Coverage and Secure Use Analysis of Content Security Policies via Clustering

IEEE European Symposium on Security and Privacy 2023

Mengxia Ren[1], Chuan Yue[1]

[1]Colorado School of Mines, Golden, Colorado, USA

Introduction

- Content Security Policy (CSP) is a standardized leading technique for protecting webpages against attacks such as Cross Site Scripting (XSS).
- A CSP is composed of a set of directives, each of which is a pair of whitespace-delimited directive name and directive value.

  An example of CSP:

  img-src https://img.com/img_1.png https://img.com/img_2.png
  script-src https://example.com/script.js 'self' 'unsafe-eval' 'unsafe-inline';

- CSP deployment ways: via HTTP response headers or <meta> tags
- CSP deployment modes: the enforcement mode and the report-only mode

Introduction

- Content Security Policy (CSP) is a standardized leading technique for protecting webpages against attacks such as Cross Site Scripting (XSS).
- A CSP is composed of a set of directives, each of which is a pair of whitespace-delimited directive name and directive value.

  An example of CSP:

```
img-src https://img.com/img_1.png  https://img.com/img_2.png
script-src https://example.com/script.js 'self' 'unsafe-eval' 'unsafe-inline';
```

- CSP deployment ways: via HTTP response headers or <meta> tags
- CSP deployment modes: the enforcement mode and the report-only mode

Introduction

- Content Security Policy (CSP) is a standardized leading technique for protecting webpages against attacks such as Cross Site Scripting (XSS).
- A CSP is composed of a set of directives, each of which is a pair of whitespace-delimited directive name and directive value.

  An example of CSP:

```
img-src https://img.com/img_1.png  https://img.com/img_2.png
script-src https://example.com/script.js 'self' 'unsafe-eval' 'unsafe-inline';
```

- CSP deployment ways: via HTTP response headers or <meta> tags
- CSP deployment modes: the enforcement mode and the report-only mode

Introduction

- Content Security Policy (CSP) is a standardized leading technique for protecting webpages against attacks such as Cross Site Scripting (XSS).
- A CSP is composed of a set of directives, each of which is a pair of whitespace-delimited directive name and directive value.

  An example of CSP:

  img-src https://img.com/img_1.png https://img.com/img_2.png
  script-src https://example.com /script.js 'self' 'unsafe-eval' 'unsafe-inline';

- CSP deployment ways: via HTTP response headers or <meta> tags
- CSP deployment modes: the enforcement mode and the report-only mode

Introduction

- Content Security Policy (CSP) is a standardized leading technique for protecting webpages against attacks such as Cross Site Scripting (XSS).
- A CSP is composed of a set of directives, each of which is a pair of whitespace-delimited directive name and directive value.

  An example of CSP:

```
img-src https://img.com/img_1.png  https://img.com/img_2.png
script-src https://example.com/script.js 'self' 'unsafe-eval' 'unsafe-inline';
```

- CSP deployment ways: via HTTP response headers or <meta> tags
- CSP deployment modes: the enforcement mode and the report-only mode

Introduction

It is hard to properly deploy CSPs, and security issues or errors are often found in the deployed CSPs.

- Security issues of “script-src”, “object-src”, and “default-src” directives
- Deployment issues (e.g., policy misconfigurations and insecure whitelisted entries)
- CSP has been increasingly used for other purposes (e.g., frame control and TLS enforcement)

Limitations of Existing Studies

- The vulnerability analysis is for specific directives.
- CSP analysis based on the overall statistics of CSP security issues is based on some specific rules.

- It is important to analyze CSPs from both directive coverage and secure use perspectives.
  - A CSP that does not contain a vulnerable directive may not cover all needed resource thus leading to the insufficient protection of a webpage.
  - A CSP that contains a vulnerable directive may still be able sufficiently protect a webpage as long as the remaining directives can cover all resources.

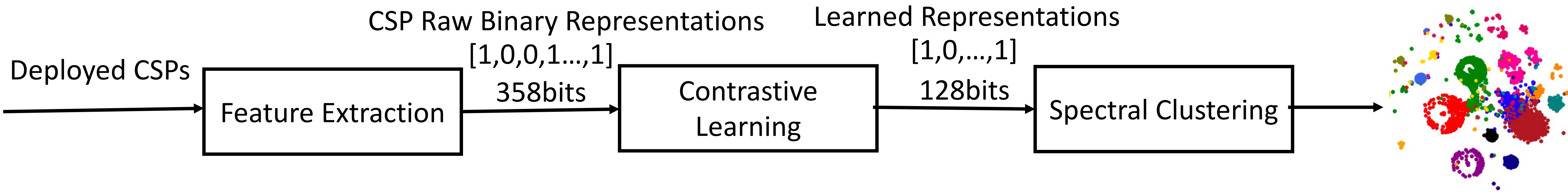
Our Goal and Approach

We aim to analyze the protection capabilities of the deployed CSPs from the directive coverage and secure use perspectives via a clustering approach.

Contrastive Spectral Clustering (CSC) Algorithm

Contrastive learning is effective in learning informative representations from unlabeled data samples for performing multiple types of downstream tasks (e.g., image, text, and graph related classification and clustering)

Spectral clustering is superior to traditional clustering approaches.



- Feature extraction of deployed CSPs
- Contrastive learning for CSP representation learning
- Spectral clustering approach for CSP clustering

Policy Feature Design and Extraction

Each policy feature is defined as a (directive name, directive value token type) pair, and its value is defined as a binary value.

e.g.,
The “script-src ‘self’ ” directive is represented as a feature (script-src,self).

The CSP “script-src ‘self’ ‘unsafe-inline’” contains two features: (script-src,self) and (script-src,unsafe-inline).

In total, we defined and extracted 530 (directive name, directive value token type) pairs based on the latest CSP Level 3 specification.

Policy Feature Design and Extraction

Table 1: Directive Names and the Allowed Directive Value Types (defined in Table 2) or Values

Directive Category	*Directive Name*	*Allowed Directive Value Types or Values*
Fetch Directives	default-src	Types I to V
	child-src	Types I to V
	connect-src	Types I to V
	font-src	Types I to V
	frame-src	Types I to V
	img-src	Types I to V
	manifest-src	Types I to V
	media-src	Types I to V
	prefetch-src	Types I to V
	object-src	Types I to V
	worker-src	Types I to V
	script-src	Types I to V
	script-src-attr	Types I to V
	script-src-elem	Types I to V
	style-src	Types I to V
	style-src-attr	Types I to V
	style-src-elem	Types I to V
Document Directives	base-uri	Types I to V
	sandbox	Type VI
Navigation Directives	form-action	Types I to V
	frame-ancestors	Types I and II, 'self', 'none'
	navigate-to	Types I to V
Other Directives	block-all-mixed-content	No value is needed
	upgrade-insecure-requests	No value is needed
	trusted-types	'none', 'allow-duplicates', policyname
	plugin-types	Type VII
	require-sri-for	script, style
	require-trusted-types-for	'script'

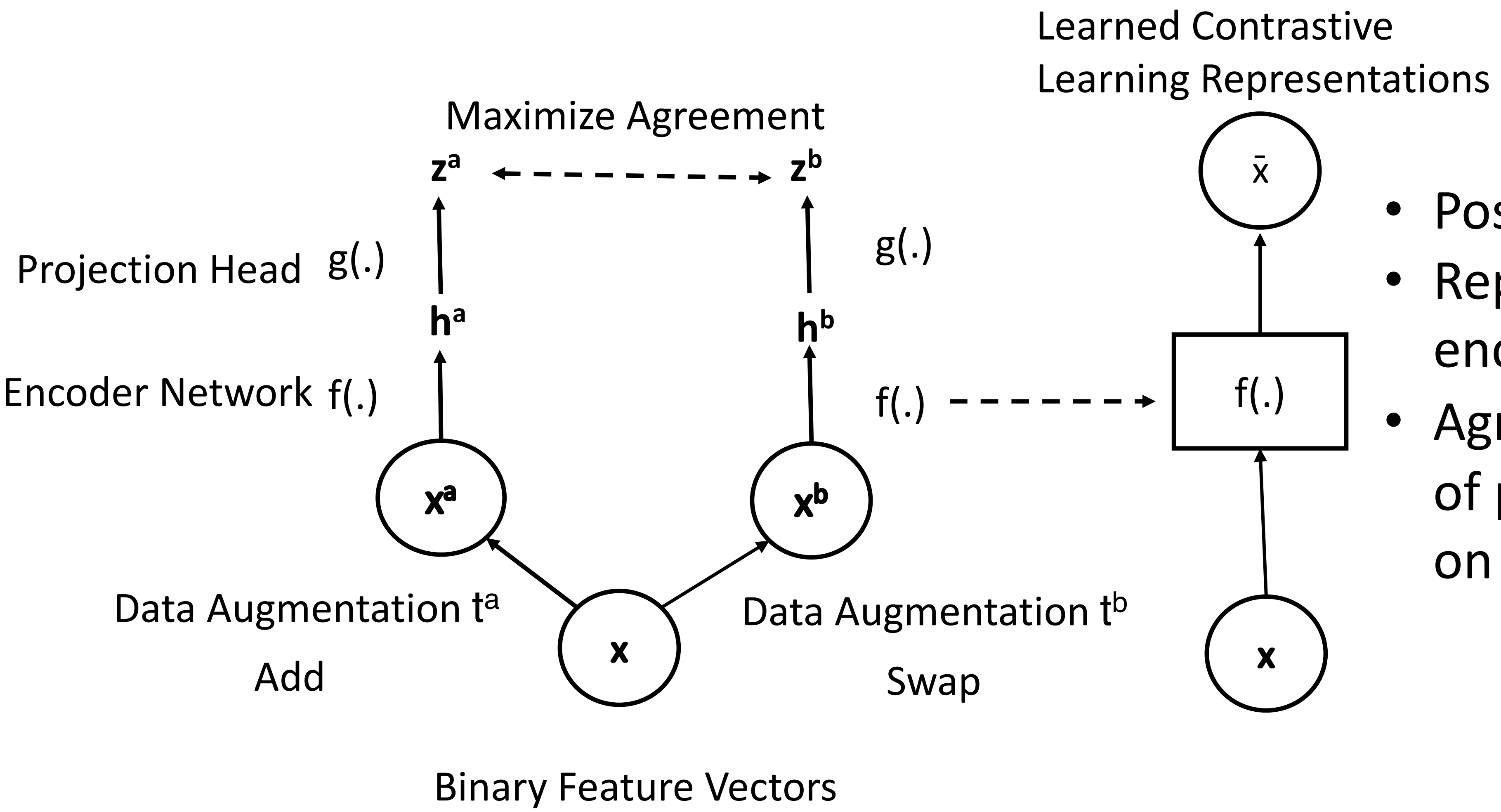
Policy Feature Design and Extraction

Table 2: Directive Value Types and Value Token Types

Directive Value Type#	*Directive Value Type*	*Directive Value Token Types*
I	scheme-source	7 in total: https, http, wss, ws, data, blob, other schemes
II	host-source	5 in total: a host-source value is specified with the syntax: *[scheme-part "://"] host-part [":" port-part] [path-part]*; we bin possible values into five types: *, *.external.domain (*.exdo), *.same.domain (*.sado), external domain (exdo), same domain (sado)
III	keyword-source	8 in total: 'self', 'unsafe-inline', 'unsafe-eval', 'strict-dynamic', 'unsafe-hashes', 'none', 'report-sample', 'unsafe-allow-redirects'
IV	nonce-source	1 in total: 'nonce-<base64-value>'
V	hash-source	3 in total: 'sha256-<base64-value>', 'sha512-<base64-value>', 'sha384-<base64-value>'
VI	sandbox values	15 in total: "", allow-downloads, allow-downloads-without-user-activation, allow-forms, allow-modals, allow-orientation-lock, allow-same-origin, allow-scripts, allow-storage-access-by-user-activation, allow-top-navigation, allow-top-navigation-by-user-activation, allow-pointer-lock, allow-popups, allow-popups-to-escape-sandbox, allow-presentation
VII	plugin-types values	1 in total: all MIME type <type>/<subtype> tokens are binned into one type
VIII	customized values	3 in total: style, script or 'script', 'allow-duplicates'

Contrastive Learning (CL) Algorithm

Contrastive learning is effective in learning informative representations from unlabeled examples



- Positive samples generation
- Representation learning via encoder network f
- Agreement Maximization of positive samples based on learned representations

Contrastive Spectral Clustering (CSC) Algorithm

- Learning the representations by CL algorithm
- For each number of clusters k_i ($k_i \in \{k_{min}, k_{max}\}$):

  Clustering CSPs based on the representations learned by the CL algorithm into k_i clusters via Spectral Clustering (based on eigenvector matrix & K-Means)

- Selecting the optimal number of clusters k_{opt} with the corresponding clustering result based on the Silhouette score

Data Collection

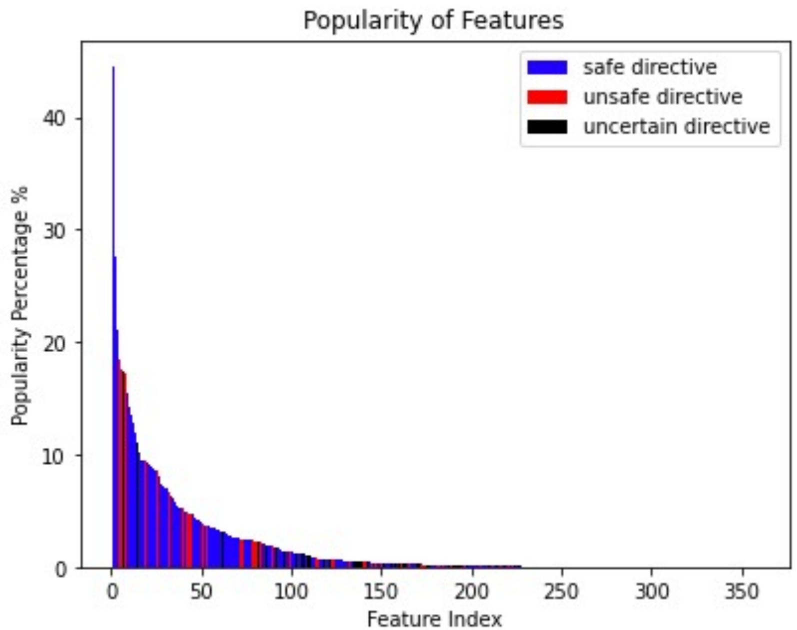
- A constructed Google Chrome browser extension:
  - Visiting the homepages and 10 subpages of the Alexa top 100K websites from Nov. in 2021 to Apr. in 2022.
  - Collecting all HTTP(s) requests and responses, collecting CSPs, and saving the loaded HTML documents.

Our dataset includes 13,317 CSP deployed homepages, 358 CSP features, and 110,718 subpages of the 13,317 CSP deployed homepages

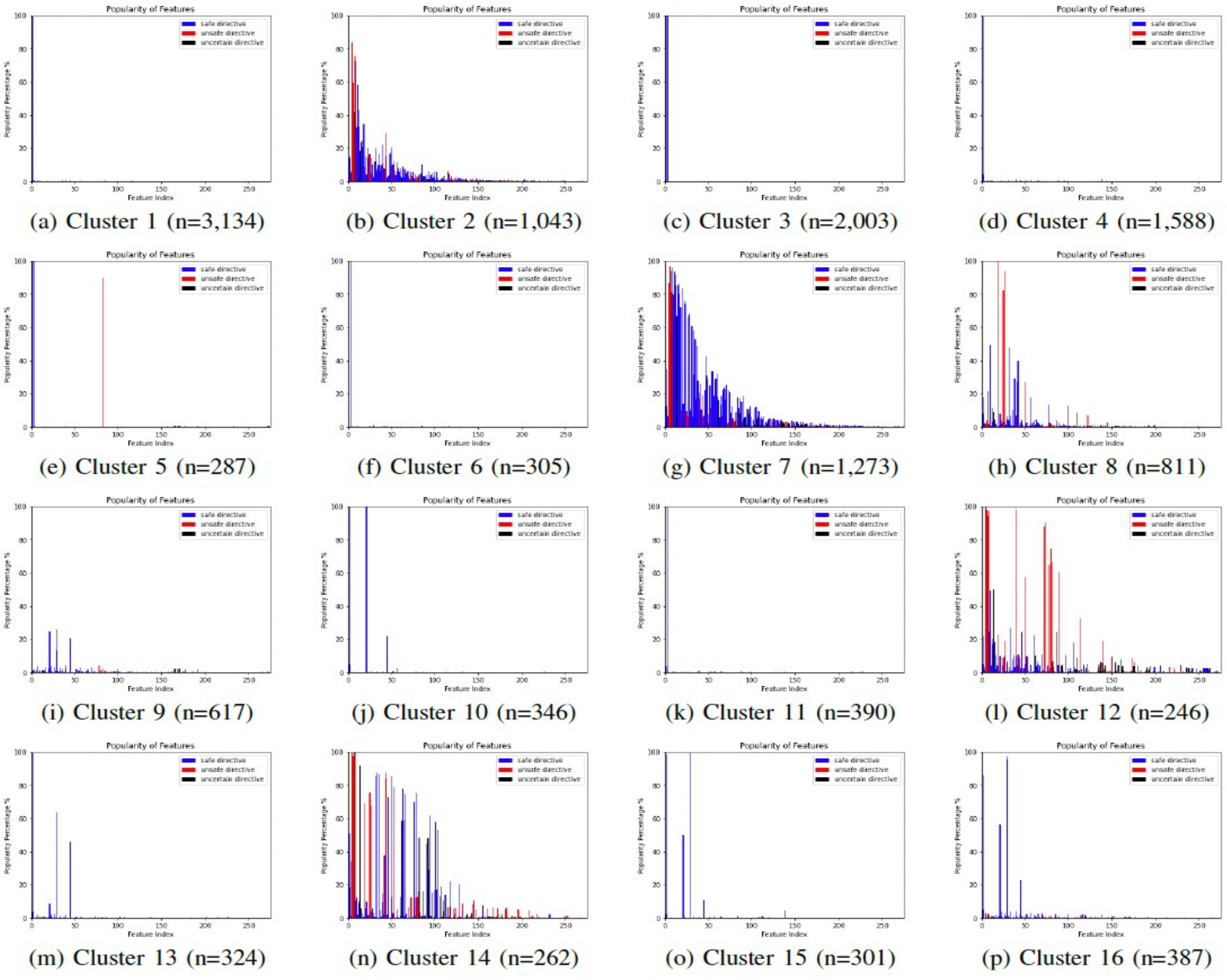
Popularity of Features

- Among these 358 CSP features, 219, 70, and 69 are labeled as safe, unsafe, and uncertain
  - ➢ Safe features: clearly provide some control on resources or behaviors and would not incur potential risks
  - ➢ Unsafe features: clearly incur potential risks
  - ➢ Uncertain features: other policy features

Top 1: “upgrade-insecure-requests”
Top 2: “frame-ancestors ‘self ’ ”
Top 3: “block-all-mixed-content”
Top 4 : “frame-ancestors ‘none’ ”
Top 5: “script-src ‘unsafe-inline’ ”
Top 6: “style-src ‘unsafe-inline’ ”



Clustering Results



(a) Cluster 1 (n=3,134) (b) Cluster 2 (n=1,043) (c) Cluster 3 (n=2,003) (d) Cluster 4 (n=1,588)
(e) Cluster 5 (n=287) (f) Cluster 6 (n=305) (g) Cluster 7 (n=1,273) (h) Cluster 8 (n=811)
(i) Cluster 9 (n=617) (j) Cluster 10 (n=346) (k) Cluster 11 (n=390) (l) Cluster 12 (n=246)
(m) Cluster 13 (n=324) (n) Cluster 14 (n=262) (o) Cluster 15 (n=301) (p) Cluster 16 (n=387)

Figure 5: Feature Popularity of Each Cluster

Group 1: Clusters 1,3,4,6, and 11
Group 2: Clusters 2 and 7
Group 3: Clusters 5, 9, 10, 13, 15, and 16
Group 4: Clusters 8, 12, and 14

Coverage and Secure Use Analysis

Table 3: Summary of the Main Aims of the CSPs and the CSP Security Level Analysis Results for the 16 Clusters

Cluster No.	Main Aims of the CSPs in the Cluster	Number of Websites in the Cluster	Overall Level on Directive Coverage	Overall Level on Directive Secure Use
1	TLS enforcement via "upgrade-insecure-requests"	3,134	low-level	high-level
2	XSS mitigation via "script-src" and "style-src" directives	1,043	low-level	low-level
3	TLS enforcement via "block-all-mixed-content"; Framing control via "frame-ancestors 'none' "	2,003	low-level	high-level
4	Framing control via "frame-ancestors 'self' "	1,588	low-level	high-level
5	TLS enforcement via "upgrade-insecure-requests" and "block-all-mixed-content"; Framing control via "frame-ancestors *"	287	low-level	low-level
6	TLS enforcement via "block-all-mixed- content"	305	low-level	high-level
7	XSS mitigation via fetch directives with external domain combinations and a "self" value	1,273	medium-level	medium-level
8	XSS mitigation via a "default-src" directive	811	low-level	low-level
9	Framing control via whitelisting sources	617	low-level	low-level
10	Framing control via "frame-ancestors exdo" and "frame-ancestors 'self' "	346	low-level	high-level
11	Framing control via "frame-ancestors 'none' "	390	low-level	high-level
12	XSS mitigation via fetch directives with a "*" value	246	low-level	low-level
13	Framing control via "frame-ancestors *.sado" and "frame-ancestors 'self' "	324	low-level	high-level
14	XSS mitigation via fetch directives with blob:, data:, and https: schemes	262	low-level	medium-level
15	Framing control via "frame-ancestors *.exdo"	301	low-level	high-level
16	Framing control via "frame-ancestors *.exdo" and "frame-ancestors 'self' "	387	low-level	high-level

- No cluster has its CSPs at the high-level on directive coverage, and CSPs in 15 clusters are generally at the low-level on directive coverage.
- CSPs in nine, two, and five clusters are generally at the high-level, medium-level, and low-level on directive secure use.

High-level Takeaways from Our Study

- Clustering approach is beneficial
- The importance of analyzing CSPs from both directive coverage and secure use perspectives
- Four new findings:
  - Unique CSP patterns and limited aims
  - 15 clusters are at the low-level on the directive coverage and five clusters are at the low-level on the secure use of directives
  - Web development platforms (e.g., Shopify, Webflow, and HubSpot) contributed to the specific CSP patterns of many websites
  - Severe problems (e.g., no fetch directives, allowing any webpages to embed a current webpage, and the prevalence of unsafe “default-src” directives) in specific clusters

Recommendations for Web Developers

- Developers of CSP-deployed websites:
  - ➢ improve CSP from both the directive coverage and the secure use perspectives
- Web developers using a web development platform:
  - ➢ ascertain and leverage the CSP support of the web development platform
  - ➢ upgrade the protection capability of their CSPs when they further customize the policies
- Web developers:
  - ➢ avoid those severe problems in specific clusters

Conclusion

- We proposed to take the clustering approach for analyzing the security levels of the deployed CSPs from the directive coverage and secure use perspectives.
- We designed 530 policy features based on the latest CSP Level 3 specification.
- We designed a CSC algorithm that leverages the advantages of spectral clustering and contrastive learning for automatically categorizing CSPs.
- We performed a large-scale measurement study on 100K websites, categorized the CSPs deployed on 13,317 homepages into 16 clusters with different characteristics, and analyzed the security levels of the CSPs in each unique cluster to help promote the proper deployment of CSPs.

Thank you!