data Flow Tracking

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

The diagram illustrates the flow of data in the provided JavaScript code. A blue curved arrow labeled 'data' points from the variable `event.data` in line 4 to the argument of the `window["e" + "val"]` call in line 4. A red bracket labeled 'eval' encloses the string concatenation `"e" + "val"`. A green curved arrow labeled 'True' points from the condition `1 === 1` in line 3 to the opening brace of the if-block in line 4.

Suspicious Data Flow Tracking

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

The diagram illustrates the flow of data in the provided JavaScript code. The variable `event` is highlighted in yellow and is passed as an argument to the `eval` function. The `eval` function is highlighted in yellow and receives the string `"e" + "val"` as its argument. The resulting expression `window["e" + "val"]` is also highlighted in yellow and is called with the `data` parameter, which is also highlighted in yellow. A purple devil icon is positioned above the `event` variable, symbolizing malicious intent or a security vulnerability.

Suspicious Data Flow Tracking

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

The diagram illustrates the execution flow of the provided JavaScript code. A purple devil icon is positioned above the word 'event' in the event handler. A green curved arrow labeled 'True' points from the condition '1 === 1' to the red bracket under the line 'window["e" + "val"]'. A red bracket labeled 'eval' is placed under the string '["e" + "val"]'. A blue curved arrow labeled 'data' points from the variable 'data' in the event object to the parameter of the eval call.



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 })
```

The diagram illustrates the execution flow of the provided JavaScript code. A purple devil icon is positioned above the code. A green curved arrow starts from the value 'True' at line 1 and points to the red bracket under the line 'window["e" + "val"]'. A blue curved arrow starts from the yellow box 'data' at line 4 and points to the yellow box 'eval' at line 4. The word 'eval' is highlighted in yellow.



```
// 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 })
```

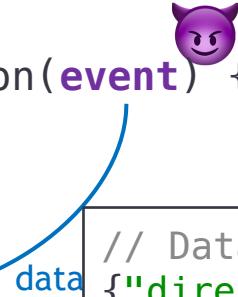
The diagram illustrates a security vulnerability in a content script. It shows the original code with annotations: a green arrow labeled 'True' points from the condition '1 === 1' to the 'eval' call; a red bracket labeled 'eval' encloses the 'eval' call; and a blue arrow labeled 'data' points from the 'data' parameter in the event object to the 'eval' call.



```
// 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     True → window["e" + "val"](event.data);
5     ↳ eval
6   }
7 }
8 }
9 }
10 }
```



```
// 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     True → window["e" + "val"](event.data);
5     eval
6       data
7       event = {"data": 42};
8       eval(event.data);
9   }
10 })
```

The diagram illustrates the data flow in the code. A green arrow labeled 'True' points from the condition '1 === 1' to the call to `window["e" + "val"]`. A red bracket labeled 'eval' encloses the assignment to `event = {"data": 42};` and the call to `eval(event.data);`. A blue arrow labeled 'data' points from the variable `event` in the assignment statement to the parameter `event` in the call to `eval`. A blue arrow labeled 'data' also points from the parameter `event` in the call to `eval` to a purple devil icon above the code.

```
// 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

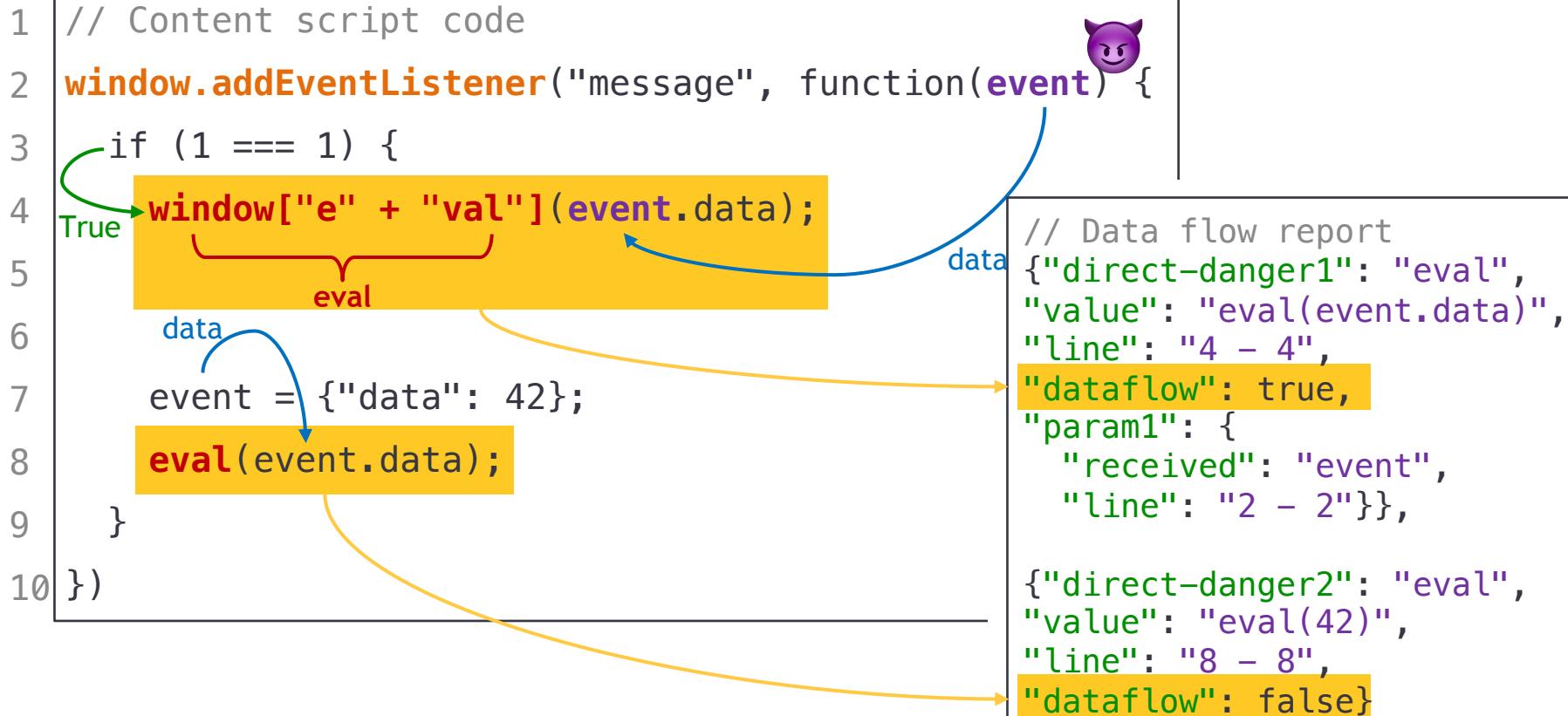
```
1 // Content script code
2 window.addEventListener("message", function(event) {
3   if (1 === 1) {
4     True → window["e" + "val"](event.data);
5     eval
6       data
7       event = {"data": 42};
8       eval(event.data);
9   }
10 })
```

The diagram illustrates the data flow in the code. A green arrow labeled 'True' points from line 4 to line 5. A red bracket labeled 'eval' encloses line 5. A blue arrow labeled 'data' points from line 5 to a purple devil icon at the top right. Another blue arrow labeled 'data' points from line 7 to line 8.

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

{"direct-danger2": "eval",
"value": "eval(42)",
"line": "8 - 8",
"dataflow": false}
```

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

```
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
10})
```

The diagram illustrates the data flow between the Content script and the Background page. It starts with the Content script code (lines 1-6). An orange arrow points from the 'content' variable in line 5 to the 'content' parameter in the 'sendMessage' call. Another orange arrow points from the 'tabId' variable in line 5 to the 'tabId' parameter in the 'sendMessage' call. From the 'sendMessage' call, an orange arrow labeled 'message' points to the 'onMessage' listener in the Background page code (line 8).

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 chrome.runtime.onMessage.addListener(function(request) {
9   chrome.tabs.executeScript(request.tabId, {code: request.content});
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 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"}}
```

Suspicious Data Flow Tracking

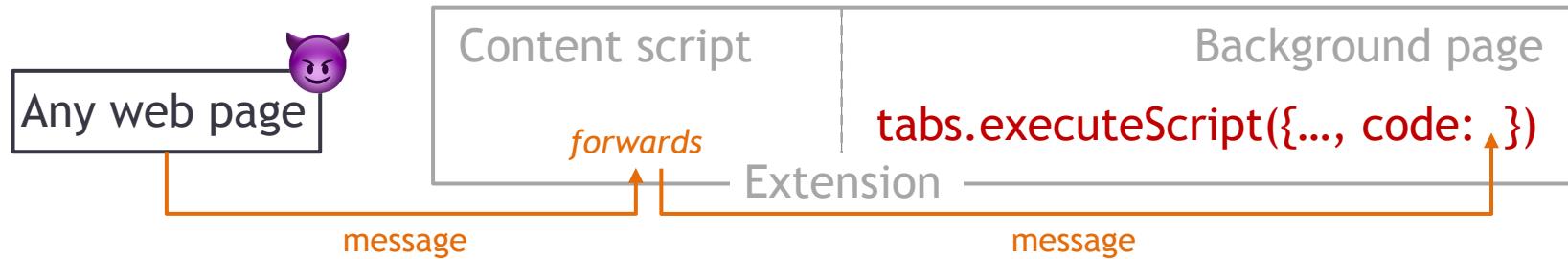
```
1 // Content script code
2 window.addEventListener("message", function(event) {
3   content = event.data.content;
4   tabId = event.data.tabId;
5   chrome.runtime.sendMessage({content: content, tabId: tabId});
6 })
```

```
7 // Background page code
8 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"}}
```

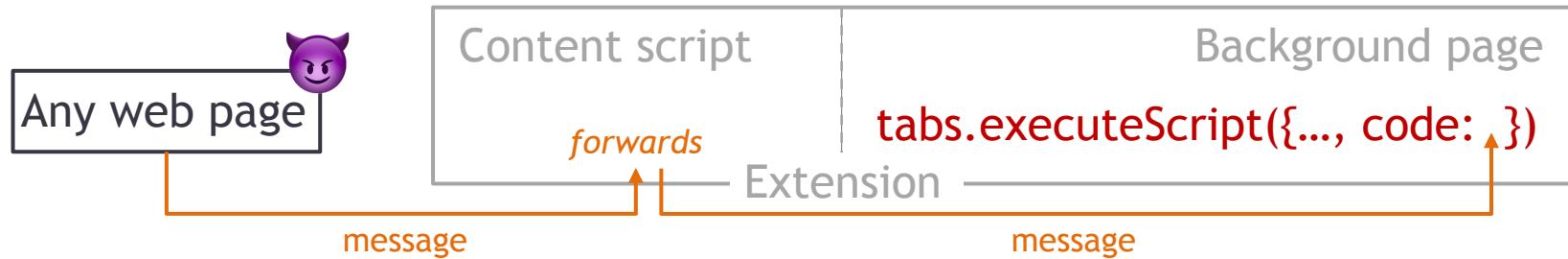
Case Studies of Vulnerable Chrome Extensions

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

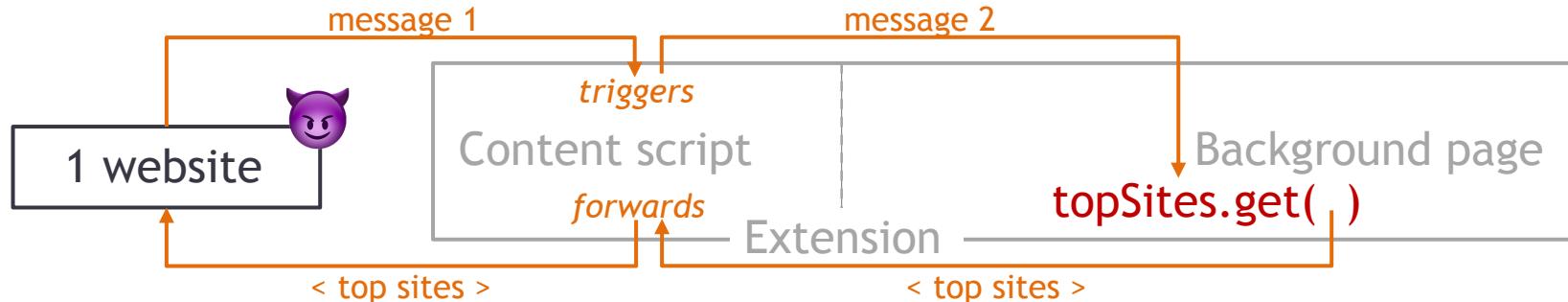


Case Studies of Vulnerable Chrome Extensions

- 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!)



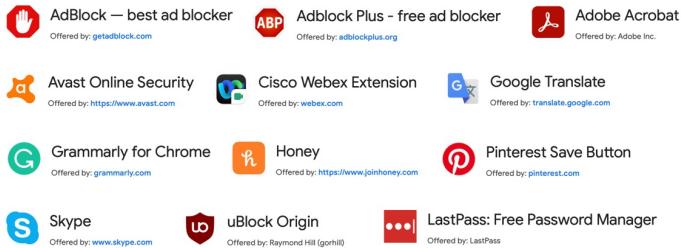
 Aurore54F/DoubleX

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

Browser Extensions are Popular



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

What is in the Chrome Web Store?

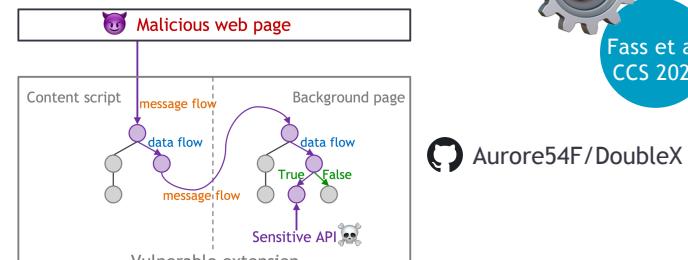


- 350M users installed SNE in the last 3 years
- These SNE stay in the Chrome Web Store *for years*
- Extensions have a **short life cycle** in the CWS (60% stay 1 year)
- Critical **lack of maintenance** in the CWS (60% received no update)

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

Detecting Vulnerable Extensions with DOUBLEX



- DOUBLEX detects suspicious data flows in browser extensions
- 184 vulnerable extensions | Precision: 89% | Recall: 93%



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