Blue Triangle

The Ultimate Guide to Content Security Policy (CSP)

How to build, implement, and manage your CSP in 2019.

The Basics

A Content Security Policy (CSP) can help protect your site from data breaches caused by cross-site scripting (XSS) and formjacking attacks. A CSP also prevents client-side malware from injecting unwanted ads on your website.

This article will guide you through the process of building, implementing, and managing your Content Security Policy.

What is a Content Security Policy (CSP)?

A Content Security Policy (CSP) is a browser security standard that controls what domains, subdomains, and types of resources a browser is allowed to load on a given web page.

CSPs are implemented via an HTTP header (preferred delivery), but it can also be placed on a web page using a `<meta>` tag. CSPs are compatible with most modern desktop and mobile browsers, including Chrome, Firefox, Internet Explorer, Edge, Opera, and Safari.

Here's the first line of bluetriangle.com's CSP header:

```
Content-Security-Policy: default-src 'self' 'unsafe-inline' 'unsafe-eval' *.cloudfron
t.net *.btttag.com *.googleadservices.com *.googletraveladservices.com *.adroll.com *
.cloudfront.net everesttech.net
```

...and the `<meta>` tag:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self' 'unsafe-inline
' 'unsafe-eval' *.cloudfront.net *.btttag.com *.googleadservices.com *.googletravelad
services.com *.adroll.com *.cloudfront.net everesttech.net ... />
```

Here's how it works:

Why You Need a CSP

CSPs are used to detect and prevent certain types of attacks. The credit card skimming attacks at Ticketmaster UK, British Airways, Newegg, and thousands of other sites could have been prevented with a properly implemented CSP.

Below are some of the common attack vectors that a CSP can insulate you from.

Formjacking & Cross-Site Scripting (XSS)

If a hacker (e.g. Magecart) injects code into your checkout pages, a CSP automatically blocks the code from sending your customer's payment information to the hacker's domain.

Browser Hijacking and Ad Injection

Client-side malware and browser extensions cause unwanted ads to appear on your users' browsers. A CSP prevents these ads from loading when affected users go on your website.

Unauthorized Piggyback Tags

One tag could also be loading multiple tags from vendors you have not authorized. A CSP eliminates this security risk.

Blue Triangle

Build

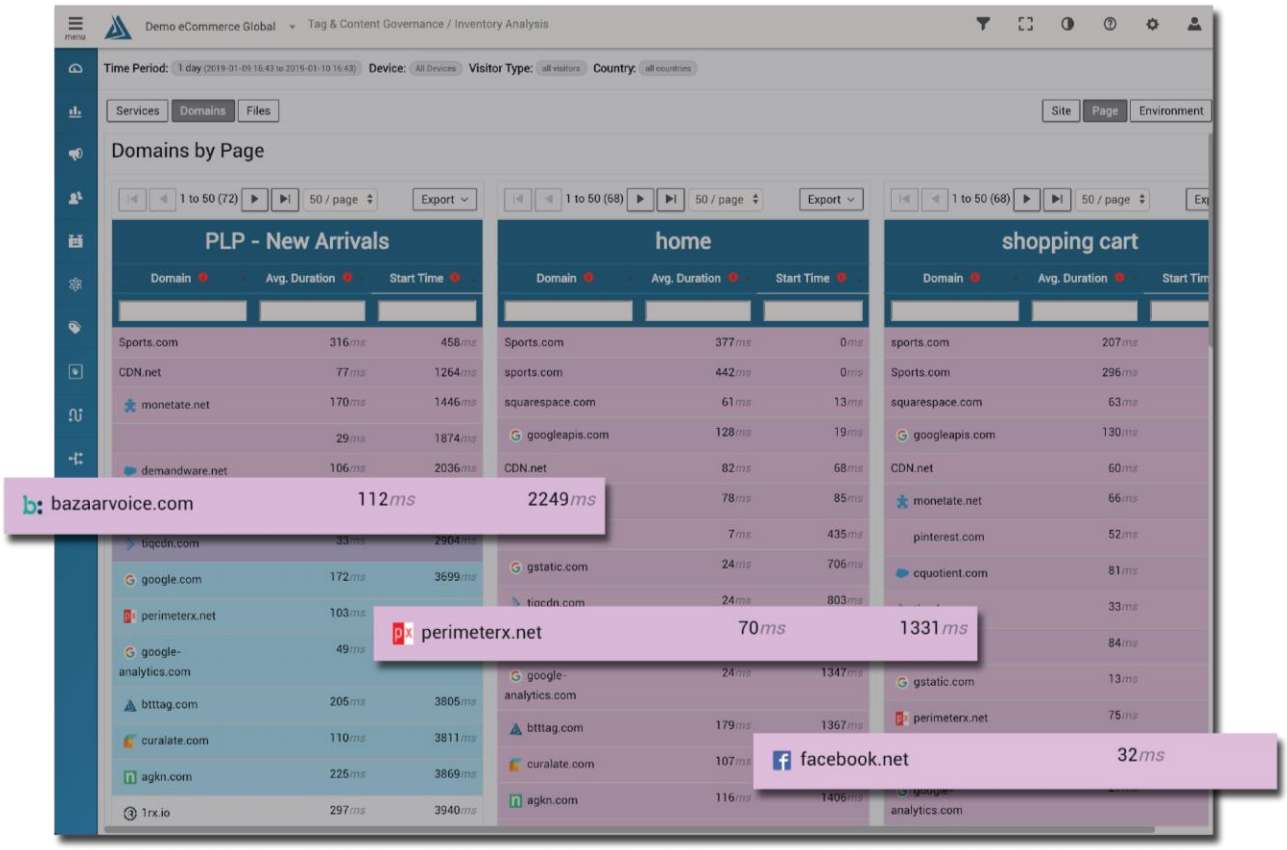
Building a CSP can seem like a daunting task if you don't know where to start. Luckily for you, we have your back. In this section, we're going to go through the basic checkpoints needed to build a CSP for your site.

Step 1

Inventory your first and third-party domains

Step one of the CSP journey is having a comprehensive understanding of who has access to your website, and what they're loading. Often-times the results are surprising! Outside of your expected first-party domains, there is likely a large swath of third-party domains living on your website. These domains serve a variety of purposes – from collecting data to displaying banner ads on your website. Inventorying all these domains for *every single page* of your site is critical to getting off on the right foot.

Every single page? We know this may seem overwhelming, but this process is important if you want to implement a stricter CSP on your more sensitive pages (checkout, account creation, etc.) as opposed to your home page or blog.

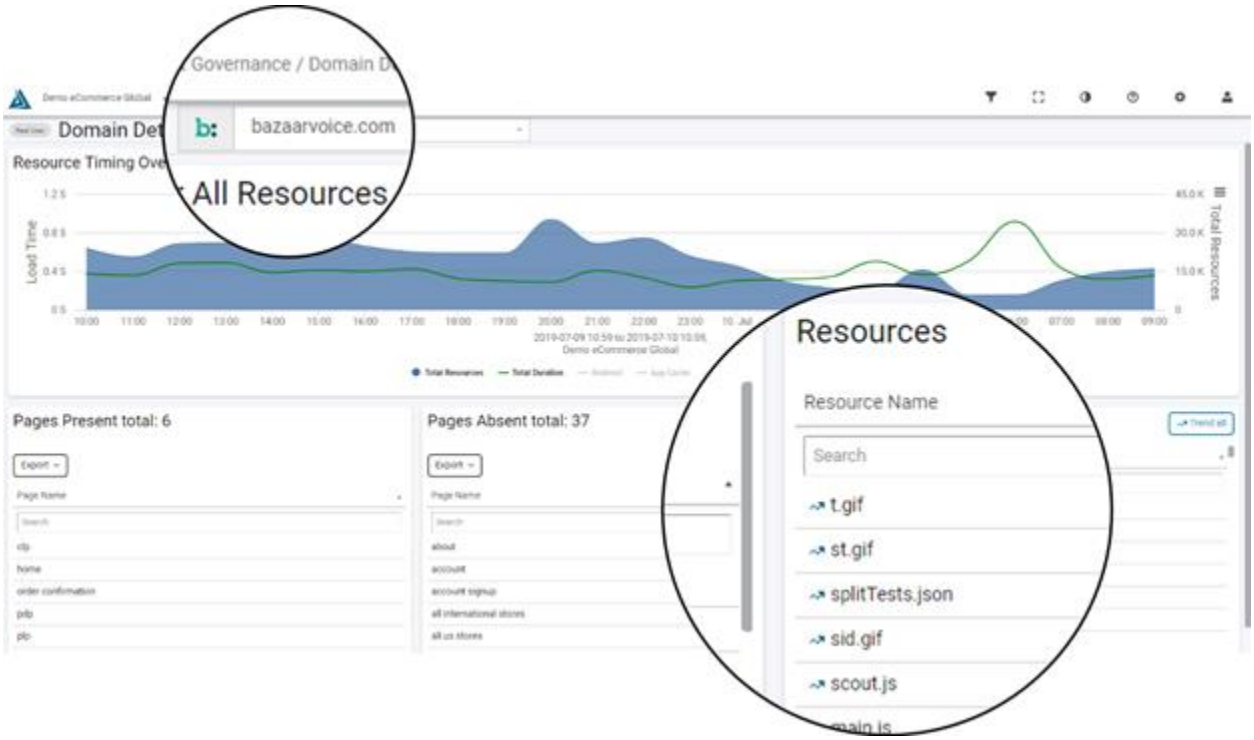


Blue Triangle

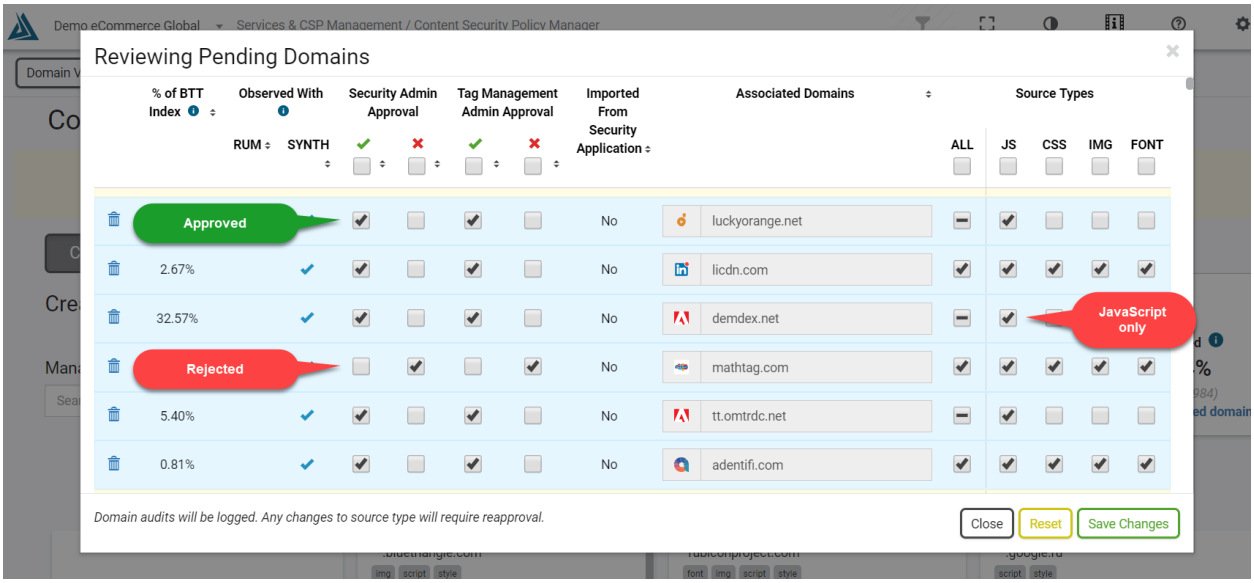
Step 2
Create the Whitelist

Now that you have taken inventory of all your site's domains, it is time to create a whitelist. Your whitelist is a compilation of the first and third-party domains that you would like to allow onto your site. Be sure to include the type of resources each of these domains is allowed to load. That will save you time in the build stage.

To visualize this process, we can use Blue Triangle's CSP Manager to identify every resource that the domain bazaarvoice.com is loading:



Then we can go through the approval process here:



Remember, you don't have to treat the whitelist as if it is the sole determinant for what is allowed on *all* pages on your site. Consider having a whitelist for the most sensitive areas of your site, and one for all others. Meaning, that the whitelist on sensitive pages such as check-out pages, should only allow what is completely necessary to the page's core functionality. Pages that don't collect personal information can have a much more relaxed, one-size-fits-most security policy.

Step 3
Building a CSP

Now that you've determined your whitelist(s) of approved domains, you can begin building your `Content-Security-Policy` Header(s) and/or `meta` tag(s). If you've already specified what resources each domain is allowed to load, you can properly utilize CSP directives.

CSP directives give you control over which domains can load specific types of resources (JavaScript, fonts, images, etc.).

For example, if you only want JavaScript to load from Google and AdRoll, you would add the `script-src` directive here:

```
Content-Security-Policy: script-src google.com adroll.com
```

But for the domains that do not have a directive assigned for them, the `default-src` directive is applied.

If you want to only allow JavaScript to load from Google and AdRoll, but want to allow Yahoo to load all resource types, your CSP would look like this:

```
Content-Security-Policy: default-src yahoo.com; script-src google.com adroll.com
```

Below is a list of the most common CSP directives.

Default-src	Default policy used in any case except when overridden by a more precise directive
Script-src	Policy dedicated to scripts
Object-src	Policy dedicated to plugins
Style-src	Policy dedicated to styles (CSS)
Img-src	Policy dedicated to images (img, but also url() or img()) from CSS, or link element related tp an image type.
Media-src	Policy dedicated to media (video, audio, source, track). This is a helpful directive to prevent video ad injections
Report-src	Allows you to define an URI to where CSP violation reports will be sent (important). If a piece of content is blocked by a browser, the browser will send a report with detailed information to this URI. Be careful, if your traffic is high, this could mean a lot of reports
Connect-src	Policy dedicated to connections from XMLHttpRequest object or a WebSocket. This directive may include allowing your landing page to appear after a consumer is curious about signing up for a free trial or a demo.

(Source: blog.dareboost.com)

Make sure you put a space between each domain within each directive. If any of your domains has a subdomain, you will want to denote that like `*.adobe.com`

Be careful to not make any typos and ensure every domain is accounted for.

Implement

Many website owners believe that implementing a CSP is difficult, time consuming, costly, and impossible to maintain. But with proper CSP implementation and the help of a CSP Manager, protecting your site is quick, simple, and effective.

Implementing Your CSP Header

Modern browsers (except IE) support the `Content-Security-Policy` HTTP header. This is the preferred delivery mechanism for a CSP.

When first implementing a CSP, it is recommended that you begin by adding the `Content-Security-Policy-Report-Only` HTTP header. This does not actively deny content from loading on your site. Instead, it alerts you of what domains and resources would be blocked by a fully enforced CSP.

Starting with a *report-only* CSP header lets you fine-tune your policy over a 1-2 week period. Since many third-party vendors cycle through various domains to send and receive data, it is important to catch and categorize them all during this time. Otherwise, you may encounter major issues with third-party functionality or even general site functionality.

Once you identify what domains you'd like to give access to your site, you can begin building your `Content-Security-Policy` HTTP header, which looks like this:

```
Content-Security-Policy: default-src 'self' *.adobe.com
```

Implementing Your CSP Meta Tag

You can also the implement your CSP directly in the HTML markup with a CSP `<meta>` tag.

Meta tag CSPs are implemented much in the same way as header CSPs. The meta tag needs to load in the `<head>` so you eliminate unwanted domains from accessing your site before the CSP kicks into action. Here's an example of a meta tag CSP:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self' 'un
safe-inline' 'unsafe-eval' *.cloudfront.net *.btttag.com *.googleadservice
s.com *.googletraveladservices.com *.adroll.com *.cloudfront.net everestte
ch.net ... />
```

Manage

Website audits and the management of a CSP can seem laborious and difficult if you have not already gone through the first two steps of a CSP (Building and Implementation). However, once you've implemented your CSP, it is important that you are alerted on violations and update your CSP after every code change. This is critical to staying secure and ensuring your site and third-parties function as intended.

Alerting on Violations

There are 2 directives you need to set in your CSP Header to be alerted on violations. The first is `report-uri`, which is now deprecated but some browsers still support it. The second is `report-to`, which is intended to replace `report-uri`, but is only supported by Chrome and Android Webview (as of July 2019). Until `report-uri` is fully adopted, you'll need to add

both directives to your CSP Header if you want to be alerted on violations. These directives are not supported with a CSP Meta-tag.

Now you need to specify where the violations will be sent to.

For the `report-uri` directive, you need to specify a URI where the report can POST to. Here's what it looks like:

```
Content-Security-Policy: ...; report-uri https:///cspendpoint.com
```

For the `report-to` directive, you need to add a header that specifies an endpoint for the violation reports. In the CSP itself, you'll need to specify the JSON field value that is referenced in the `report-to` header.

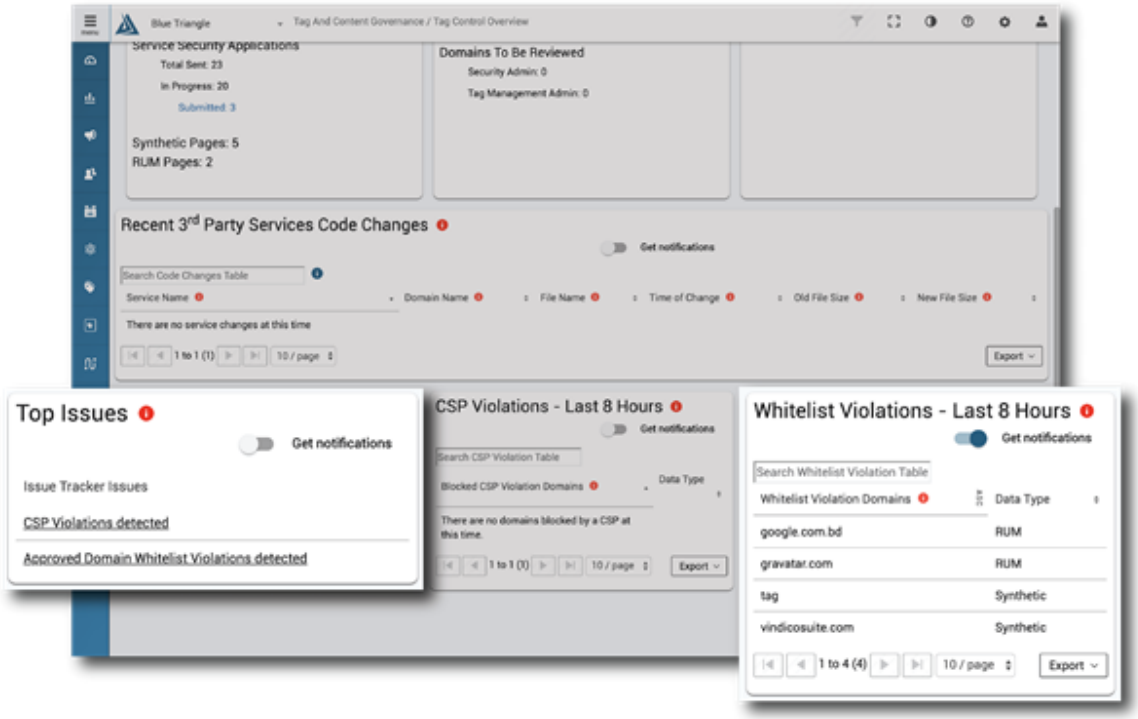
The `report-to` header:

```
Report-To: { "group": "endpoint-1",
             "max_age": 10886400,
             "endpoints": [
               { "url": "https://example.com/csp-reports", "priority": 1 },
               { "url": "https://backup.com/csp-reports", "priority": 2 }
             ] }
```

The CSP with report-to directive:

```
Content-Security-Policy: ...; report-to csp-endpoint
```

If you do not want to add these directives and assign an endpoint, you can use a CSP Manager that alerts you when violations occur. Here's what this looks like inside of Blue Triangle:



Blue Triangle

Blue Triangle's CSP Manager identifies if the violations are occurring from a clean-room environment (Synthetic) or from a real-user perspective (RUM = Real User Monitoring). That way you can identify if a violation is stemming from a browser extension or malware as opposed to a new third-party that was introduced to your site either from one of your own vendors or from a code change.

Updating Your CSP

Every time there is a code change, especially if a new technology is implemented, you need to update your CSP. This is critical to protect your site and ensure new technologies function as intended. Be sure to version your CSPs and include a date so you can keep track of releases.

Blue Triangle's CSP Manager

Ready to implement a CSP on your site? Have a CSP you need to manage? Blue Triangle's CSP Manager can help you secure your site quickly and efficiently. Don't be the next headline.