CSP: Advanced Tactics

Jason Gillam

Jason Gillam

- CIO - Secure Ideas, LLC
- IANS Faculty member
- OWASP Project Committee Member
- Hacking things since ~2009
- Programming since ~1984

- Other Interests
  - Automation
  - Business Strategy
  - Gaming
  - Homebrewing

A Quick Review

Content-Security-Policy Examples

Here a few common scenarios for content security policies:

Allow everything but only from the same origin

```
default-src 'self';
```

Only Allow Scripts from the same origin

```
script-src 'self';
```

Allow Google Analytics, Google AJAX CDN and Same Origin

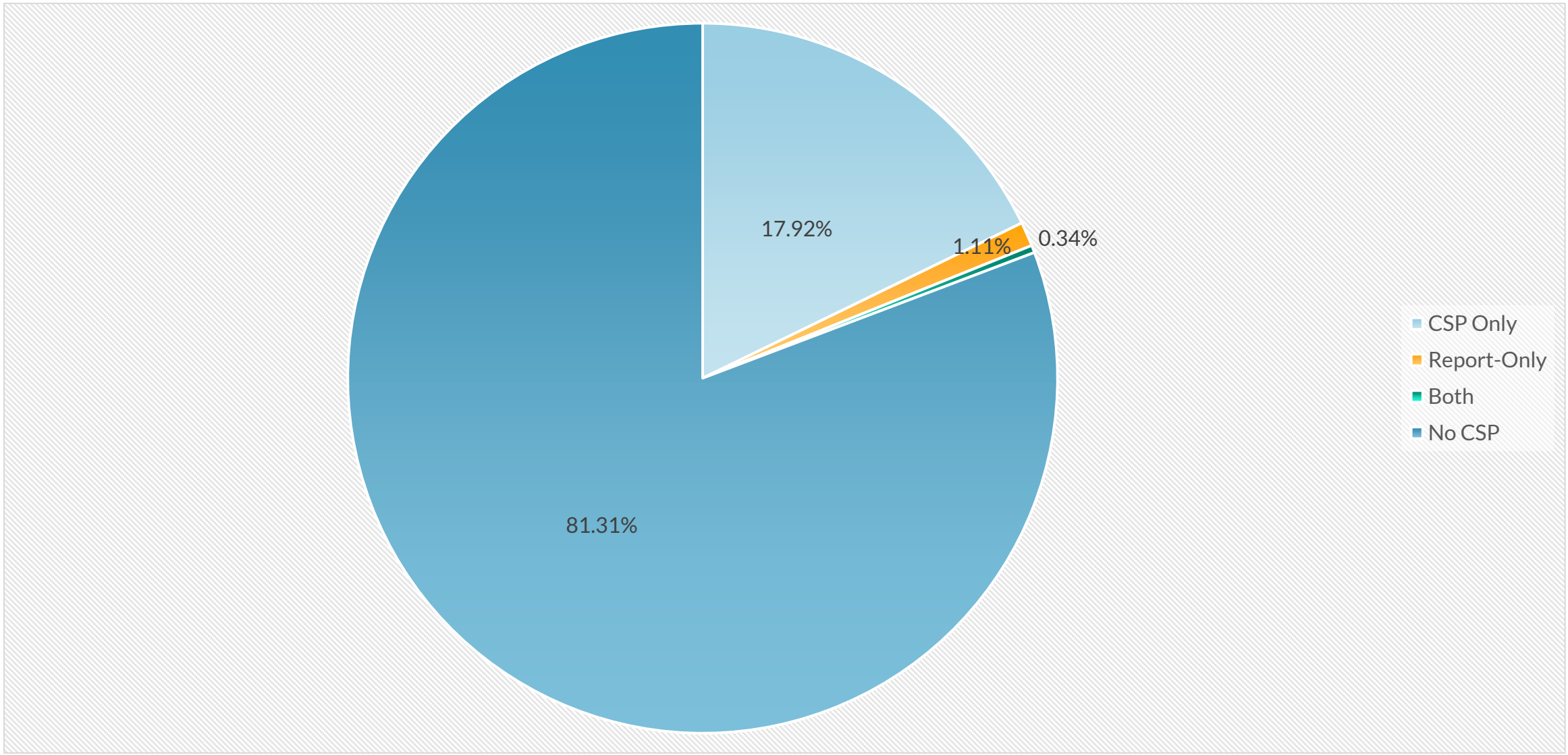
```
script-src 'self' www.google-analytics.com ajax.googleapis.com;
```

Starter Policy

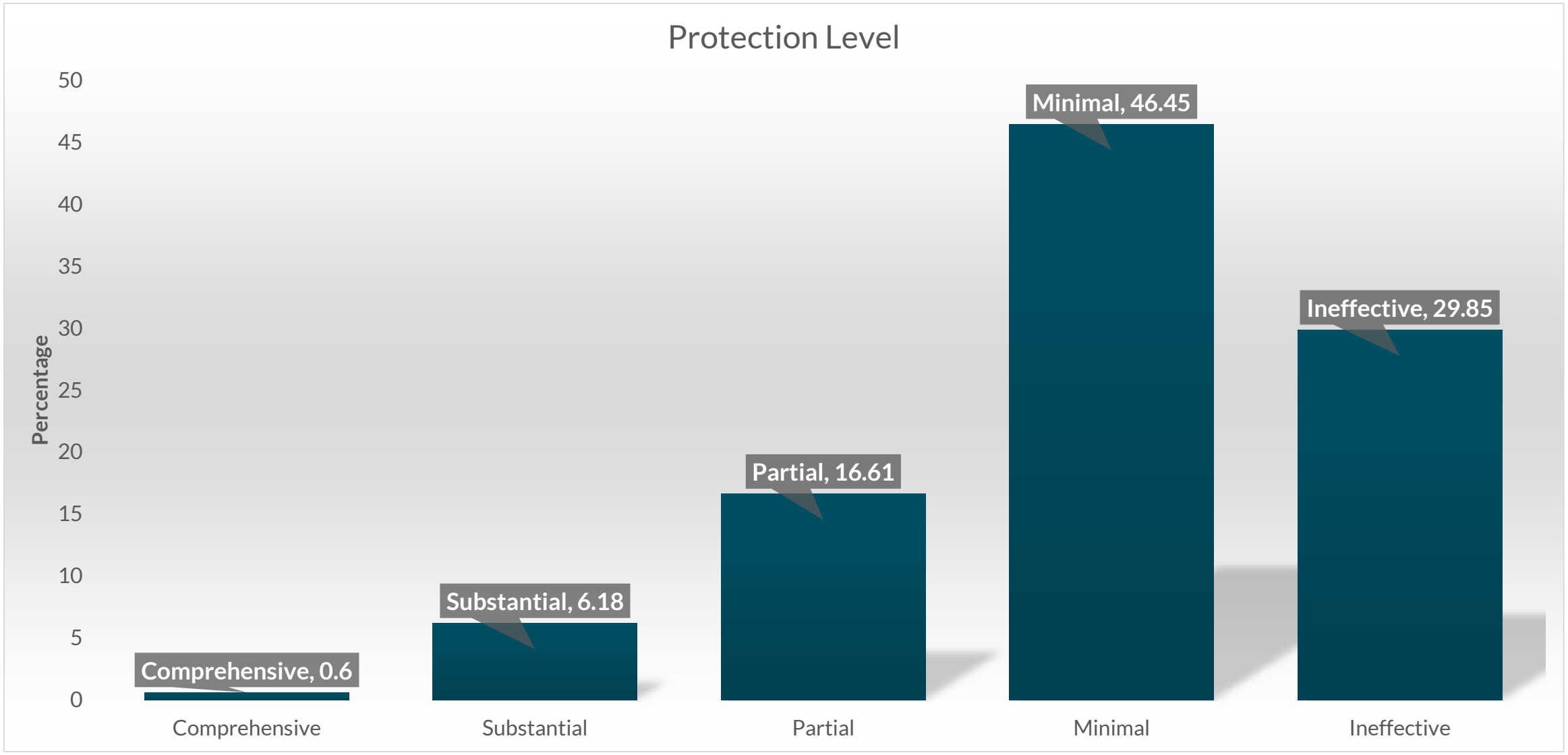
This policy allows images, scripts, AJAX, and CSS from the same origin, and does not allow any other resources to load (eg object, frame, media, etc). It is a good starting point for many sites.

```
default-src 'none'; script-src 'self'; connect-src 'self'; img-src 'self'; style-src
'self';
```

How well has the internet adopted CSP in 2025?



CSPs exist, but are they working?



Tiered Rating System: 6 categories, rated 1-5

Script Execution Control

- 1 = missing script-src with no default-src; 5 = nonce with strict-dynamic, no unsafe-inline, trusted types API

Style Injection Control

- 1 = missing style-src with no default-src; 5 = nonces or hashes

Object/Media Control

- 1 = missing object-src with no default-src; 5 = no plugin execution permitted

Frame Control

- 1 = missing frame-ancestors; 5 = frame-ancestors set to ‘self’ or ‘none’

Form Action Control

- 1 = missing form-action; 5 = form-action restricted to ‘self’

Base URI Control

- 1 = missing base-uri directive; 5 = base-uri restricted to ‘self’ or ‘none’

Thresholds

Comprehensively Protected	• All areas at 3 or higher
Substantially Protected	• 4-5 areas at level 3 or higher
Partially Protected	• 2-3 areas at level 3 or higher
Minimally Protected	• All scores below 3 but at least one score = 2, OR exactly 1 component ≥ 3
Ineffective	• All areas at level 1

CSP Essential Directives

frame-ancestors

- Prevent clickjacking

form-actions

- Controls form submissions

base-uri

- Prevents base tag injection

object-src

- Controls plugins/embeds (often none)

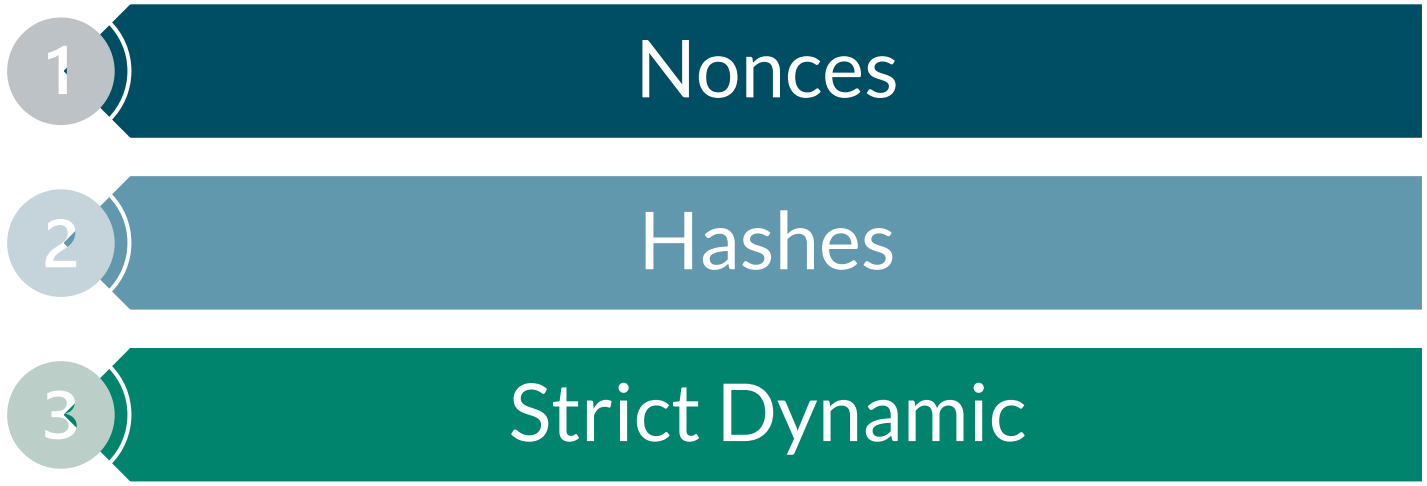
img-src

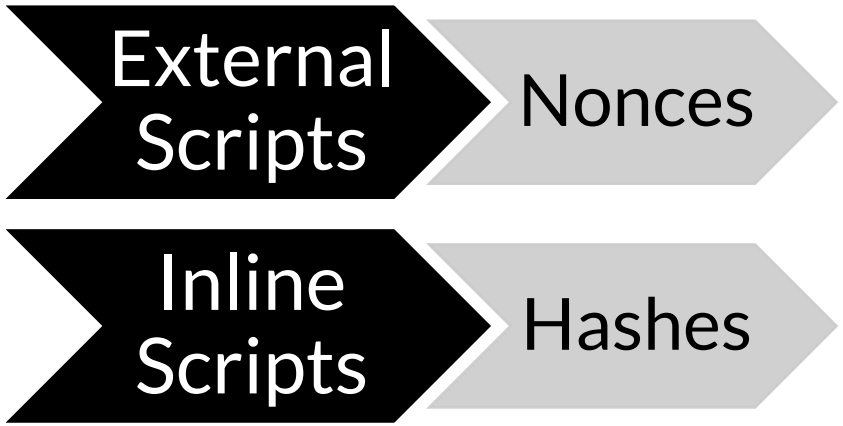
- Image loading restrictions

connect-src

- XHR/fetch/websocket connections

Advanced Tactics: The Big Three





Modern Architecture: Use a standard (code) library

Helmet

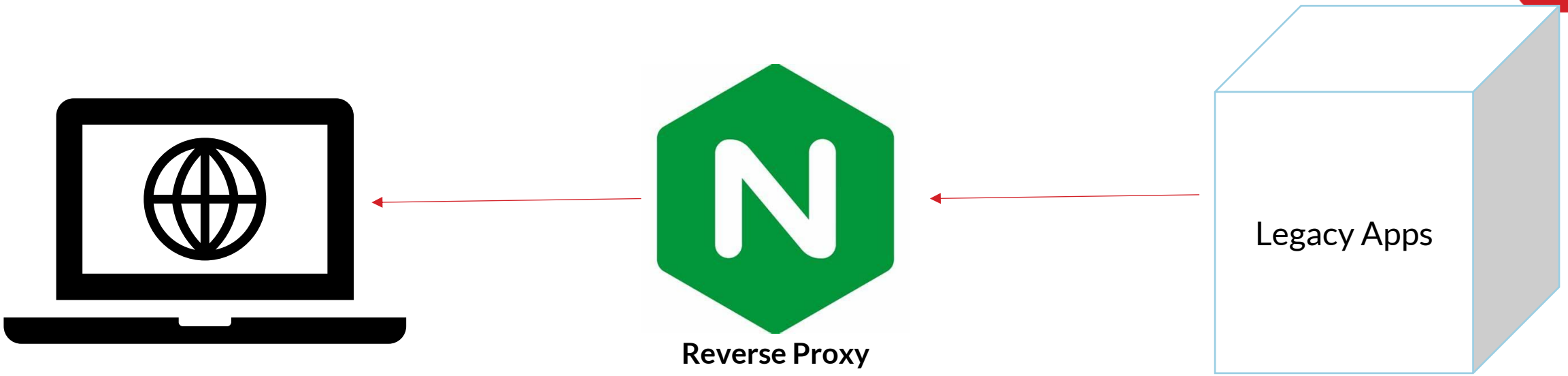
Help secure Express apps by setting HTTP response headers.

```
import helmet from "helmet";

const app = express();

app.use(helmet());
```

Helmet sets the following headers by default:

- `Content-Security-Policy` : A powerful allow-list of what can happen on your page which mitigates many attacks
- `Cross-Origin-Opener-Policy` : Helps process-isolate your page
- `Cross-Origin-Resource-Policy` : Blocks others from loading your resources cross-origin

What About Legacy Applications?



- Insert nonce placeholders in legacy apps (e.g. __NONCE__)
- Reverse proxy (during the response):
  - Generates random value (nonce)
  - Replaces __NONCE__ placeholder with random value

Demos

Key Takeaways

1. A checkbox for having a CSP, without actually analyzing its effectiveness, is almost as bad as not having a CSP
2. Simply using a reverse-proxy NONCE-replacement tactic is more effective than most CSPs in the study
3. Modern / new apps can use existing libraries such as helmet to do the heavy lifting

Jason@secureideas.com