CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

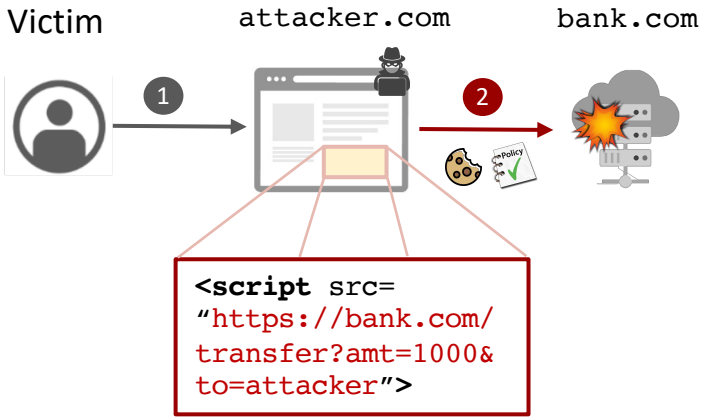
JAW: Studying Client-side CSRF with Hybrid Property Graphs and Declarative Traversals

Soheil Khodayari and Giancarlo Pellegrino
CISPA Helmholtz Center for Information Security

SCAN ME

30th USENIX Security Symposium
August 11-13, 2021

Cross-Site Request Forgery (CSRF)



Robust defenses already known:

- Referrer/Origin Checks

  getOrigin(req_2) != bank.com

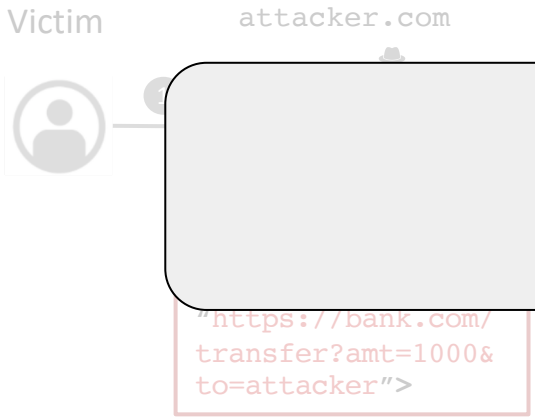
- Hard-to-guess tokens

  getToken(req_2) != CSRFToken

- SameSite Cookies
  - *SameSite=Lax* by default

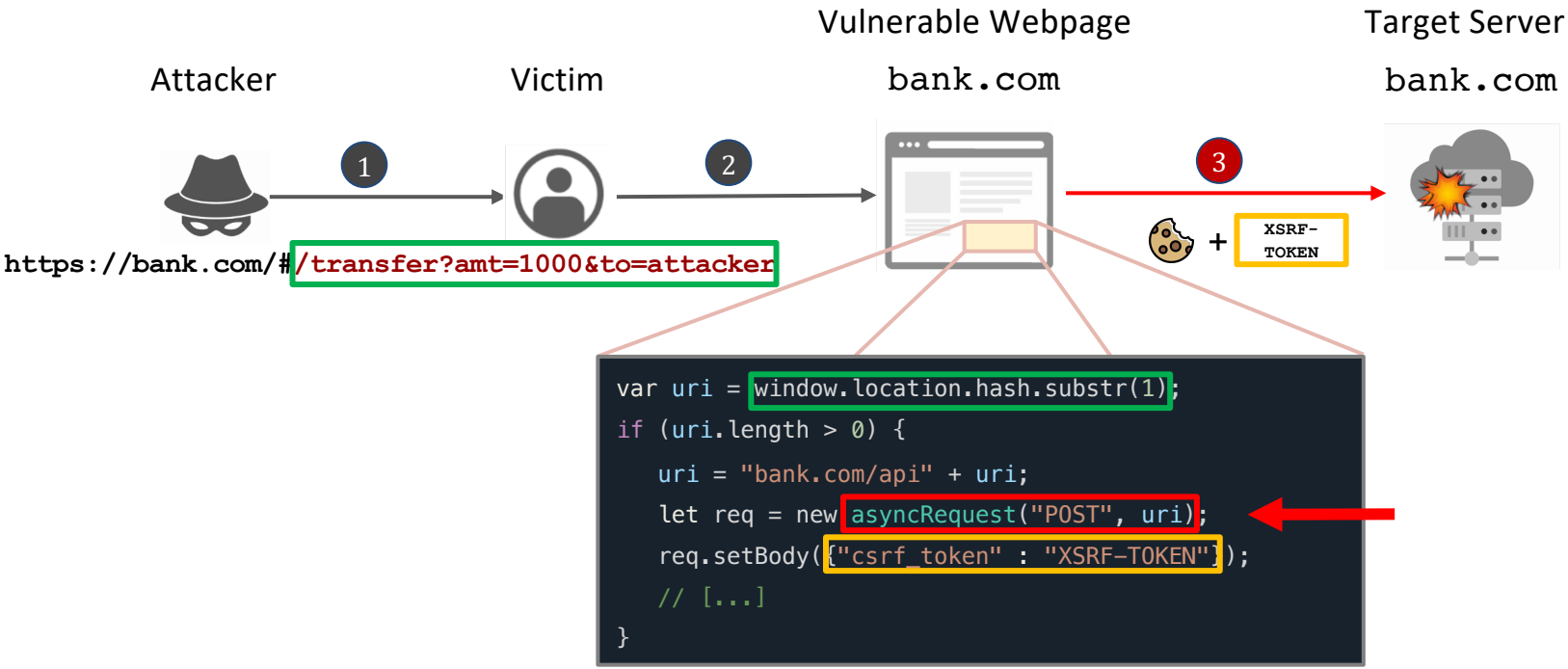
  isAuthenticated(req_2) != True

Cross-Site Request Forgery (CSRF)



Are **CSRF** attacks solved?

Client-side CSRF



Problem Statement

- Limited knowledge about client-side CSRF.
  - Facebook in 2018[1]
- **Objective:** studying client-side CSRF vulnerabilities
  - **(RQ1)** Prevalence of client-side CSRF in webapps?
  - **(RQ2)** Attacker models and exploitations?
  - **(RQ3)** Degree of attacker control?
    - E.g., path, query, domain, body

```
POST /path/file.php?q=v\r\n
Host: example.com\r\n
\r\n
{body}
```

Challenge: analysis of JavaScript programs to study client-side CSRF is not an easy task.

- **(C1)** Canonical representation for JavaScript
- **(C2)** Event-based transfer of control
- **(C3)** Dynamic web execution environment
- **(C4)** Modeling shared third-party code

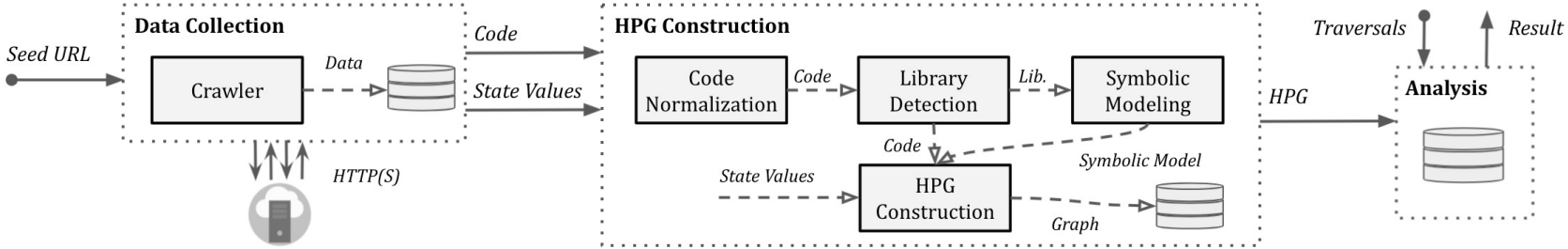
[1] **Source:** facebook.com/notes/996734990846339/

Approach Overview: JAW

- A scalable, graph-based framework for detection and exploratory analysis of client-side CSRF vulnerabilities
- Components
  - Data Collection
  - Graph Construction
  - Analysis Traversals

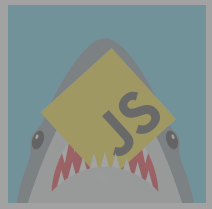
SCAN ME

https://soheilkhodayari.github.io/JAW



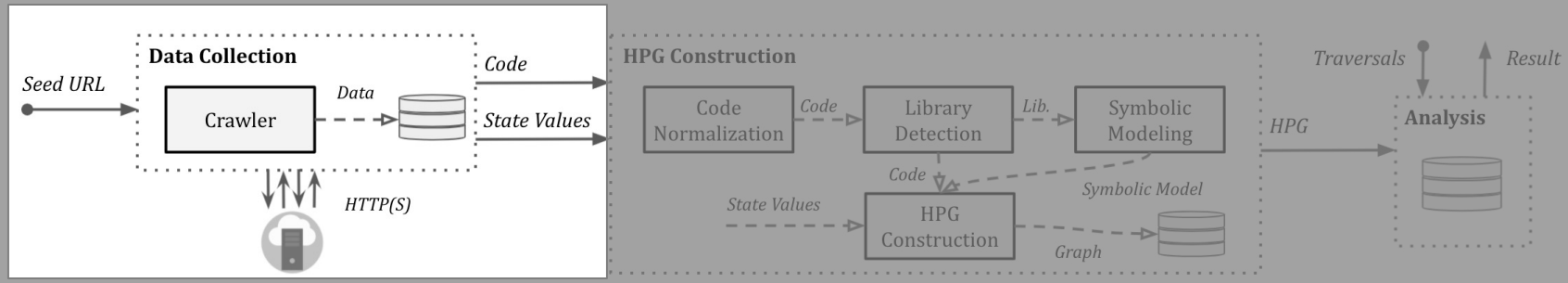
Approach Overview: JAW

- A scalable, graph-based framework for detection and exploratory analysis of client-side CSRF vulnerabilities
- Components
  - Data Collection
  - Graph Construction
  - Analysis Traversals



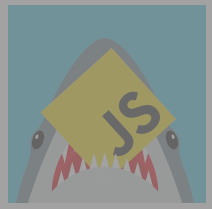
SCAN ME

https://soheilkhodayari.github.io/JAW



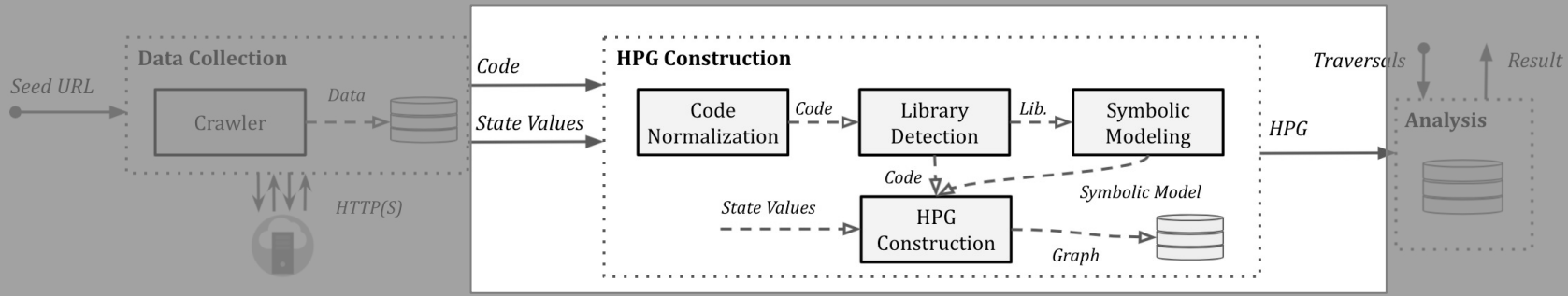
Approach Overview: JAW

- A scalable, graph-based framework for detection and exploratory analysis of client-side CSRF vulnerabilities
- Components
  - Data Collection
  - Graph Construction
  - Analysis Traversals



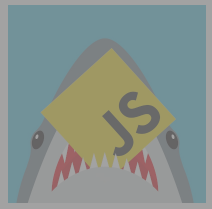
SCAN ME

https://soheilkhodayari.github.io/JAW



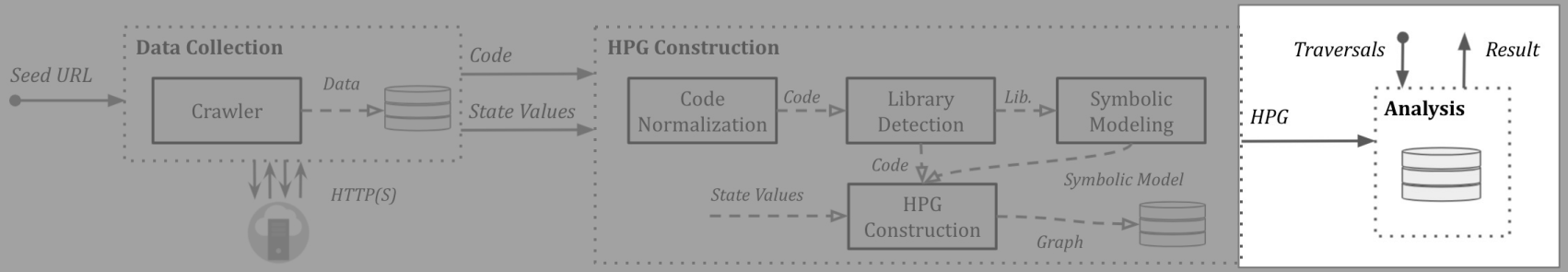
Approach Overview: JAW

- A scalable, graph-based framework for detection and exploratory analysis of client-side CSRF vulnerabilities
- Components
  - Data Collection
  - Graph Construction
  - Analysis Traversals



SCAN ME

https://soheilkhodayari.github.io/JAW



Hybrid Property Graphs (HPGs): Building Blocks

- **Code Representation (Static)**
  - Abstract Syntax Tree (AST)
  - Control Flow Graph (CFG)
  - Program Dependence Graph (PDG)
  - Inter-Procedural Call Graph (IPCG)
  - Event Registration, Dispatch and Dependency Graph (ERDDG)
  - Semantic Types and Symbolic Models

CPG for C/C++ [Yamaguchi et al, S&P'14]

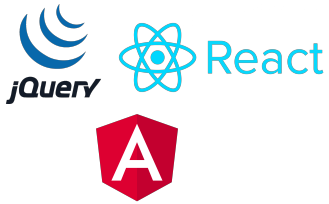
CPG for PHP [Bakes et al., EuroS&P'17]

HPG for JavaScript

- **State Values (Dynamic)**
  - Event Traces
  - Environment Properties

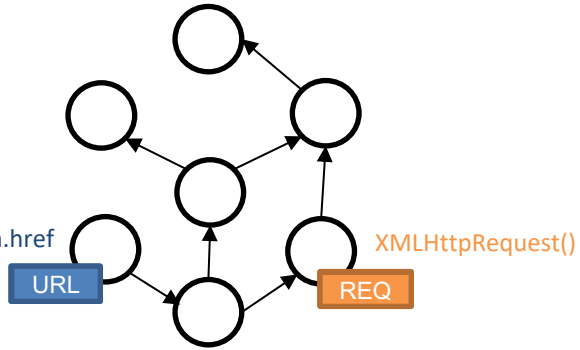
Symbolic Models and Semantic Types Propagation

- External libraries: over 60% of the total LoC of each webpage.
- **Problem:**
  - Existing approaches: Inefficient, include library code in the analysis
- **Idea:** Shared models for JS libraries

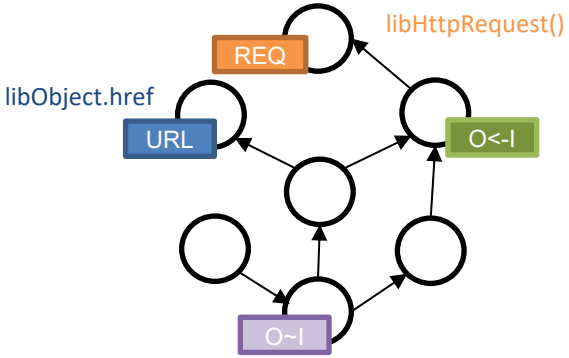



```
<script src="lib.js"/>
```

HPG for lib.js



Symbolic Model for lib.js



Evaluation: Forgeable Requests

- Evaluated JAW with all webapps from the Bitnami catalog
  - 106 webapps
  - 228M LoC
- Detected 12,701 forgeable requests affecting 87 webapps

Exploitations

- Manually looked for practical exploitations in 516 requests
- Created exploits for 203 requests of seven webapps
  - SuiteCRM, SugarCRM, Neos, Kibana, Modx, Odoo, and Shopware
  - Account takeover, deleting user assets, …

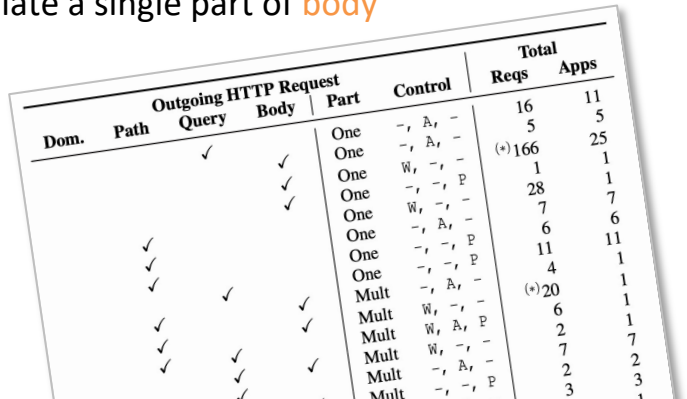
Input Source	Forgeable	Apps
DOM.COOKIES	67	5
DOM.READ	12,268	83
*-Storage	76	8
DOC.REFERRER	1	1
POST-MESSAGE	8	8
WIN.NAME	1	1
WIN.LOC	280	12
Total Forgeable	12,701	87
Total Requests	49,366	106

Evaluation: Analysis of Forgeable Requests

- Exploitation landscape can be influenced by:
  - Type of controllable fields
  - Operation to forge a field

```
POST /path/file.php?q=v\r\n
Host: example.com\r\n
\r\n
{body}
```

- Identified 25 distinct templates. For example:
  - 185/ 516 requests: manipulate any part of domain + path + query
  - 20/ 516 requests: manipulate multiple parts of path + body
  - 166/ 516 requests: manipulate a single part of body
  - See the paper for more



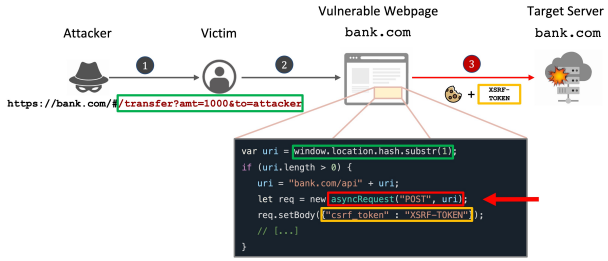
Outgoing HTTP Request					Total	
Dom.	Path	Query	Body	Part	Control	Reqs	Apps
		✓	✓	One	–, A, –	16	11
			✓	One	–, A, –	5	5
			✓	One	W, –, –	(*)166	25
				One	–, –, P	1	1
				One	W, –, –	28	1
				One	–, A, –	7	1
				One	–, –, P	6	7
				One	–, –, P	11	6
				Mult	–, A, –	4	11
				Mult	W, –, –	(*)20	1
				Mult	W, A, P	6	1
				Mult	W, –, –	2	1
				Mult	–, A, –	7	1
				Mult	–, –, P	2	7
				Mult		3	2
						1	3

Conclusion

https://soheilkhodayari.github.io/JAW

CISPA HELMHOLTZ CENTER FOR INFORMATION SECURITY

Client-side CSRF



Hybrid Property Graphs (HPGs): Building Blocks

- **Code Representation (Static)**
  - Abstract Syntax Tree (AST)
  - Control Flow Graph (CFG)
  - Program Dependence Graph (PDG)
  - Inter-Procedural Call Graph (IPCG)

  CPG for C/C++ [Yamaguchi et al, S&P'14]

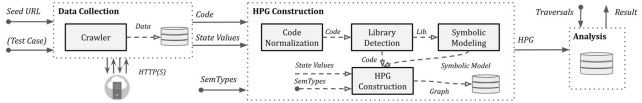
  CPG for PHP [Bakes et al., EuroS&P'17]
  - Event Registration, Dispatch and Dependency Graph (ERDDG)
  - Semantic Types and Symbolic Models

  HPG for JavaScript
- **State Values (Dynamic)**
  - Event Traces
  - Environment Properties

Approach Overview: JAW

- A scalable, graph-based framework for detection and exploratory analysis of client-side CSRF vulnerabilities
- Components
  - Data Collection
  - Graph Construction
  - Analysis Traversals

https://soheilkhodayari.github.io/JAW



Evaluation: Forgeable Requests

- Evaluated JAW with all webapps from the Bitnami catalog
  - 106 webapps
  - 228M LoC
- Detected 12,701 forgeable requests affecting 87 webapps

Exploitations

- Manually looked for practical exploitations in 516 requests
- Created exploits for 203 requests of seven webapps
  - SuiteCRM, SugarCRM, Neos, Kibana, Modx, Odoo, and Shopware
  - Account takeover, deleting user assets, ...

Input Source	Forgeable	Apps
DOM.COOKIES	67	5
DOM.READ	12,268	83
*-Storage	76	8
DOC.REFERRER	1	1
POST-MESSAGE	8	8
WIN.NAME	1	1
WIN.LOC	280	12
Total Forgeable	12,701	87
Total Requests	49,366	106

SCAN ME

@Soheil__K

Thank You!