CDF: Predictably Secure Web Documents

Peter Snyder[*], Laura Watiker[†], Cynthia Taylor[*], Chris Kanich[*]

[*] University of Illinois at Chicago
[†] Oberlin College

Overview

- The web is great! But complex!
- Complexity makes reasoning about privacy and security difficult for consumers
- Consider giving advice to non technical users
- **Knowing what we know now**:
  Is there a way to improve web security and privacy, without preventing authors from creating the types of sites users want?

The Web Today

- Interactivity is delivered as (mostly) unrestricted JavaScript

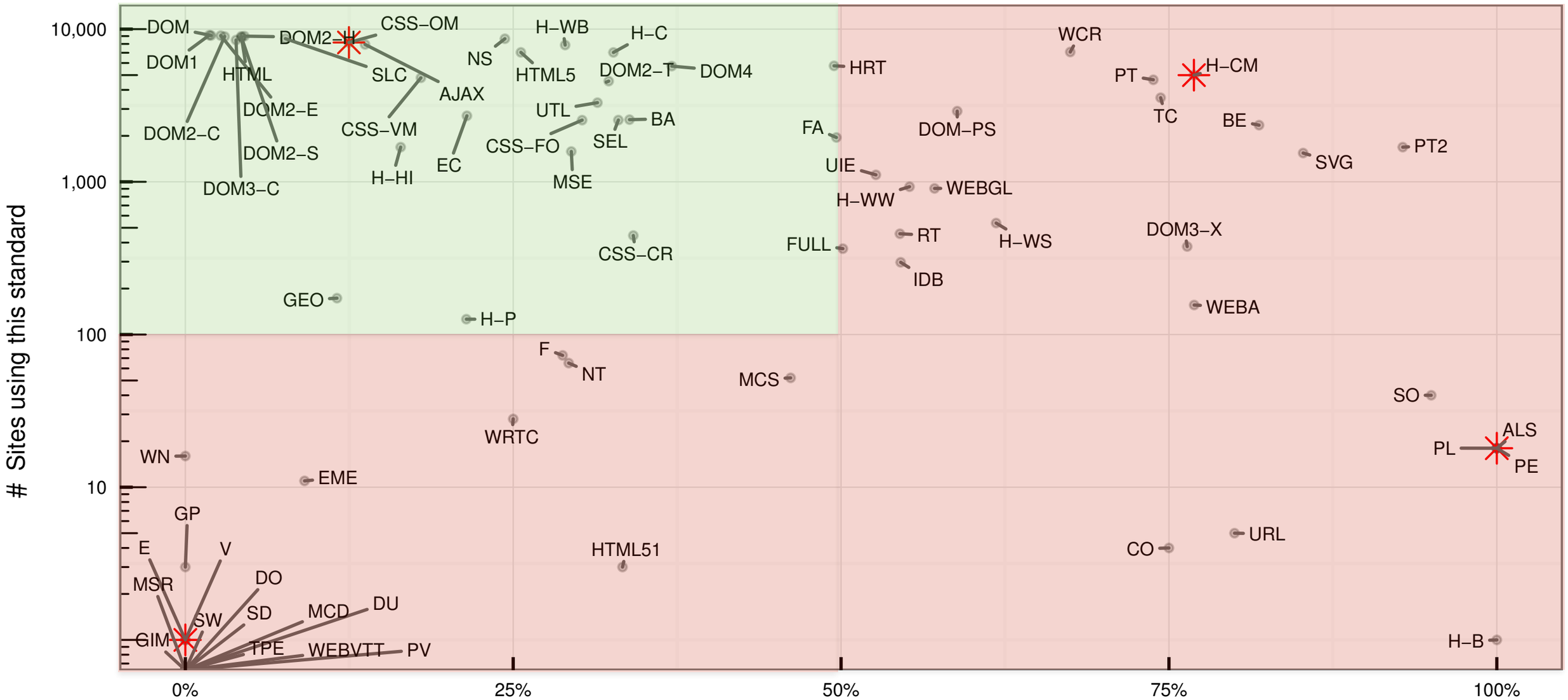
- Difficult to know code will be **benign and "useful"**:
  - form validation
  - improve user experience
  - drive user-serving widgets and page elements

- Or **malicious**:
  - fingerprint the user
  - exploit a vulnerability
  - from untrusted source (XSS)

Complexity vs. Benefit

Web API Standard	# Sites Uses	% Blocked
Gamepad	3	0.0%
Performance Timeline, Lv. 2	1,728	93.7%
WebRTC 1.0	28	29.2%
XMLHttpRequest	7,957	13.9%

Complexity vs. Benefit



Goals

Keep

- HTTP(S)
- Decentralized / Rapid Deployment
- Interactivity
- Styling / Presentation
- Web Browsers

Gain

- Predictability
- Security
- Privacy
- Removing arbitrary code execution

Approach:
Contained Document Format

1. **Document Format**:

    - JSON format, simple to check

    - Structure (like HTML)

    - Declarations of interactivity (vs. implementation)

2. **Client Proxy**: Translates CDF -> HTML+JS

3. **Trusted Libraries**: Implement safe interactivity

CDF Documents

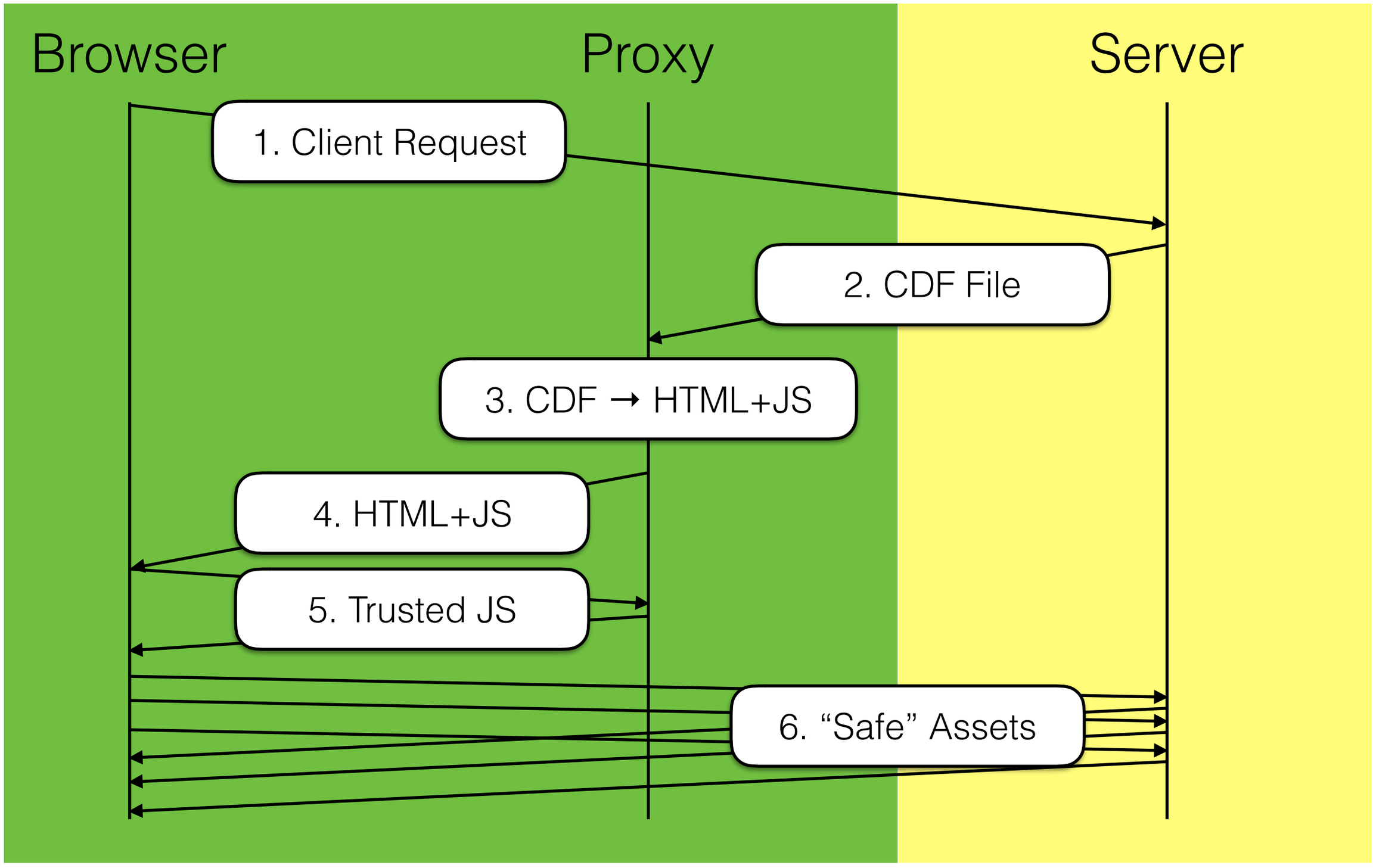
- **Structure:**
  - Comparable to HTML tags
  - Forces separation of structure and text
- **Events:**
  - Designate when something should happen
  - Taken from common DOM and framework provided events
- **Behaviors:**
  - Designate what happens when an event triggers
  - Static definition, safely converted into JavaScript by TCB
  - Selected from common web idioms (element manipulation, timers, tabs, network communication, etc)

Parser Example

```
{
  "t": "button",
  "c": [{"text": "click me"}],
  "e": [{
    "t": "click",
    "b": {
      "t": "states",
      "s": {
          "stateId": "text-change",
          "wrap": true,
          "states": [[[
            "button", {
              "t": "replace-sub",
              "c": {
              "text": "click on"
            }}]], [[
            "button", {
              "t": "replace-sub",
              "c": {
              "text": "click off"
}}]]]}}}]}
```

```
<button>click me</button>
<script>
let buttons = document.getElementsByTagName("button");
let stateIndex = 0;
let textStates = ["click on", "click off"];
buttons[0].addEventListener("click", function (event) {
  let newTextIndex = stateIndex++ % textStates.length;
  let newText = textStates[newTextIndex];
  event.target.innerHTML = newText;
});
</script>
```

CDF Flow


Browser
Proxy
Server
1. Client Request
2. CDF File
3. CDF → HTML+JS
4. HTML+JS
5. Trusted JS
6. “Safe” Assets

Advantages

- **Limited Trusted Base**
  No plugins, restricted Web API use

- **Client Side Fingerprinting**
  No JS means no JS based approaches (font / plugin enumeration, canvas fingerprinting, etc.)

- **Predictable Information Flow**
  No iframes, no HTTP referrers, restrictions on forms, “tracking speed bump"

- **Page Defacement / XSS**
  Typing in CDF documents, no script injection

Usability Tests

- **Popular blog:**
  http://www.vogue.com/

- **Online-banking:**
  https://www.bankofamerica.com/

- **Social media:**
  https://twitter.com/

- **Collaborative web application:**
  HotCRP

Conclusion

- Modern web provides web authors great flexibility
- This flexibility makes it difficult for consumers to reason about security and privacy online
- With (relatively) small changes, the web could provide more predictable privacy and security, without sacrificing expressivity.
- CDF is a design experiment to explore different privacy / capability tradeoffs.
- Source: https://github.com/bitslab/cdf
- Thank you!