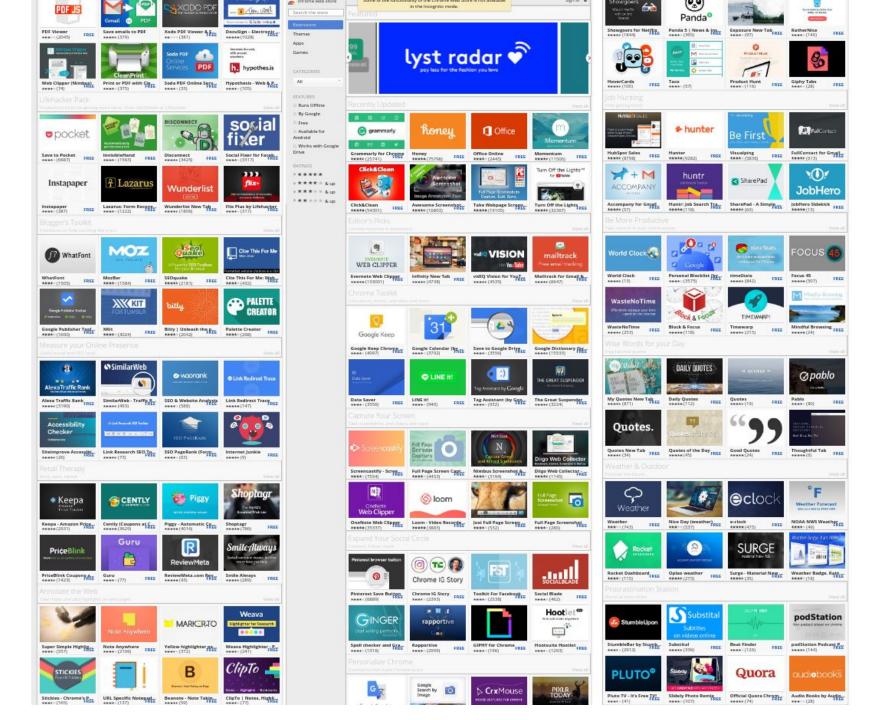
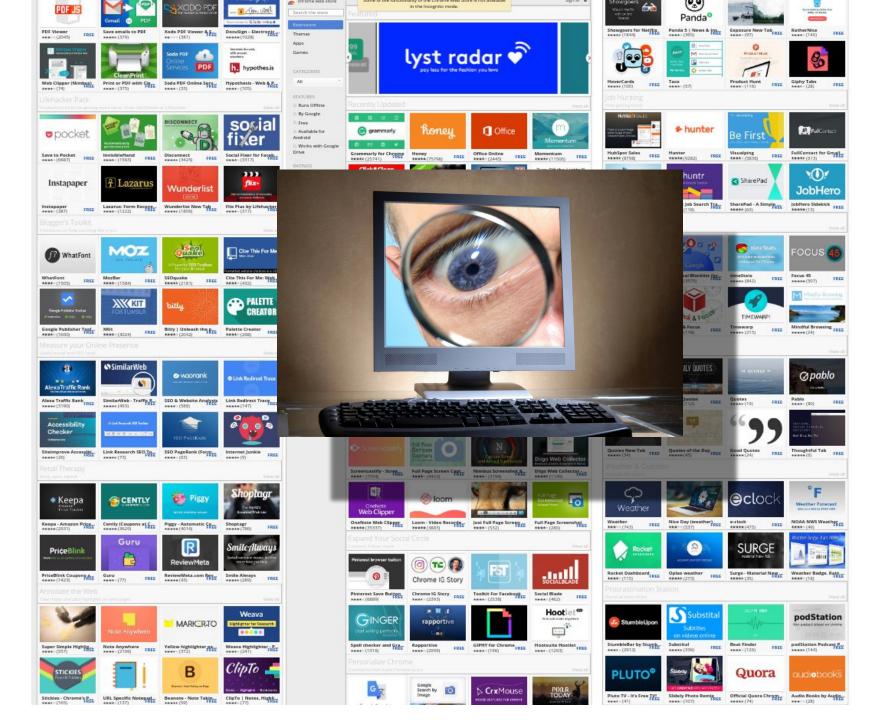


#### **Ex-Ray: Detection of History-Leaking Browser Extensions**

Michael Weissbacher, Enrico Mariconti, Guillermo Suarez-Tangil, Gianluca Stringhini, William Robertson, Engin Kirda

Northeastern University, University College London





#### **Overview**

- Extension Basics
- Extension Privacy Risks
- HoneyPot Probe
- Detection Methodology
- System Design and Evaluation
- Conclusion and Discussion

#### **How extensions work**

- Additions to browser core functionality
- Powerful API based on permissions
  - Modification of active pages
  - Modification of requests / responses
  - Often access to all visited pages
  - Access to cookies
  - Access to previous history

### **Extension Privacy Risks**

- Privacy leaks through
  - Modifications to the site: referrer
  - Request or response interception
  - Polling active tab
  - 0 ...
- No unified way of detection
- Previews work:
  - Manual analysis
  - Leaking keywords, search traffic

### **Extension Privacy Mitigations**

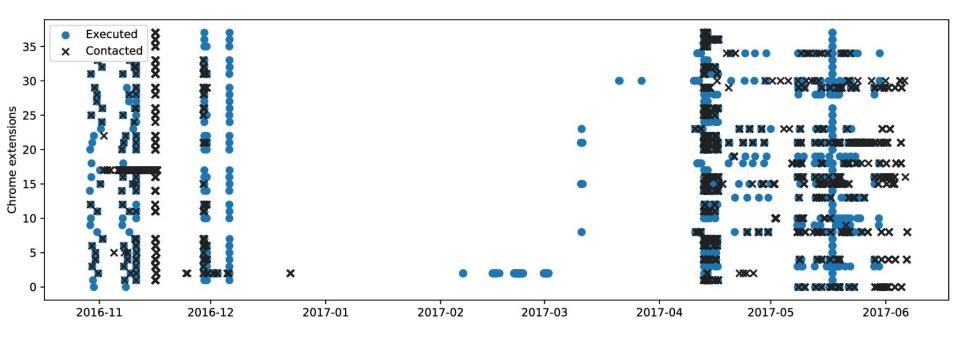
- Permissions can restrict access to sites
- Extensions often over-request access
- Only modest permissions required to leak history

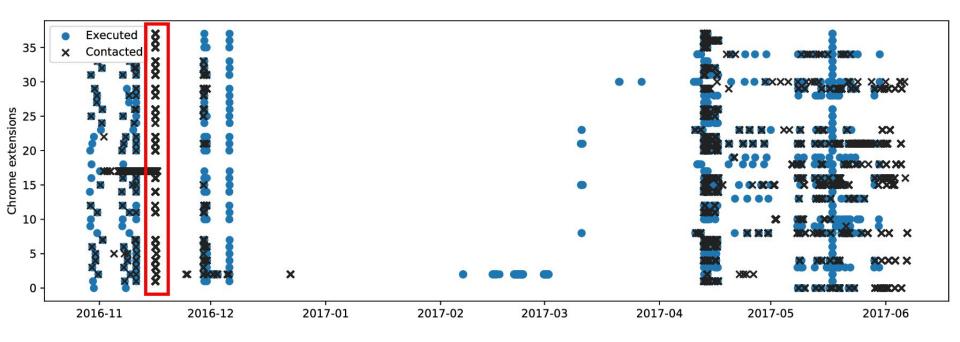
## **Tracker Comparison**

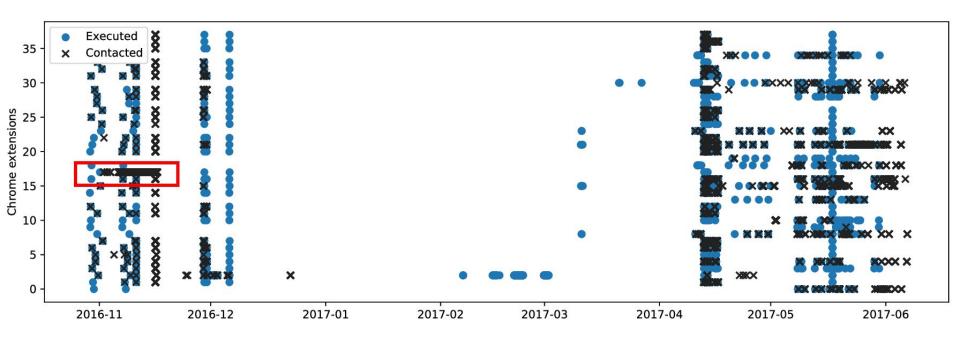
- On Websites:
  - Opt-in: Website owner
  - Opt-out: Ghostery
- In Extensions:
  - (typically) all websites
  - Implicit Opt-in through installation
  - No opt-out

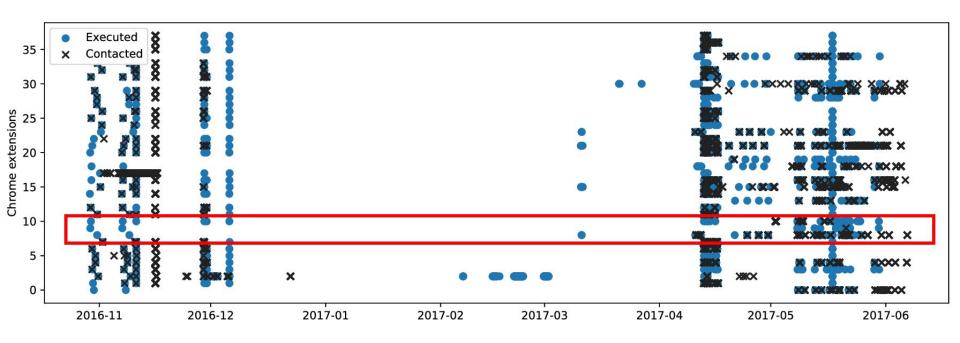
"Is this an issue in practice?"

- Extensions run in a container
- Web and DNS local
- URLs unique to extension
- Public Internet only for other resources
- Browsing our website...
- ... which is also available on the public Internet









# **HoneyPot Incoming Connections**

Extension Name	Installations	Connection Origin
Stylish - Custom themes	1,671,326	*.bb.netbynet.ru *.moscow.rt.ru *.spb.ertelecom.ru
Pop Up Blocker for Chrome	1,151,178	*.aws.kontera.com 176.15.177.229 *.bb.netbynet.ru
Desprotetor de Links	251,016	*.aws.kontera.com *.moscow.rt.ru *.bb.netbynet.ru

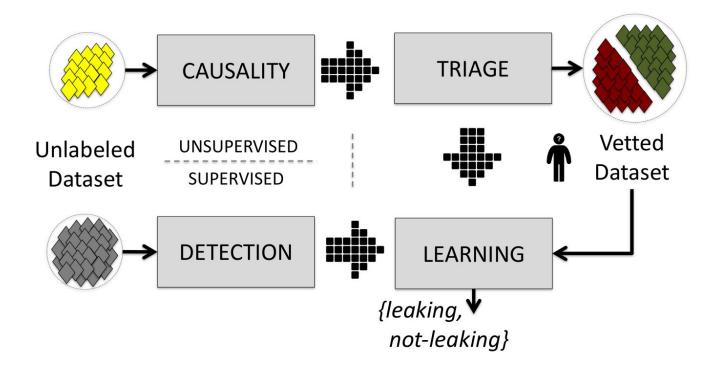
- Connections prove use of data
- Excluding VPN: 38 Extensions
- Connection often immediately after execution
- They leak immediately
- Indications for collaboration, no proof
- No malicious activity detected
- Motivation for automated detection system

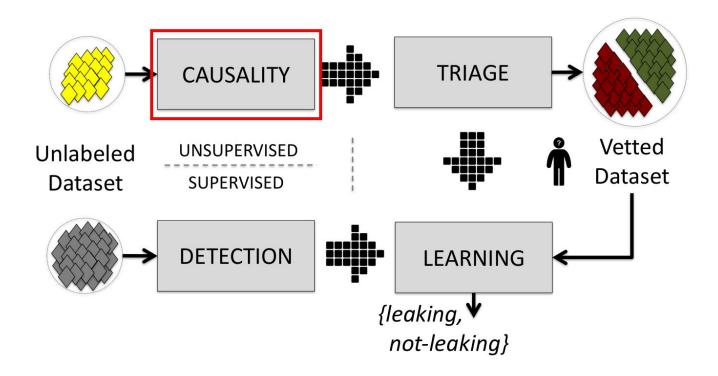
**Hypothesis**: For tracking samples, sent data size should grow in relation to history provided to the extension.

### **Ex-Ray Goals**

- Robust Detection
  - Traffic obfuscation / encryption
  - Method of data collection / exfiltration
- Automated detection of leaks
- Large scale

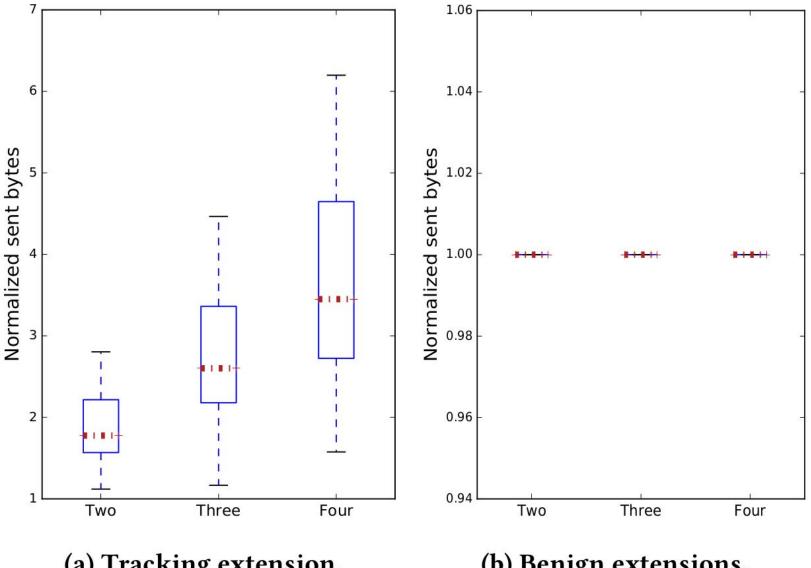
- Controlled environment
- Execution in multiple stages
- Vary size of browsing history
- Supervised and unsupervised methods





## **Causality**

- Varying history as variable over stages
  - Stage 0: example.com/example/index.html
  - Stage 1: example.com/example/<500characters>/index.html
  - 0 ...
- Expectations
  - Benign: no change
  - Otherwise: ?



(a) Tracking extension.

(b) Benign extensions.

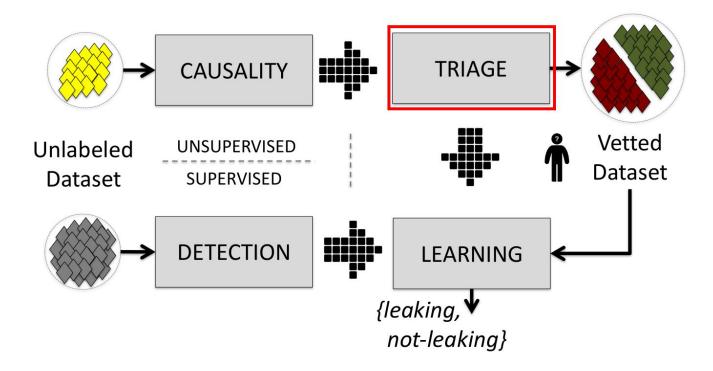
## **Causality**

- Counterfactual analysis
- Input variable: size of history
- Output variable: <data sent, destination> tuples
- Invariants of trackers
- Goal: find deterministic tracking
- Supervised method: trained on benign and leaking datasets

## **Causality**

#### Three steps

- 1. Minimum intercept: threshold
- 2. Minimum slope: increase
- 3. Level of confidence: proximity to model



- Quantify leakage
- Prioritize extensions
- Supports human analyst in prioritizing extensions

- L: number of leaked URLs between experiments
- |si|, |sj|: number of bytes sent to domain
- τ: expected threshold for increase

$$L(s_i, s_j) = \frac{|s_j| - |s_i|}{\tau}$$

- Score: Likelihood of a leak
- s: transition between stages (i=>j)

$$score(x) = \prod_{s} e^{L(s)}$$

$$\operatorname{leak}(x) = \begin{cases} \operatorname{not-leaking} & \text{if } score(x) \leq 1 \\ \operatorname{possibly-leaking} & \text{if } 1 < score(x) \leq 100 \\ \operatorname{likely-leaking} & \text{otherwise.} \end{cases}$$

Result: indicators for manual analysis

## **Triage Samples: Leak**

```
QR Code Generator

4e+18 connectionstrenth.com

394.88 a.pnamic.com

28.22 eluxer.net

4.48 rules.similardeals.net

1.16 code.jquery.com
```

## **Triage Samples: Benign**

```
Bible Quote of the Day

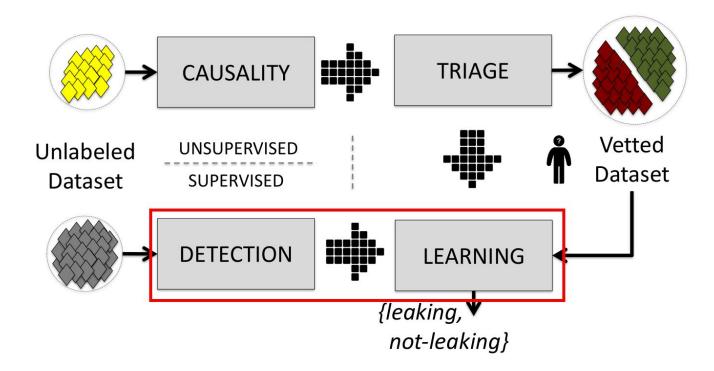
1.00 www.gstatic.com

1.00 chromium-i18n.appspot.com

1.00 ssl.gstatic.com

1.00 localhost

0.67 www.google.com
```



#### **Behavioral Detection**

- Based on previously flagged extensions
- Clang Libtooling instrumentation
  - C++ source code rewriting
  - 11,132 function trace points
- API call analysis
- Input: n-grams of API calls

## **Evaluation: Causality**

- Crawled store for extensions > 1,000 installations
- 10,691 Extensions total
- 212 flagged: 1.9%
- 184 manually verified as leaking
- 28 wrongly identified
- False Detection Rate: 0.27%
  - Flagged due to ads
  - Possible improvement: increase # stages

#### **Evaluation: Behavioral**

- Best parameters:
  - o n-gram: 2
  - F1 score: 96.43%
- Distinguishing calls:
  - URL manipulation
  - JavaScript manipulation
- Most distinguishing sequence:
  - extensions.browser.extension\_prefs.GetExtensionPref()
  - chrome.browser.extensions.shared\_user\_script\_master.GetScriptsMetadata()

## **Noteworthy Samples: Causality**

- Not detectable by state-of-the-art leakage detection systems
- Previously unknown leakage channels
  - Strong Encryption
  - Unsupported Protocol

## Web of Trust (WOT)

- Provides crowd-sourced "trust" ranking
- 1.2M installations
- Extension received media coverage for selling user data
- RC4 encryption (See crypto.js file)
- Can be implemented similarly to Google Safe Browsing (offline)

### CouponMate

- WebSockets: Protocol not supported by previous systems
- Protocol growing in popularity: 0.96%

▼ STORE DATA 116 {host: "www.example.com", url: "http://www.example.com/example",...}

```
STORE_DATA 116 {"host":"www.example.com","url":"http://www.example.com/","checksum":["972558cbec8cf1419c39a979af8ede252eee4c54
STORE_DATA 116 972558cbec8cf1419c39a979af8ede252eee4c54
STORE_DATA 116 {"host":"www.example.com","url":"http://www.example.com/example","checksum":["972558cbec8cf1419c39a979af8ede252eee4c54
STORE_DATA 116 972558cbec8cf1419c39a979af8ede252eee4c54
PING
PONG
```

url: "http://www.example.com/example"

host: "www.example.com"

▼ checksum: ["972558cbec8cf1419c39a979af8ede252eee4c54"]
0: "972558cbec8cf1419c39a979af8ede252eee4c54"

#### Possible remediations

- Stores should analyze extensions to warn users
- Implement API to inspect background traffic
- Invasive tracking as single purpose rule violation

#### **Conclusions**

- Robust detection method for privacy leaks
- Prototype: Ex-Ray
- Supervised and Unsupervised methods
- 10,691 extensions analyzed
- 212 flagged
- Found two novel leaking channels in use

### Thank you for your attention

**Questions?** 

**Paper and Data:** 



https://goo.gl/nezKGp