# FPFlow: Detect and Prevent Browser Fingerprinting with Dynamic Taint Analysis

*Tianyi Li[1], Xiaofeng Zheng[2], Kaiwen Shen[2], Xinhui Han[1]*
*[1]Peking University, [2]Tsinghua University*

## INTRODUCTION

### Browser Fingerprinting
➤ Collecting a set of browser attributes to uniquely identify the web user.
➤ The diversity of many browser properties has been exploited.
➤ Hardware feature extracted with rendering functions (e.g., toDataURL) can track user across different browsers.

### Prevention of Browser Fingerprinting
➤ Many protection mechanisms has been proposed.
✘ Can only detect or prevent a certain kind of browser fingerprinting.
✘ Not all of them actually "protect" the web user.
✘ Some protection methods makes the browser easier to be fingerprinted.
✘ Many methods prevent fingerprinting by sacrificing client functionalities (e.g., disabling Canvas API or Fixing the window size).

### Contribution
➤ We proposed an information-flow based method to detect and prevent browser fingerprinting.
➤ We implemented FPFlow based on Chromium as the prototype of our method.
➤ We conducted a large-scale experiment on FPFlow on Alexa top 10,000 sites and found 71.3% site potentially performing browser fingerprinting.
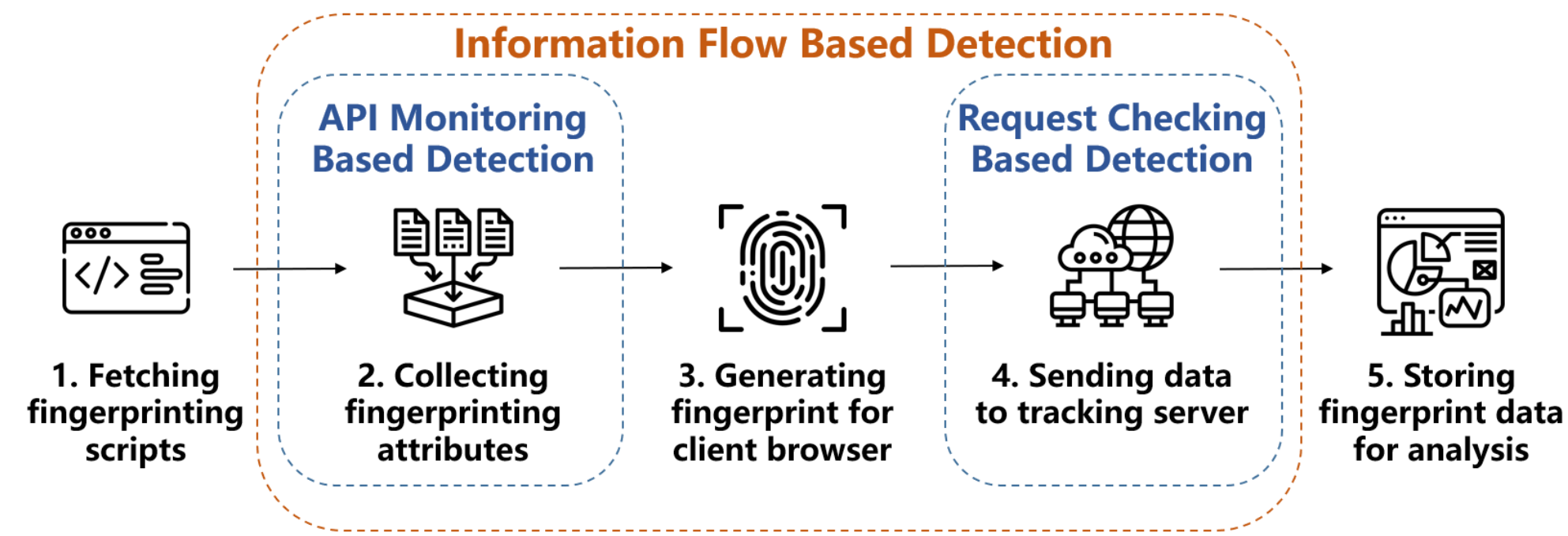
## MOTIVATION



**Information Flow Based Detection**

**API Monitoring Based Detection** → **Request Checking Based Detection**

1. Fetching fingerprinting scripts
2. Collecting fingerprinting attributes
3. Generating fingerprint for client browser
4. Sending data to tracking server
5. Storing fingerprint data for analysis

**Figure 1: The Process of Browser Fingerprinting**

### API Monitoring Based Detection
➤ Logging access to certain DOM APIs
✓ Canvas & AudioContext fingerprinting
✘ Property-based fingerprinting
✘ Doesn't know if the data is sent out

### Request Checking Based Detection
➤ Searching for fingerprinting attributes in web requests.
✓ Property-based fingerprinting
✘ Canvas & AudioContext fingerprinting
✘ Fingerprint encoding or hashing

### Information Flow Based Detection
➤ Taint Source: Fingerprinting Attributes
➤ Tant Sink: Web Request
➤ Runtime Propagation: Decide whether the fingerprinting attributes are sent out
✓ Canvas & AudioContext fingerprinting
✓ Property-based fingerprinting
✓ Confirm that fingerprinting attributes are sent out
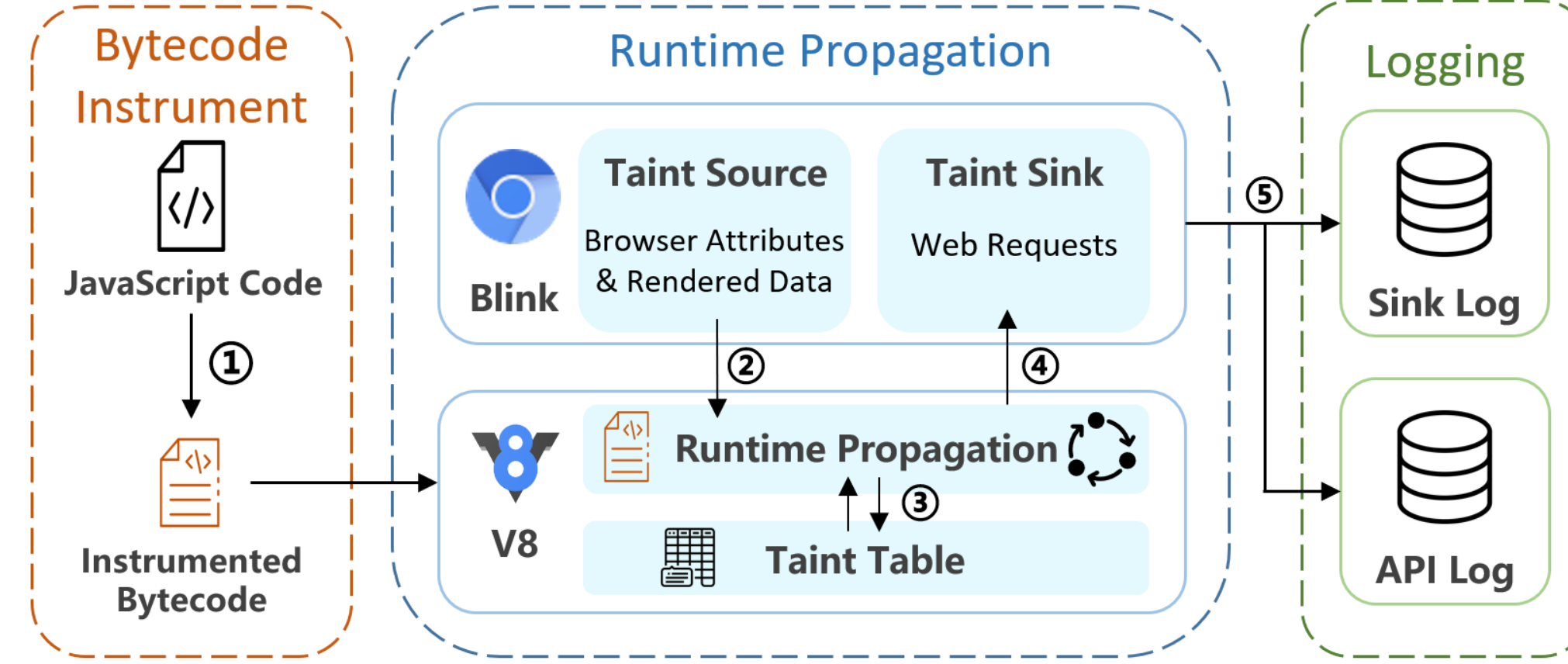✓ Can handle fingerprint encoding and hashing

## SYSTEM DESIGN



**Figure 2: Abstract Architecture of FPFlow**

① **Bytecode Instrument**
- When loading a script, V8 parses the source code to AST then generates bytecode.
- FPFlow adds additional bytecodes for taint propagation at bytecode generation phase.
- FPFlow propagates taint at property load, basic operations, and Native function calls.

② **Get Tainted**
- When JavaScript accesses to fingerprinting attributes (DOM properties or functions), the JavaScript object is marked as tainted.

③ **Taint Propagation**
- FPFlow maintains a taint table to store object taints.
- During execution, FPFlow propagates taint and update the taint table.

④ **Taint Sink**
- When the web page initiates a web request, FPFlow checks whether it contains taint data.
- If so, FPFlow marks the request as fingerprinting request, and can stop it to prevent fingerprinting.

⑤ **Logging**
- Sink log contains the target URL, method, carried taint and stack trace of the request.
- API log records all the accesses to DOM API.

**Table 1: Selected Taint Source and Sink**

| Type | | DOM APIs |
|---|---|---|
| Taint Source (208 in total) | Properties (206) | userAgent, innerHeight, colorDepth, Cookie etc. |
| | Functions (2) | toDataUrl, getChannelData |
| Taint Sinks (5 in total) | | XMLHTTPRequest, HTMLElement.src, WebSocket, Fetch, Navigator.sendBeacon |

### Challenges
➤ FPFlow should only propagate taint for V8 native functions implemented in C++. We solve this problem by collecting the address of all native functions during the V8 bootstrap phase.
➤ A runtime fingerprinting protection framework requires a low overhead. FPFlow reduces the overhead by optimizing the storage and propagation of JavaScript object taint.

## PRELIMINARY RESULTS

### Large scale result
➤ 7,132 sites (71.3%) are potentially performing browser fingerprinting.
➤ 6,654 sites are sending user data to third-party domain.

**Table 2: Most Used Tracking Services**

| Service | Sites | Service | Sites |
|---|---|---|---|
| doubleclick.com | 5,745 | rubiconproject.com | 1,283 |
| google-analytics.com | 4,941 | adnxs.com | 729 |
| google.com | 1,941 | criteo.com | 535 |
| googlesyndication.com | 1,672 | rlcdn.com | 499 |
| facebook.com | 1,556 | casalemedia.com | 486 |

**Table 3: Most Used Fingerprint Attributes**

| Attribute | Sites | Attribute | Sites |
|---|---|---|---|
| Cookie | 6,829 | Platform | 3,428 |
| UserAgent | 6,827 | CookieEnabled | 3,334 |
| History | 4,848 | Language | 3,310 |
| AppVersion | 4,823 | Plugins | 2,790 |
| Resolusion | 3,483 | ColorDepth | 2,783 |

**Table 4: Usage of Request Methods**

| Request Methods | Sites |
|---|---|
| Element.src | 6,345 |
| XMLHTTPRequest | 5,020 |
| Fetch | 2,251 |
| sendBeacon | 1,961 |
| WebSocket | 203 |

### Fingerprinting Script Behavior

**Tracker Loader**
- A script loading many tracking scripts according to configuration.
- Different websites have different configurations.

```
[{
    "id": 6,
    "name": "Google User Match",
    "content": "var kuid = Krux('get', 'user'); ... new Image().src = \
    'https://cm.g.doubleclick.net/pixel?google_nid=krux_digital&google_hms='+baseEncodedKuid"
}, {
    "id": 21,
    "name": "Acxiom",
    "content": "var kuid = Krux('get', 'user'); ... \
    var liveramp_url = 'https://idsync.rlcdn.com/379708.gif?partner_uid=' + kuid;"
}, {
    "id": 153.
```

**Fingerprint encoding**
- Some sites will encode browser fingerprint, especially Canvas fingerprint before sending to the remote server.

```
getLocation: function() {
    return fingerprint.util.MD5.hex_md5(location.href.split("?")[0])
},
getUserAgent: function() {
    return fingerprint.util.MD5.hex_md5(navigator.userAgent)
}
.....
t.push("canvas fp:" + fingerprint.util.MD5.hex_md5(r.toDataURL())),
```

**Fingerprinting Beacon**
- Sending many requests to one URL.
- May have different parameters each time.

```
https://g2.gumgum.com/hbid/imp?si=20829&pi=3&gdprApplies=0&uspConsent=1YN
N&schain=1.0,1!ezoic.ai,92a6db08b43e2ef47ed9427cb6e2953f,1,,,&vw=800&vh=6
00&sw=800&sh=600&pu=https://www.thewindowsclub.com/&ce=true&dpr=1&jcsi={"
t":0,"rq":8,"pbv":"3.27.0"}&ogu=https://www.thewindowsclub.com&ns=10240
https://g2.gumgum.com/hbid/imp?si=20875&pi=3&gdprApplies=0&uspConsent=1YN
N&schain=1.0,1!ezoic.ai,92a6db08b43e2ef47ed9427cb6e2953f,1,,,&vw=800&vh=6
00&sw=800&sh=600&pu=https://www.thewindowsclub.com/&ce=true&dpr=1&jcsi={"
t":0,"rq":8,"pbv":"3.27.0"}&ogu=https://www.thewindowsclub.com&ns=10240
......
```

### Limitations
➤ FPFlow uses dynamic taint analysis to analyze information flow, leading to potential false positives and false negatives.
➤ We are currently working on evaluating the accuracy of our detection result.

*For more information, contact: litianyi@pku.edu.cn*