# WiP: Client Side Protections Against Rogue JavaScript

**Ross Copeland**, Drew Davidson

# Current State of Web Security

# The Web as an Attack Vector

- The features and capabilities webpages have continue to evolve and grow in complexity
- This leads to more opportunities for user information to be leaked or stolen

# Attacks that Still Persist

- Cross-Site Scripting (XSS)
- ClickJacking
- IFrame Injection
- Credential Theft

Noteable Attack:

- Magecart. Stealing close to 7 million dollars

# Industry Deployed Defenses

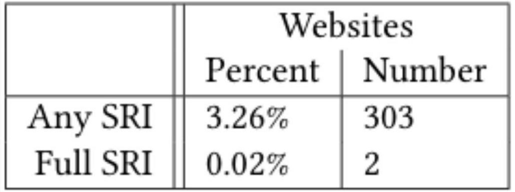
- Developer specifies scripts that should be run based off of origin or content
    - Same-Origin Policy (SOP)
        - Content from different origins cannot interact with each other
    - Content Security Policy (CSP)
        - Policy based enforcement to ensure origin of script
    - Subresource Integrity (SRI)
        - Uses cryptographic hash of script to verify contents of 3rd party script

# Problem with Modern Threat Model

- In all previous defenses:
  - The user places the responsibility of protecting all information the user views as sensitive with the developer
  - The user trusts the developer views the same sources of information as sensitive
- In practice:
  - Users are more scared of how easily stolen their information could be on the internet [1]
  - Users also are sure that their information will be stolen [2]
  - The developer and the user's privacy stances may not align completely
  - Even amongst users, their privacy policies may not be the same

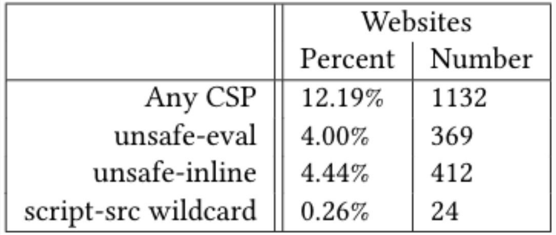
# Lack of use of CSP and SRI

- After surveying the Alexa top 10k, we prove that in the wild, defenses are rarely implemented
- When implemented, many times they are done ineffectively



| | Websites | |
|---|---|---|
| | Percent | Number |
| Any SRI | 3.26% | 303 |
| Full SRI | 0.02% | 2 |

SRI Deployment



| | Websites | |
|---|---|---|
| | Percent | Number |
| Any CSP | 12.19% | 1132 |
| unsafe-eval | 4.00% | 369 |
| unsafe-inline | 4.44% | 412 |
| script-src wildcard | 0.26% | 24 |

CSP Deployment

# Why Dev Policy != User Policy?

- The developer may not want to implement a defense for users that will break functionality of the website
- The developer may include data collection code to enhance the user experience or for advertising, but would also betray the user’s privacy
- The website visited may also be malicious

# Design Specs

The solution to this new threat model must be:

- Expressive enough to mediate the origin and functionality of JavaScript at a fine-grained level
- Implemented at a lower root of trust so that it can not be subverted by the page's JavaScript
- Adaptable by the user to fit the each individual's unique privacy stance on what information is allowed to leave the browser.

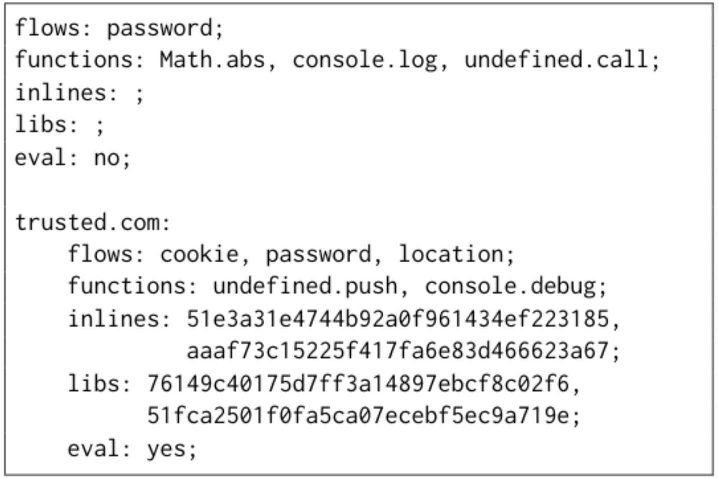
# Identity Armour (Previous Work)

- We created a system Identity Armour that is a user-defined policy enforcement engine
- Identity Armour is deployed in a browser extension, making it highly modifiable and deployable
- Our system is able to enforce the provenance and execution of scripts at a function call level of granularity

# Problem Solved?

# Limitations of Identity Armour

- The highly technical policies had to be crafted by the user
- To use the older browser features, the extension had to be packaged in a much older version of Firefox, affecting performance
- The older version of Firefox was also not compatible with the webpages of the modern web



```
flows: password;
functions: Math.abs, console.log, undefined.call;
inlines: ;
libs: ;
eval: no;

trusted.com:
    flows: cookie, password, location;
    functions: undefined.push, console.debug;
    inlines: 51e3a31e4744b92a0f961434ef223185,
             aaaf73c15225f417fa6e83d466623a67;
    libs: 76149c40175d7ff3a14897ebcf8c02f6,
          51fca2501f0fa5ca07ecebf5ec9a719e;
    eval: yes;
```

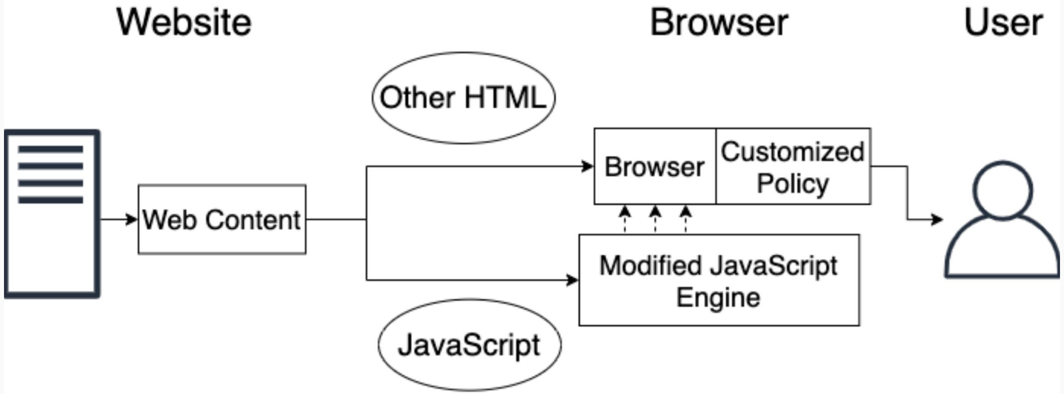
Example Policy

# Our Proposal

- The three main contributions of our proposal are:
  - **A privacy wizard**, which is a simple survey that help to determine a baseline for the user's privacy stance
  - A solution **written into the browser**
    - Implementing a browser solution would reduce the overhead in Identity Armour and have a solution that is compatible with modern web pages
  - ***A learning component*** that is able to create policies for the user that meets their specific privacy stance
    - Using a machine learning backend, we hope to collect data from users to help craft better user-specific policies

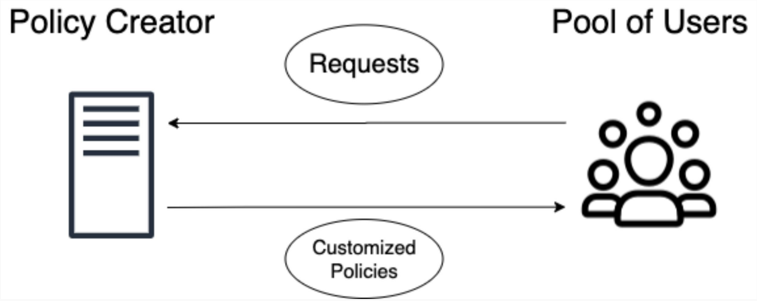
# The Browser Component

- The browser component will remain largely the same as it was in Identity Armour.
- The difference will be that instead of an extension, the tool will be implemented within the browser



# The Learning Component

- There will be a central location using machine learning with user's browsing information to create user-specific policies
- A central location is used to increase the amount of collective data to create better policies



# Discussion Questions

- How would adversarial attacks be handled in the learning component backend?
- How would the policies be verified that they align with the user's privacy stance?
- What type of machine learning problem would this best fit?
- What are some challenges for this to be deployable today?

# Discussion

# Citations

1. Pew Research Center. [n.d.]. Americans complicated feelings about socialmedia in an era of privacy concerns.http://www.pewresearch.org/fact-tank/2018/03/27/americans-complicated-feelings-about-social-media-in-an-era-of-privacy-concerns/ Accessed: 2018-08-06.
2. B. Stanton, M. F. Theofanos, S. Prettyman, and S. Furman. 2016. Security Fatigue.ITProfessional 18, 05 (sep 2016), 26–32. https://doi.org/10.1109/MITP.2016.84