



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

Aurore Fass ^{S,C}, Dolière Francis Somé ^{S,C}, Michael Backes ^C, and Ben Stock ^C

^S Stanford University

^C CISPA Helmholtz Center for Information Security

ACM CCS 2021

Browser Extensions...

are popular 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

Browser Extensions...

are popular 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: www.cisco.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

BUT

may introduce security and privacy threats

e.g.,



- execute arbitrary code in *any* websites, even without a vulnerability in the websites themselves

- exfiltrate sensitive user data to *any* websites

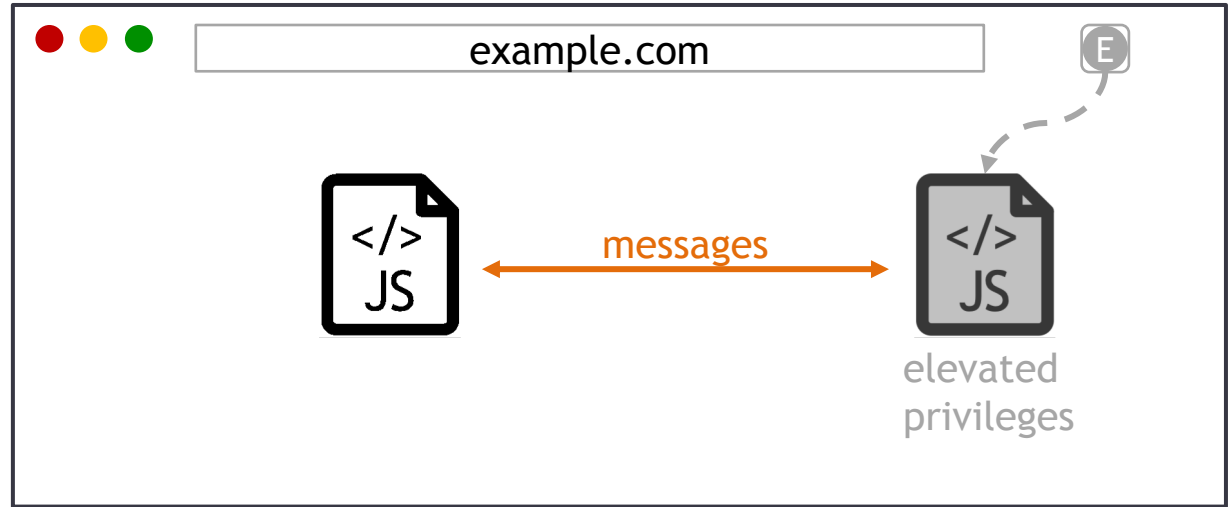
Browser Extensions are Highly Privileged

- Have access to privileged APIs and features
 - e.g., an ad-blocker can read/write web page content
- Can do tasks that web applications cannot traditionally do
 - e.g., are not subject to the SOP and can access arbitrary cross-domain data (even when a user is logged in)

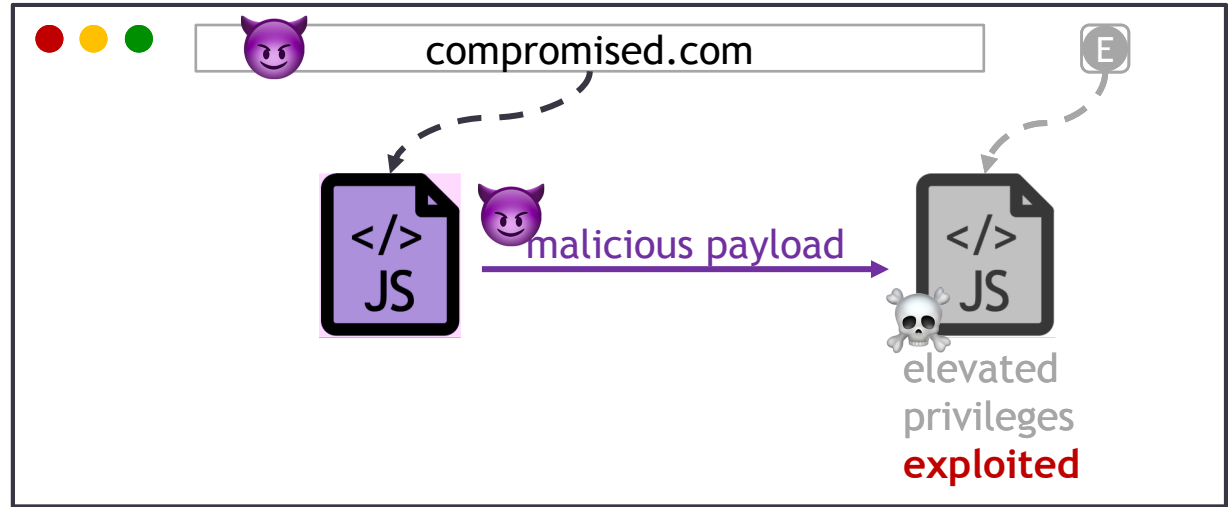
Browser Extensions are Highly Privileged

- Have access to privileged APIs and features
 - e.g., an ad-blocker can read/write web page content
- Can do tasks that web applications cannot traditionally do
 - e.g., are not subject to the SOP and can access arbitrary cross-domain data (even when a user is logged in)
- Attract the interest of attackers
 - *Malicious* extensions:  Chrome vetting system
 - *Vulnerable* extensions: 

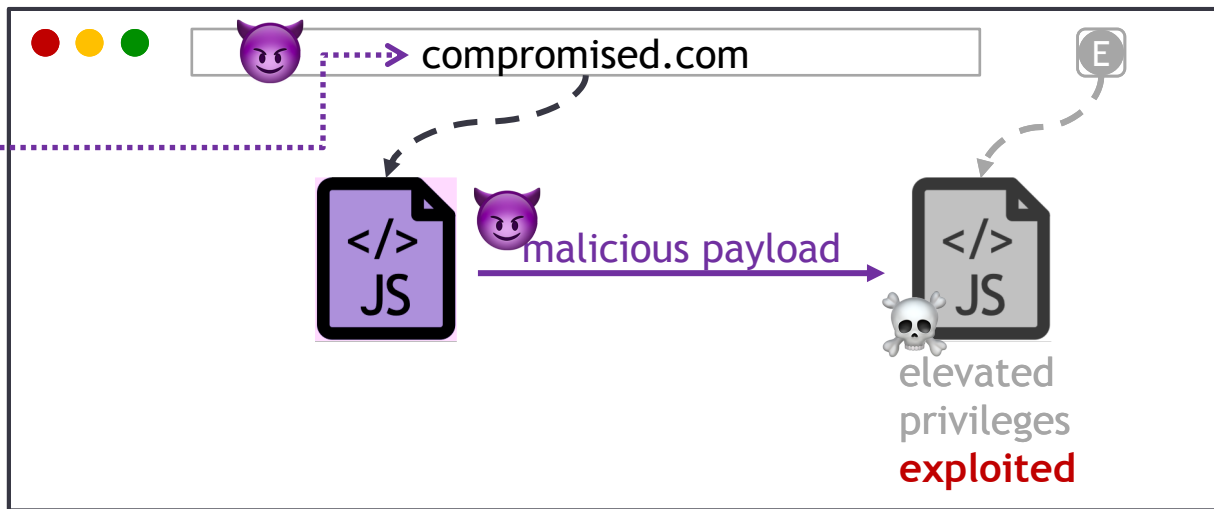
Exploiting Vulnerable Extensions



Exploiting Vulnerable Extensions



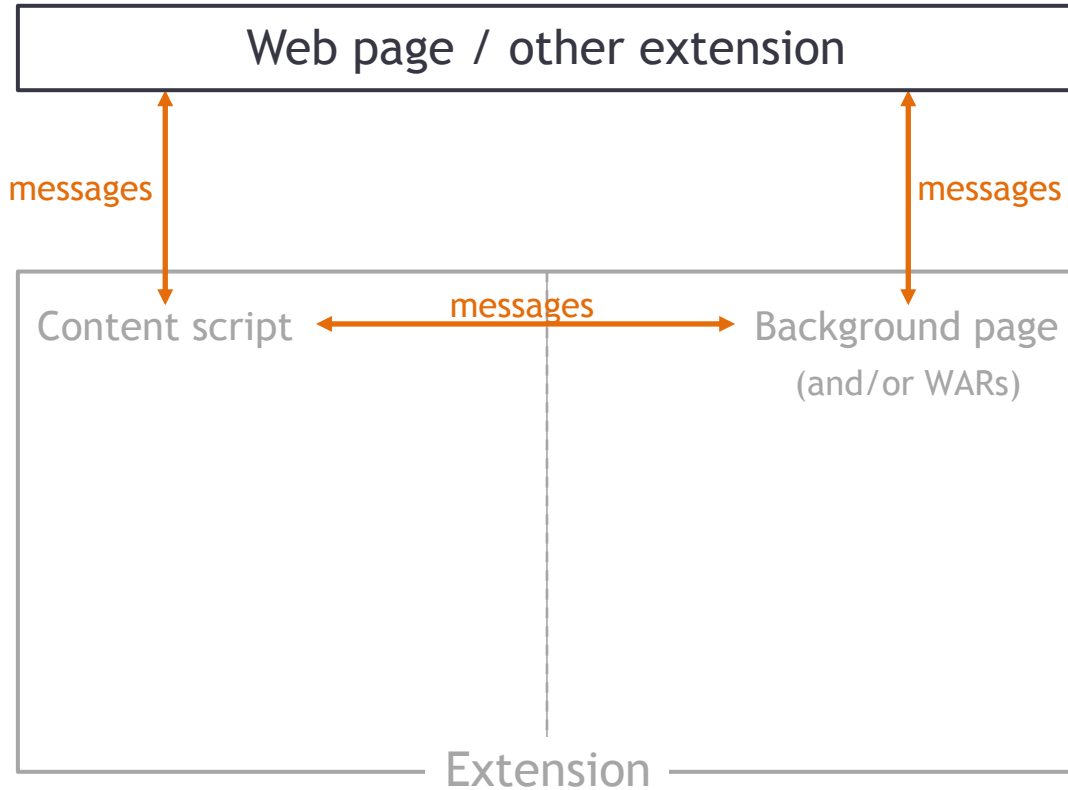
Exploiting Vulnerable Extensions



- 🦹‍♂️ Code Execution
- 🦹‍♂️ Triggering Downloads
- 🦹‍♂️ Cross-origin Requests
- 🦹‍♂️ Data Exfiltration

➤ RQ: Can we statically analyze browser extensions to detect suspicious external data flows?

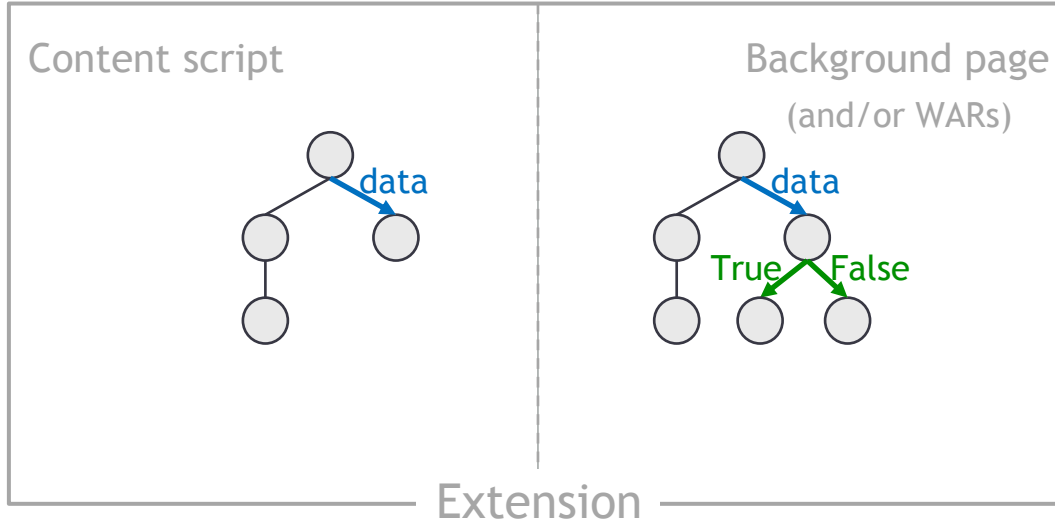
Extension Architecture and Communication



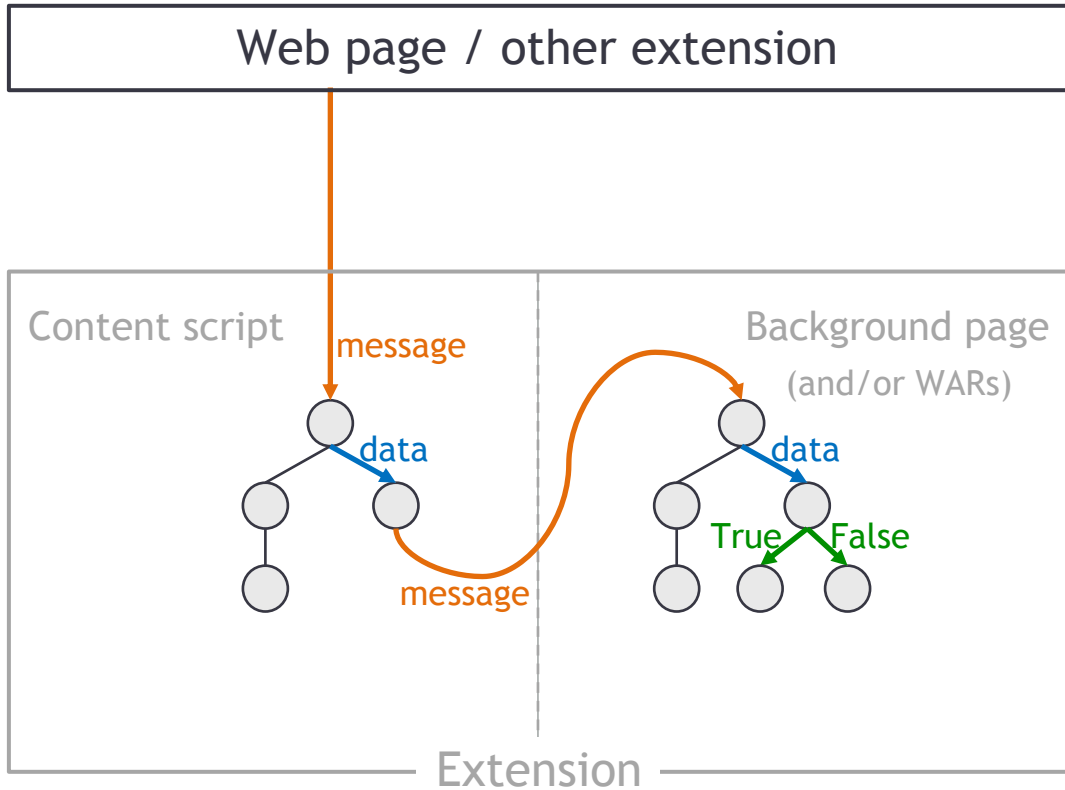
DOUBLEX: Suspicious Data Flow Detection

Web page / other extension

Per-component JS code abstraction



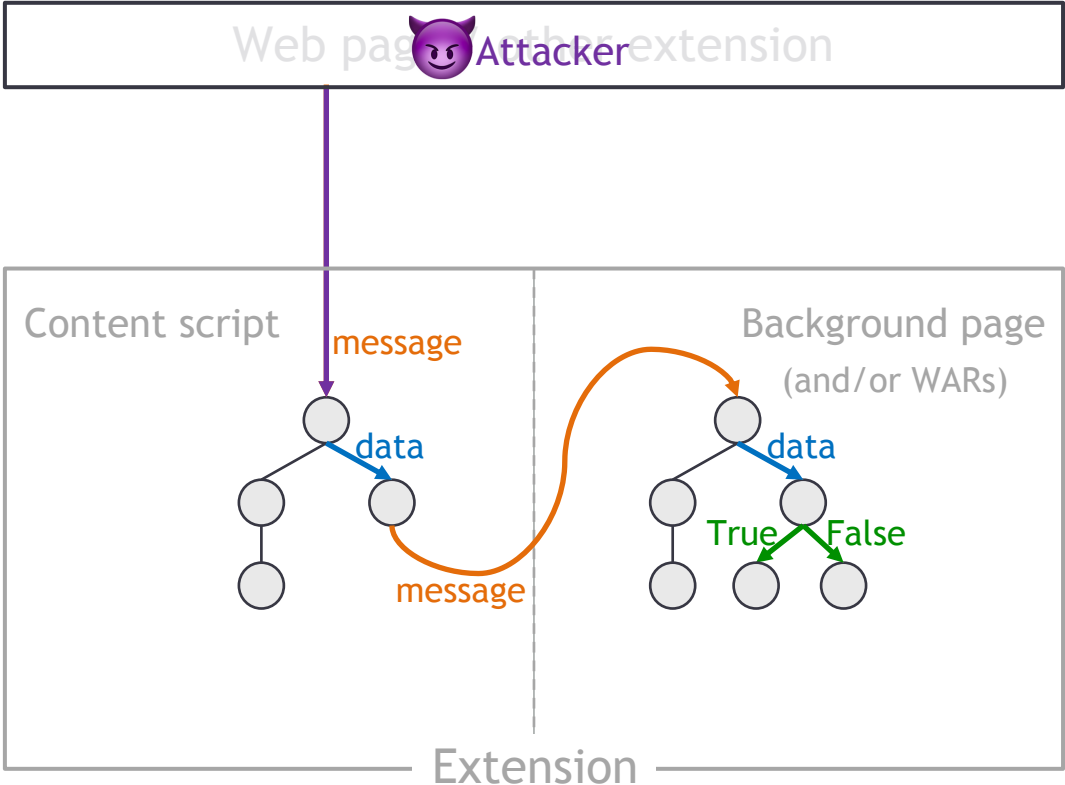
DOUBLEX: Suspicious Data Flow Detection



Per-component JS code abstraction

Extension Dependence Graph (EDG)

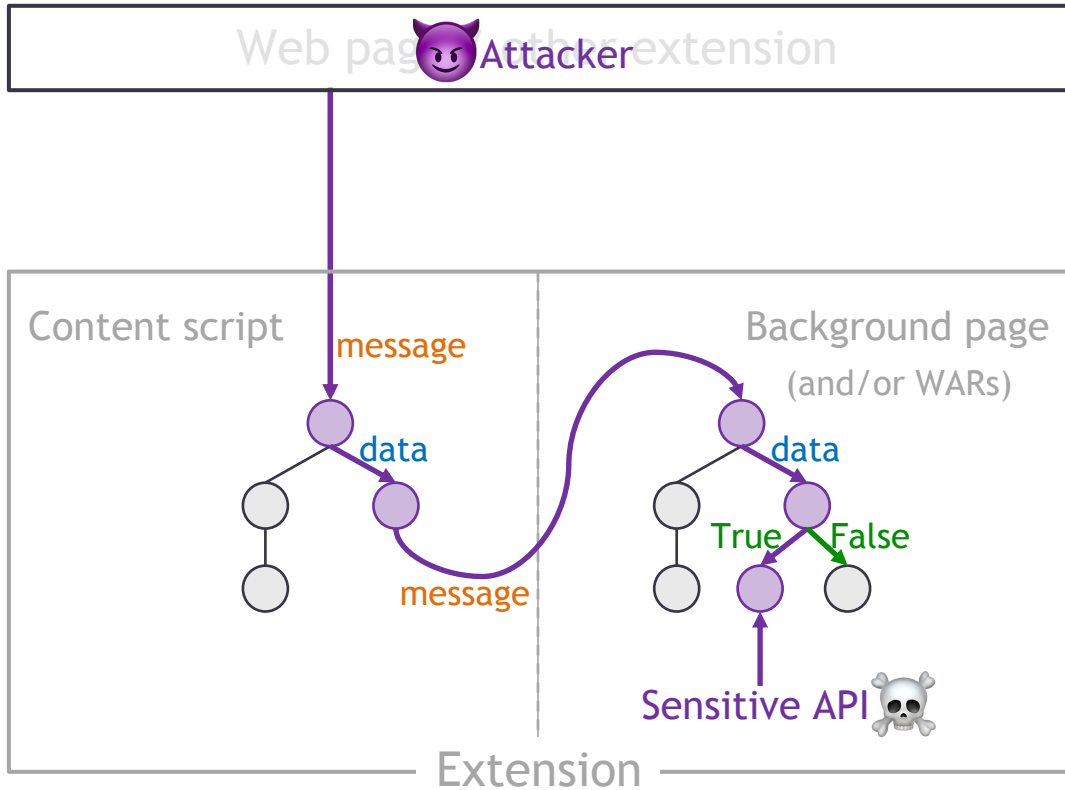
DOUBLEX: Suspicious Data Flow Detection



Per-component JS code abstraction

Extension Dependence Graph (EDG)

DOUBLEX: Suspicious Data Flow Detection



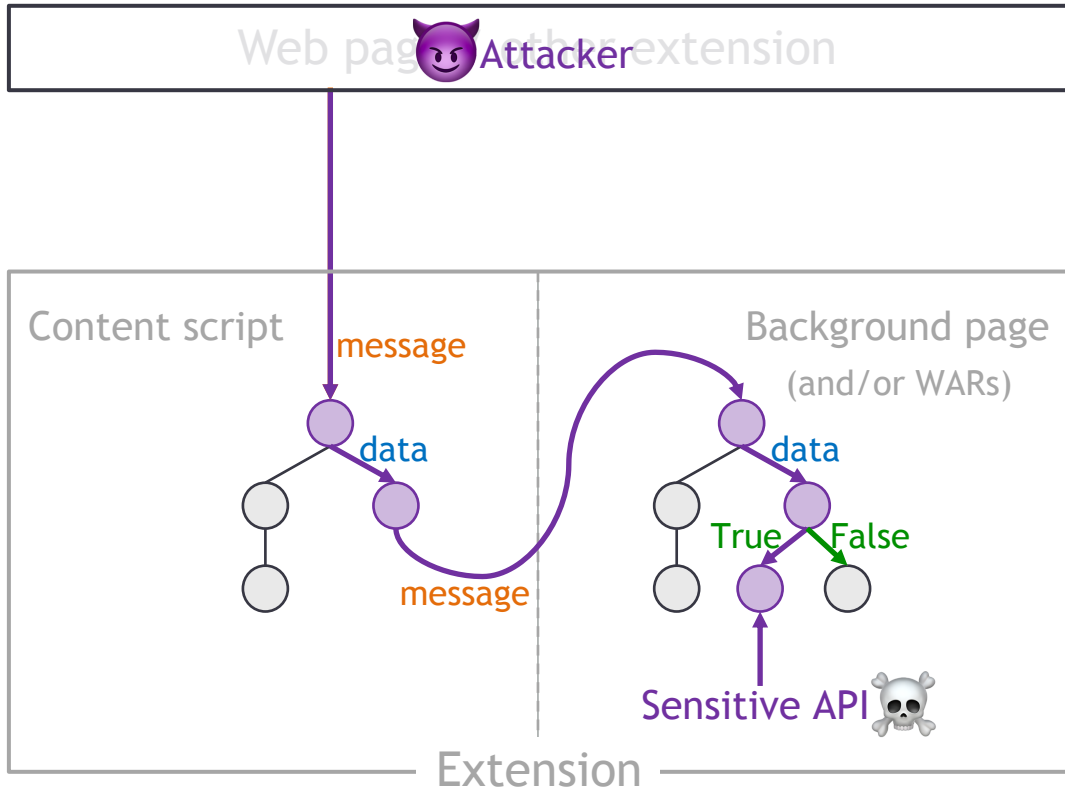
Per-component JS code abstraction

Extension Dependence Graph (EDG)

Suspicious data flow tracking

↓
Data flow report

DOUBLEX: Suspicious Data Flow Detection



Per-component JS code abstraction

Extension Dependence Graph (EDG)

Suspicious data flow tracking

Data flow report

Abstract code representation



AST (Abstract Syntax Tree)

– conditions



control flow

– variable dependencies



data flow

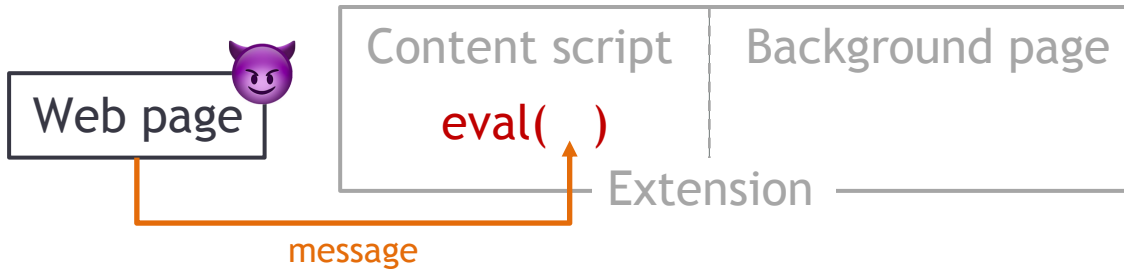
– variable values



pointer analysis

Per-Component JS Code Abstraction

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



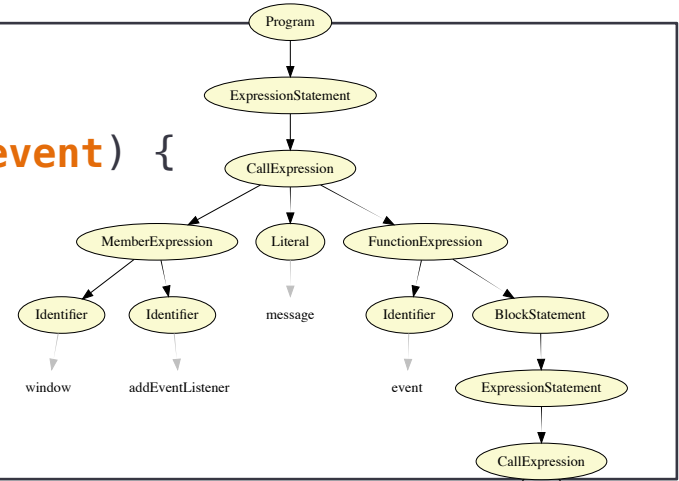
Per-Component JS Code Abstraction

```
// Content script code
```

```
window.addEventListener("message", function(event) {
```

```
    eval(event.data);
```

```
});
```



Abstract code representation

– conditions

– variable dependencies

– variable values



AST



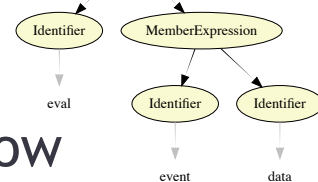
control flow



data flow



pointer analysis



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



Abstract code representation



✓ AST

– conditions



control flow

– variable dependencies



✓ data flow

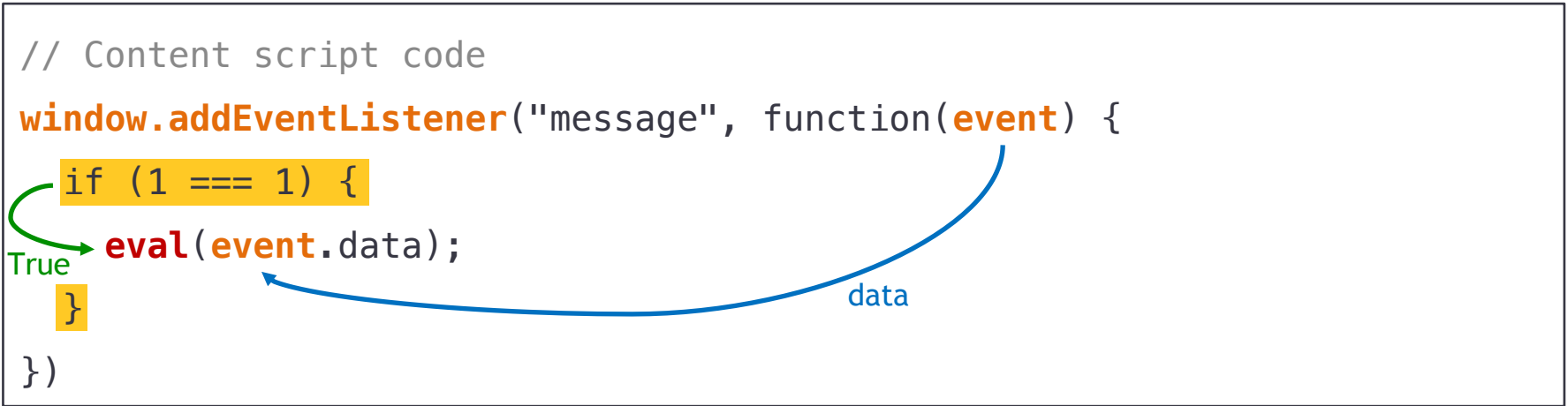
– variable values



pointer analysis

Per-Component JS Code Abstraction

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



The diagram illustrates data flow in the provided JavaScript code. A blue arrow labeled 'data' originates from the 'event.data' property access in the function call 'eval(event.data);' and points to the 'eval' function. A green arrow labeled 'True' originates from the 'if (1 === 1) {' condition and points to the 'eval' function, indicating that the condition is always true.

Abstract code representation



 AST

– conditions



 control flow

– variable dependencies



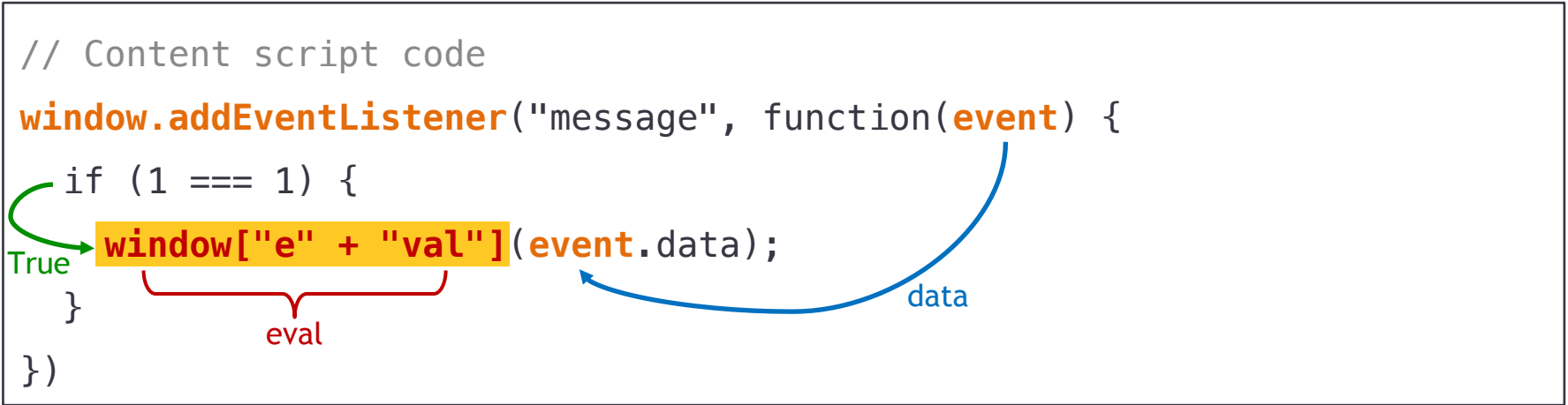
 data flow

– variable values



pointer analysis

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



Abstract code representation

– conditions

– variable dependencies

– variable values



✓ AST



✓ control flow

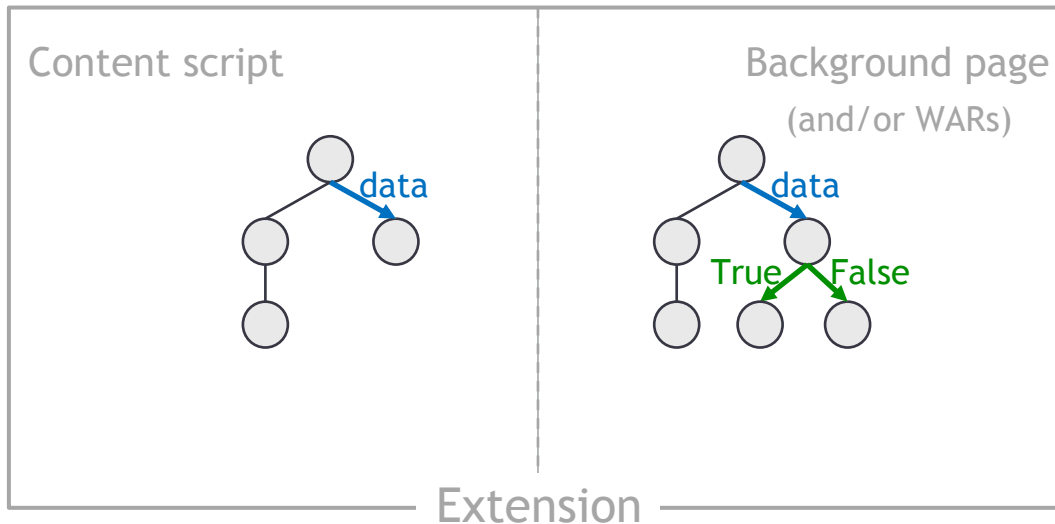


✓ data flow



✓ pointer analysis

DOUBLEX: Suspicious Data Flow Detection



Per-component JS code abstraction

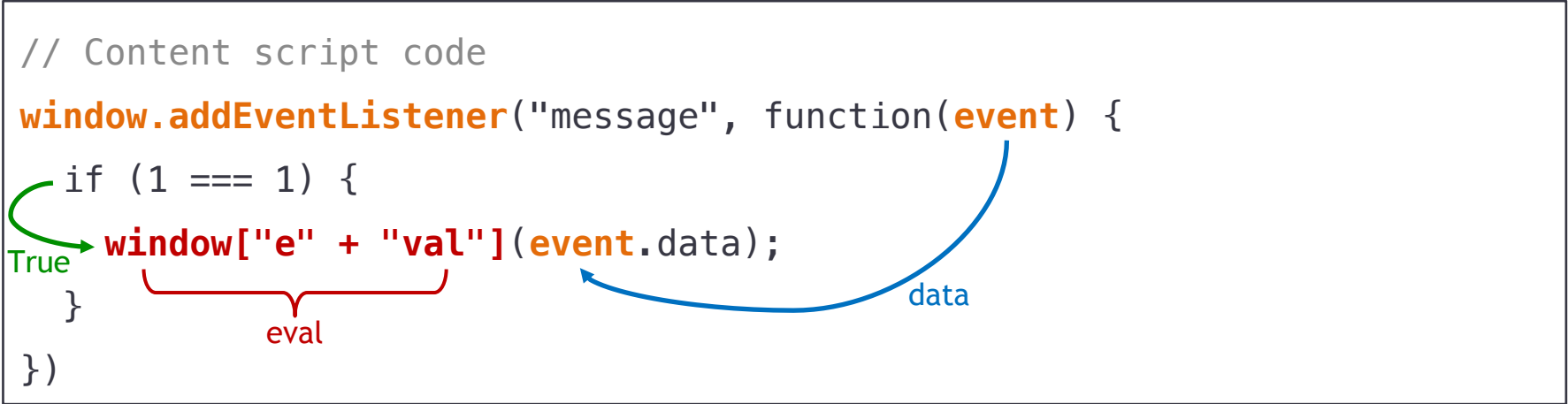
Extension Dependence Graph (EDG)

Suspicious data flow tracking

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

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

Diagram illustrating the extension dependence graph for the provided code snippet:

- A green arrow labeled "True" points from the `if (1 === 1)` condition to the `eval` expression `["e" + "val"]`.
- A blue arrow labeled "data" points from the `event` parameter to the `event.data` property access.
- A purple devil emoji is placed above the `event` parameter.

– 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});
```

message

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

– external messages

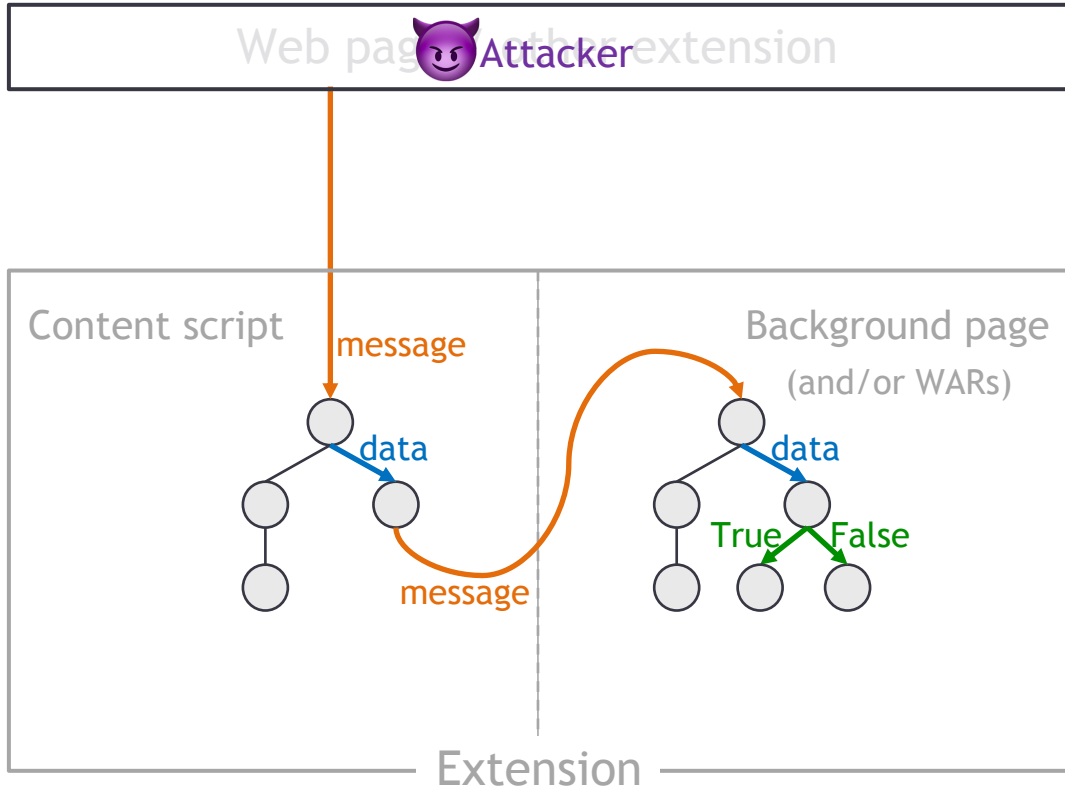


– internal messages



➤ Models message interaction within and outside of an extension

DOUBLEX: Suspicious Data Flow Detection



Per-component JS code abstraction

Extension Dependence Graph (EDG)

Suspicious data flow tracking

↓
Data flow report

Suspicious Data Flow Tracking

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

The diagram shows the code with several annotations: a purple devil emoji is placed above the `event` parameter in the function signature. A blue arrow labeled `data` points from the `event` parameter to the `event.data` property access. A red bracket labeled `eval` spans the `event.data` property access and the `eval` function call. A green arrow labeled `True` points to the `if (1 === 1)` condition.



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

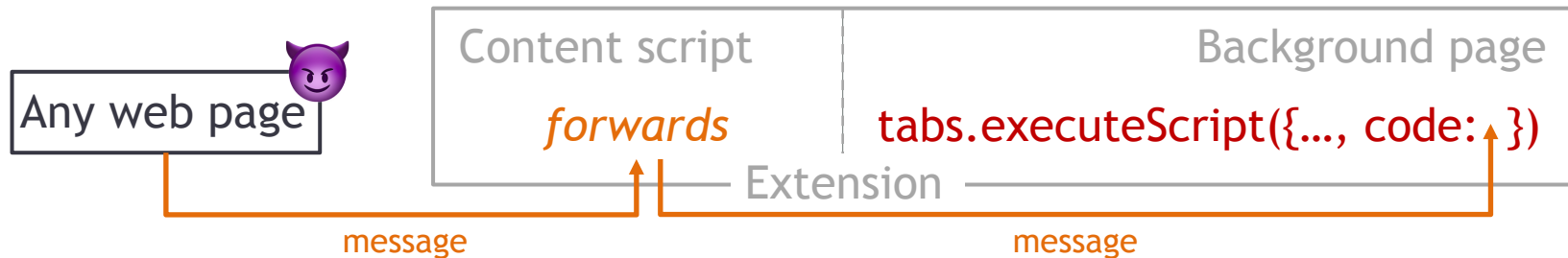
Large-Scale Analysis of Chrome Extensions

- Analyzed 155k Chrome extensions from 2021 with DOUBLEX
 - 278 suspicious extensions reported (309 suspicious data flows)
 - manual review
 - **precision: 89%** verified dangerous data flows (275 / 309)

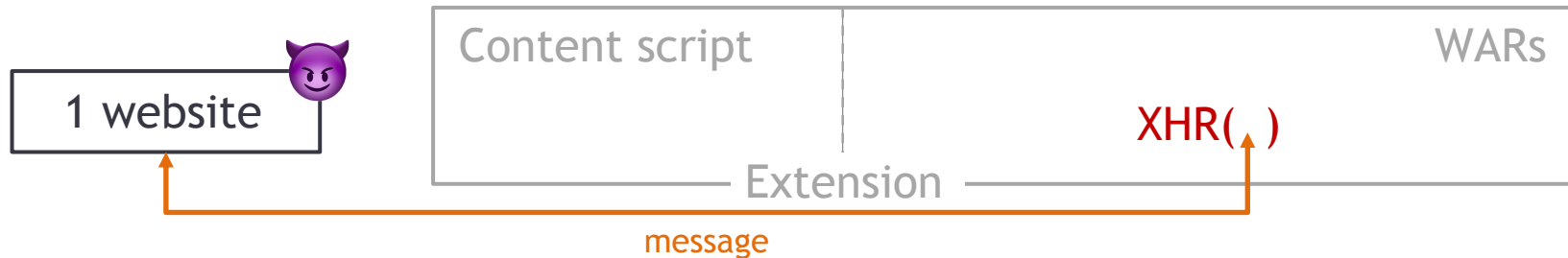
Attacker capabilities	#Reports	#Verified data flow	#Exploitable
Code Execution	113	102	63
Triggering Downloads	21	21	21
Cross-Origin Requests	95	75	49
Data Exfiltration	80	77	76
Sum	309	275	209

Case Studies of Vulnerable Chrome Extensions

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



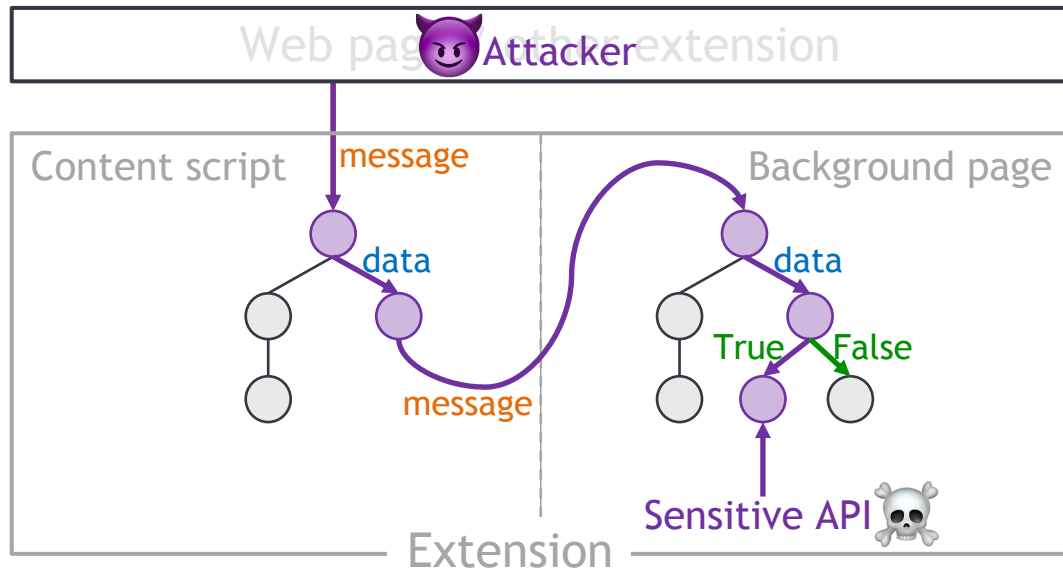
- Cross-origin requests (*koh...*, 200k+ users)



- Analyzed 155k Chrome extensions from 2021 with DOUBLEX
 - 278 suspicious extensions reported
 - manual review
 - **precision: 89%** verified dangerous data flows
 - **184 confirmed vulnerable extensions**
 - 36% can be exploited by *any* websites or extensions
 - 2.4 - 2.9 million users impacted
- Analyzed known vulnerable extensions* with DOUBLEX
 - **recall: 93%** of known vulnerabilities are detected (151 / 163)

Life Cycle of Vulnerable Chrome Extensions

- Analyzed 165k extensions from 2020 with DOUBLEX
 - 193 vulnerable extensions (184 in 2021)
 - vulnerability disclosure for 35 extensions (48 extensions when including 2021)
- Comparison of vulnerable extensions in 2020 vs. 2021
 - not in the Store anymore: 30 / 193
 - vulnerability fixed: 3 / 193
 - turned vulnerable: 5 / 184
 - new vulnerable: 19 / 184
 - **still vulnerable: 160 (87%!) ➤ Need to prevent vulnerable extensions from entering the Store → DOUBLEX**



DOUBLEX: detects vulnerable data flows in extensions

- Per-component code abstraction
- Extension Dependence Graph
- Suspicious data flow tracking

Thank you

Analyzed 155k Chrome extensions in 2021

- **184 vulnerable extensions**; 160 already vulnerable in 2020
- **precision: 89%** verified dangerous data flows
- **recall: 93%** of known vulnerabilities are detected



Aurore54F/DoubleX



@AuroreFass