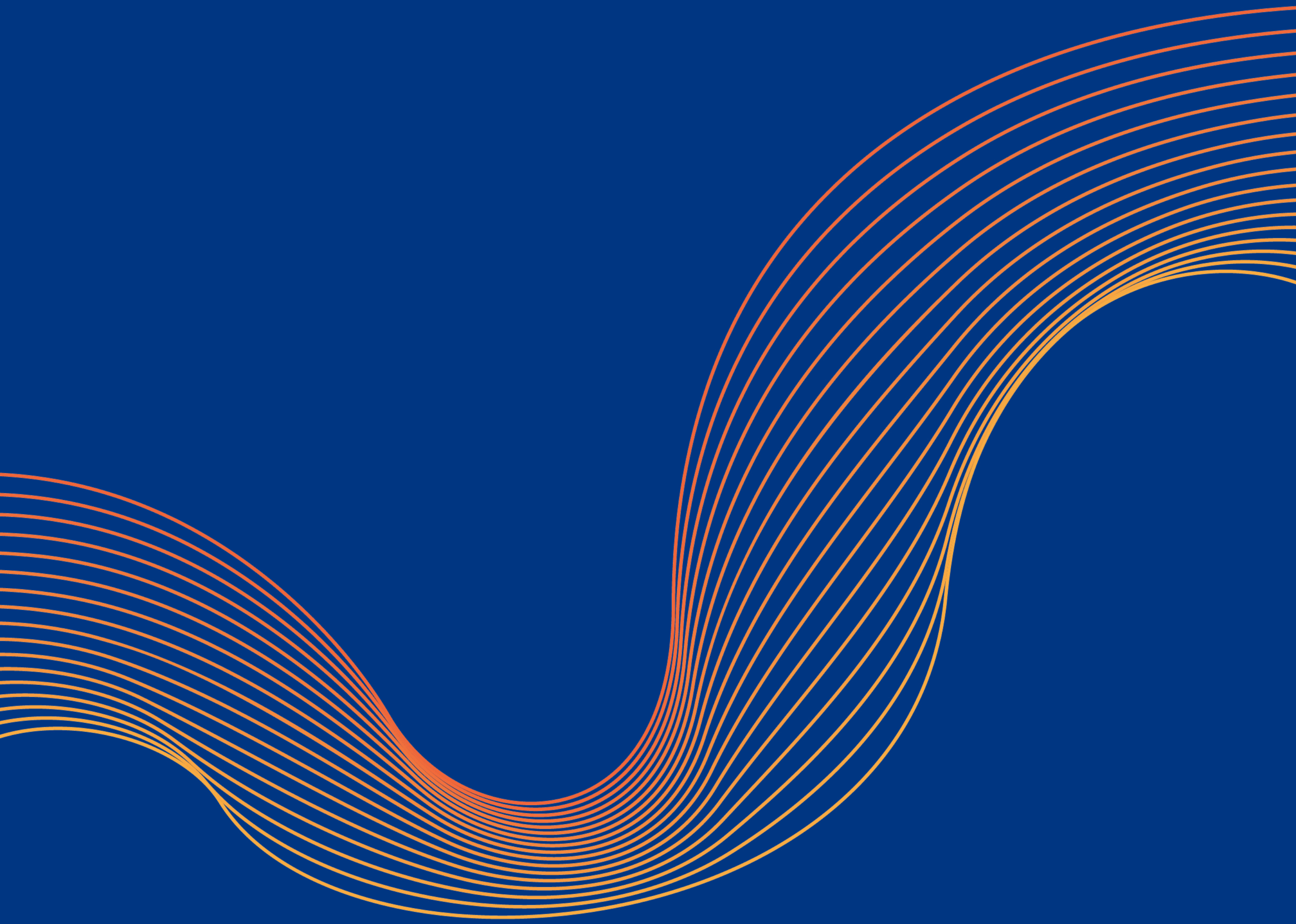
WHITEPAPER

CLOUDFLARE®

Keeping tabs on browser supply chain attacks



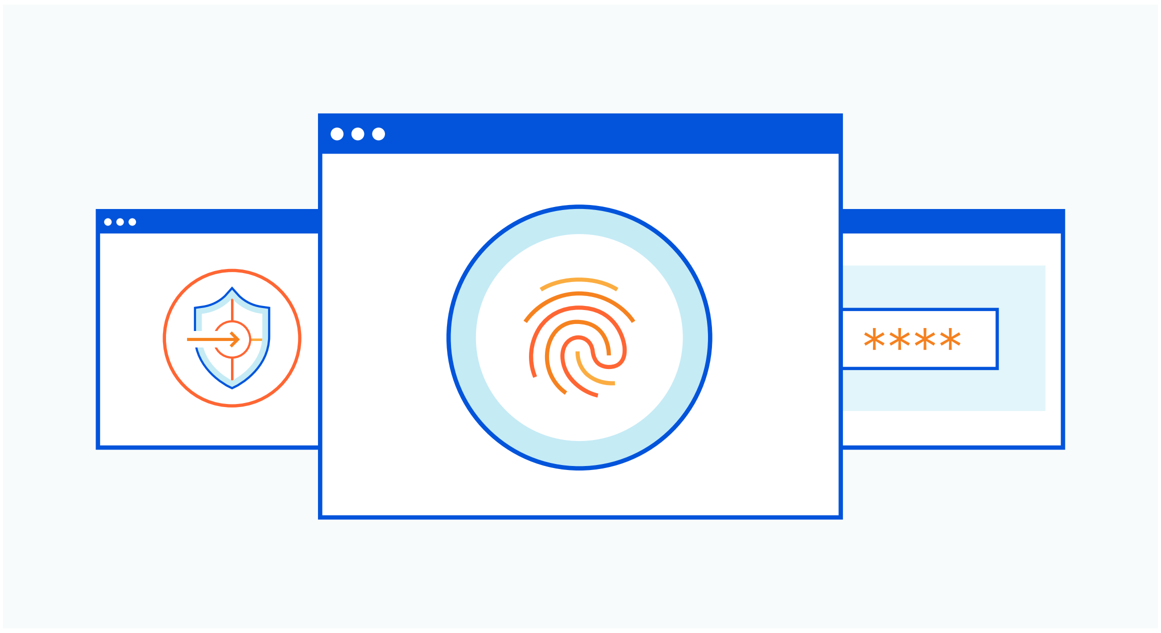
Client-side security challenges

Browsers are a website's window to the world. Whether site visitors are buying items on checkout screens, interacting with chat bots, or filling out forms to peruse site content - a large part of the end user experience is contingent on browsers being secure, fast, and private. Unfortunately, this is not always the case in practice.

Most if not all websites use dependencies - separate programs or pieces of code from third-party providers usually written in Javascript - to function as intended. While dependencies simplify software development and enhance site capabilities, they also raise a slew of security concerns because the dependencies are often loaded directly by the end-user's browser from third parties (i.e. they are loaded client-side).

Here are some reasons why client-side security is an important concern for security leaders:

- **Vast attack surface:** Around 45% of desktop requests were third-party requests in 2021. The median website uses 23 third-party solutions on desktop, and this number climbs to 91 third-party solutions for websites in the 90th percentile. Each of these dependencies is a security blind spot and a potential source of compromise.
- **Site scripts keep changing:** Cloudflare found that 55% of enterprise sites and applications load new scripts on a monthly basis. Larger applications use updated scripts even more frequently, sometimes every 48 hours (31%) or every day (13%).
- **An insecure slippery slope:** When sites add third-party code, it usually fetches even more code from remote sources that are unknown to the site owner. This further reduces visibility and control that site owners have over the end-user experience on browsers.
- **Diverging incentives:** Many third-party code snippets perform useful functions for business-facing teams (for example, capturing page visitor analytics, tracking ad conversions, or enabling push notifications). Since these third-party solutions aid in revenue generation, many teams understandably push for their increased adoption without spending a lot of time reflecting on security ramifications.



Why do we use dependencies?

Dependencies are programs or pieces of code from third-party providers that aid application functionality. Almost all websites use dependencies because they ease and speed up development, extend the impact of an application, and save time for developers by enabling them to reuse the work of third-party providers.

Here are some examples of dependencies you might find on applications:

- **Digital shopping carts:** Using dependencies can make any website an e-commerce website without needing to write tons of custom code.
- **Chat bots:** Dependencies can allow websites to run chatbot functionality, leading to better site navigation and time to value for website visitors.
- **Website analytics:** Possibly the most common dependencies are related to capturing site visitor analytics for application owners, helping them increase engagement and conversions.
- **Javascript libraries:** Not all dependencies are custom third-party code. They also include useful Javascript libraries that extend website functionality. For example, the Modernizr library detects features available in an end user's browser to support an optimal visitor experience.

The next section will introduce Magecart attacks, a pernicious form of supply chain attack that runs malicious code on end-user browsers by exploiting third-party dependencies.

45

Percent of desktop browser requests that were third-party requests in 2021

Source: HTTP Archive

23

Third-party solutions used by the median website in 2021

Source: HTTP Archive

55

Percent of enterprise sites and applications that load new scripts every month

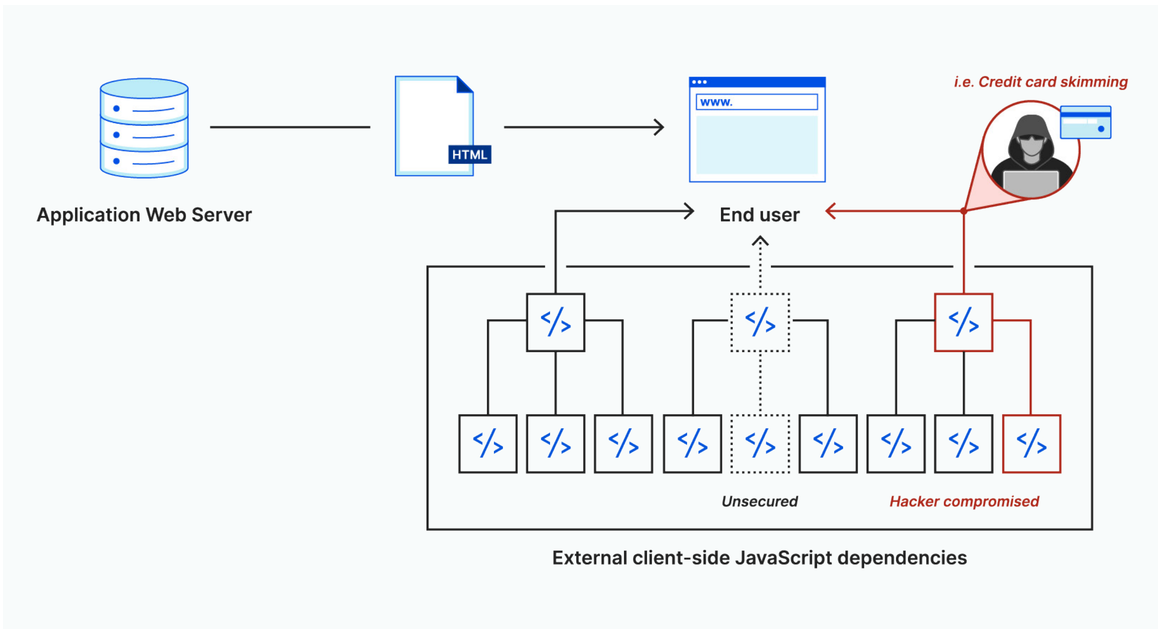
Source: Cloudflare

What is Magecart?

A Magecart-style attack is a type of software supply chain attack usually carried out in a user's browser. These attacks typically skim credit card information from website checkout forms (this is also known as formjacking) by compromising third-party dependencies used by the site. Attackers can use this data to enact fraud, sell the data on the black market, and other malicious purposes.

One intuitive way to think about these attacks is as analogs to physical card skimmers. Just like criminals hide devices within point-of-sale systems or ATMs to steal credit card details, they can also skim card details online by compromising third-party Javascript loaded in browsers.

The name Magecart was coined in 2015 and given to a loose affiliation of attack groups that employed online skimming techniques to target site visitors. These attacks have successfully targeted a large number of companies over the years such as British Airways, Ticketmaster, Twilio, NutriBullet, and Hannah Anderson. The next section will delve deeper into some of these examples.



HTML
www.
i.e. Credit card skimming
Application Web Server
End user
Unsecured
Hacker compromised
External client-side JavaScript dependencies

Fig: How a Magecart-style attack works

The impact of Magecart-style attacks can be severe and multi-pronged, including:

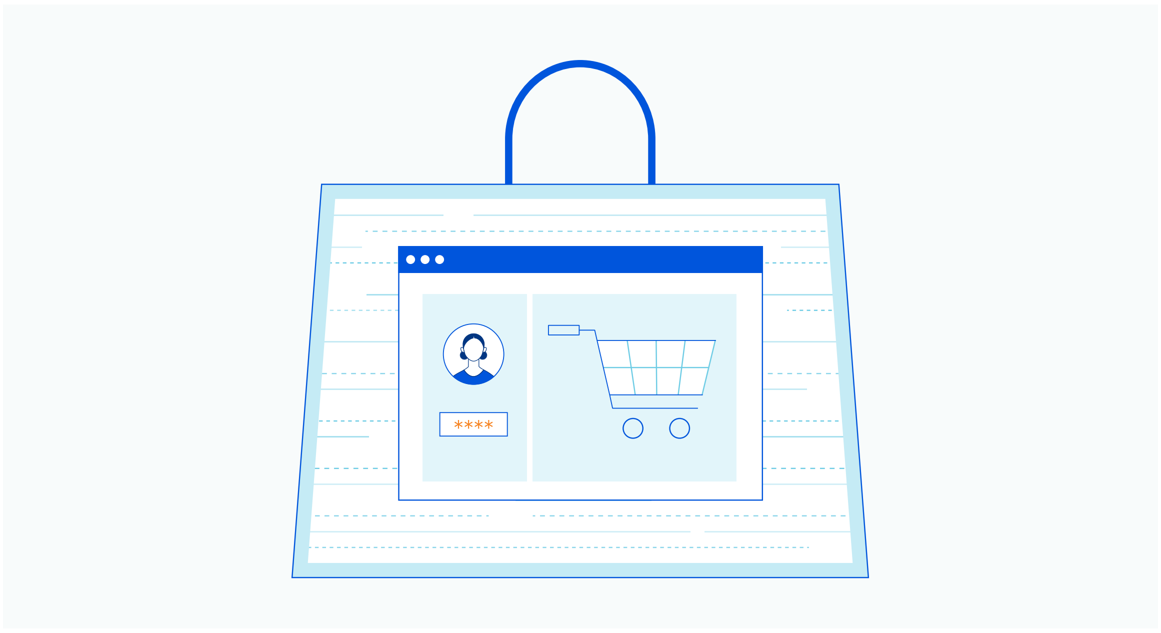
- **Compliance violations:** Successful Magecart-style attacks can open victim companies up to lawsuits, hefty fines if they are subject to regulations such as GDPR, as well as industry-specific penalties if they are regulated under PCI DSS.
- **Fraud:** In 2018, researchers found that attackers were essentially laundering funds from stolen credit cards through money mules and a reshipping company. By buying expensive goods online using stolen cards, enlisting mules to repackage and ship the goods to another location, and then selling the goods to split the proceeds, attackers made all parties in this supply chain unwitting participants in money laundering fraud.

- **Loss of revenue and customer trust:** Customers whose credit card details have been stolen might be unlikely to patronize the breached company's goods and services in the future, leading to revenue shortfalls that can especially hurt small and burgeoning businesses.
- **Relapse:** Research from 2019 found that an estimated one-fifth (20%) of websites hit by Magecart became reinfected within five days of addressing the initial attack. The labyrinthine nature of code loaded on browsers makes it easier for attackers to evolve their tactics and target different third-party dependencies even if the original attack is detected and blocked.

Although this whitepaper focuses on Magecart-style attacks, it's important to note that there are other forms of browser supply chain attacks as well - some common examples are cryptomining and adware.

Cautionary Magecart tales

Magecart-style attacks gained prominence in 2015 and have evolved continually since then, successfully targeting a host of companies in the process. The examples below explore different flavors of these attacks reported to the industry at large:



Magento

The first widely-known Magecart-style attack targeted Magento, a third-party shopping software, along with others such as OpenCart and Powerfront CMS. The name "Magecart" was born here as a portmanteau of "Magento" and "shopping cart". Although many years have passed since the initial attack, stores using outdated versions of Magento are still falling victim to Magecart-style attacks.

Ticketmaster

In June 2018, the UK division of ticket sales and distribution company Ticketmaster revealed that they had been compromised by a card-skimming campaign that successfully stole customer information. It was later discovered that this breach was part of a wide-ranging attack that targeted over 800 e-commerce sites around the world. Around 9.4 million customers were impacted by the Ticketmaster breach.

The attack was carried out by compromising scripts from Inbenta and SociaPlus, two third-party providers that served browser-side code on Ticketmaster websites. In 2020, Ticketmaster UK was levied a fine of $1.65 million by the Information Commissioner's Office (ICO) in the UK.

British Airways

In September 2018, British Airways shared that customer personal and payment information had been compromised, with potentially 380,000 customers affected. Further details highlighted this as one of the most sophisticated Magecart-style attacks on record.

The culprit was a 22-line script that was inserted in the Modernizr Javascript library (v2.6.2) using a highly targeted approach geared towards British Airways. The script ran on checkout pages on both the website and mobile app, skimming personal and payment information and sending it to an attacker-controlled server.

The server domain name was baways[.]com and was loaded with a paid SSL certificate, underscoring the effort attackers undertook to make the server (and by extension the code) seem legitimate.

Twilio

In July 2020, customer engagement platform company Twilio shared that bad actors had injected malicious code that made Twilio users load an extraneous URL on their browsers. A misconfigured AWS S3 bucket enabled attackers to compromise a Twilio Javascript library (v1.20 of the TaskRouter JS SDK) and launch a Magecart-style attack. The Twilio team replaced the bad version of the library and locked down permissions on the S3 bucket within an hour of initial detection.

This attack combined Magecart tactics with opportunistic scanning of open AWS S3 buckets for exploitation. This attack was not caused due to third-party script compromise, but by targeting Twilio's own infrastructure and libraries, showing that Magecart-style attacks come in many flavors.

Best practices to curb browser supply chain attacks

Existing browser protection technologies

Browser supply chain attacks have been in the public consciousness for a while. There are a few existing browser protection technologies that provide some reprieve against client-side threats, but they also have drawbacks that preclude complete protection.

Content Security Policy (CSP): CSP enables application owners to send an allowlist to the browser, preventing any resources outside those listed to execute. CSP is effective at curbing certain cross-site scripting attacks (XSS), especially when entirely new and malicious scripts are inserted into the workflow.

However, implementing CSP has certain operational and detection drawbacks:

- CSP fails to recognize when existing resources change from benign to malicious states. As the Magecart examples in the previous section highlighted, this drawback would enable attackers to get past CSP allowlists by inserting malicious scripts into existing, permitted resources and libraries.
- CSP requires developers to maintain and update the allowlist every time a new script is added to the site. Moreover, CSP increases the risk of the website breaking or not loading properly, which leads to overly broad CSPs being written that defeat the initial purpose.

Subresource Integrity (SRI): SRI enables application owners to specify an expected file hash for Javascript and other resources. If the fetched file doesn't match the hash, it's blocked from executing.

However, implementing SRI has the following challenges:

- Vendors update their code often and sometimes serve different files to different end users, which renders the SRI protection mechanism redundant.
- Javascript vendors sometimes serve versioned files with different hashes to end users due to minor differences such as spacing. This could result in SRI blocking legitimate files, leading to negative end user experience without any fault of the application owner.

Recommendations to curb browser supply chain attacks

Below are some best practices that will enable organizations to move towards complete protection against Magecart and Javascript supply chain attacks.

Get visibility into third-party scripts: You can't protect what you can't see, which is especially true for third-party code that executes on end user browsers. Prepare an inventory of all third-party Javascript code on your website to have a better idea of the potential attack surface.

Monitor for new and updated scripts: Look for solutions that track Javascript dependencies and third-party scripts over time to recognize when new dependencies appear or existing code is changed. This gives developers the insight they need to determine whether a new or updated script is expected or suspicious.

Leverage threat intelligence and machine learning: Utilize either in-built or integrated threat intelligence to get alerted whenever Javascript is being served from known malicious URLs. Look for solutions using machine learning or similar algorithms to identify malicious dependency behavior. Choose solutions that offer massive scale, since this will increase the breadth of available threat intelligence, the data available to improve detection algorithms, and the speed of remediation.

Seek integrated solution for easier management: Although this whitepaper covers browser supply chain attacks, it's evident that application owners need to be cognizant of many more attack vectors. Rather than implementing piecemeal solutions, look for a complete and integrated platform that protects applications against a wide range of attacks (e.g. DDoS, botnets, API abuse) and that can be monitored from a single console.

Introducing Cloudflare Page Shield

Cloudflare Page Shield is a client-side security product that protects site visitors from Magecart-style attacks and other browser supply chain threats.

Rather than having application owners manually create and maintain allowlists, Page Shield starts monitoring for Javascript dependencies with the click of a button. Going beyond CSP, Page Shield uses both threat feed integrations and machine learning classification to detect Magecart-style attacks and other elusive client-side threats.

Just like other Cloudflare services, Page Shield runs on every server in every Cloudflare data center on our global edge network, lending unprecedented line of sight into visibility and speed of response.

Visibility

Monitor third-party scripts: Through a feature called Script Monitor, Page Shield continually scans for Javascript dependencies and alerts application owners when new dependencies appear. Application owners can receive timely alerts to investigate new scripts and determine whether changes are expected or suspicious.

Script Monitor works by adding a Content-Security-Policy-Report-Only header to pages as they pass through Cloudflare's edge. When JavaScript files attempt to execute on the page, browsers will send a report back to Cloudflare. Using a report-only header removes requirements for application owners to maintain allowlists for relevant insights.

Identify and act upon script changes: Page Shield detects whenever Javascript used on the site exhibits a meaningful change. Application owners can receive alerts on Javascript file changes and review logs of the last five changes the script underwent, providing them with necessary insight to decide if the change is expected or malicious.

Detection

Detect known malicious client-side Javascript: Page Shield leverages hostname and URL based threat intelligence to detect when malicious third-party Javascript is being served on end user browsers. Application owners can receive alerts when Javascript used by the site is hosted under a malicious hostname, as well as when specific Javascript files loaded by the site are categorized as malicious.

Spot elusive browser supply chain attacks: Page Shield utilizes machine learning classification to detect new scripts whose behavior is similar to scripts used in Magecart-style attacks. Similar to the bot score provided by Cloudflare Bot Management, Page Shield provides a 0-99 score as an abstracted output of multiple machine learning algorithms. Application owners can receive a high signal alert whenever a Javascript file loaded by the application is classified as malicious.

Page Shield + Zaraz: security and speed

Cloudflare Zaraz is built on our own Cloudflare Workers platform. Zaraz is a natural complement to Page Shield, enabling customers to improve their website's speed and security while reducing third-party bloat.

What is Zaraz?

Zaraz is a third-party manager built for speed, privacy, and security. Zaraz identifies third-party scripts and loads them in Cloudflare's edge network instead of in end user browsers. Advantages include:

- **Improved website speed** by taking the burden of loading third-party scripts off end user browsers. Websites are found to be 40% faster after using Zaraz.
- **Enhanced core web vitals** to improve page rankings and SEO performance.
- **Uphold privacy requirements** by providing application owners visibility and control over data accessed by third-party solutions.
- **Reduce the attack surface** by utilizing integrations with third-party tools to load them in the cloud instead of in browsers.

Page Shield and Zaraz
Organizations looking for complete client-side protection should use both Page Shield and Zaraz. While Zaraz reduces the attack surface by loading third-party tools on the cloud, it does not integrate with all third-party tools yet. It's highly likely that organizations using Zaraz will still have third-party scripts running in browsers. Enabling Page Shield alongside Zaraz provides visibility into the entire attack surface and protects users from browser supply chain attacks while also addressing website performance and privacy concerns.

The table below highlights common use cases for both Page Shield and Zaraz.

	Page Shield	Zaraz
Use cases	**Get visibility into all third-party scripts** by tracking new and changed dependencies.	**Reduce the browser attack surface** by loading integrated (not all) third-party tools on Cloudflare's edge.
	Detect known malicious Javascript by getting alerted on Javascript served from malicious hostnames and URLs.	**Improve website performance** by unburdening browser resources from loading third-party scripts.
	Detect elusive browser supply chain attacks by utilizing machine learning insights.	**Ensure privacy compliance** by providing control over what data is accessed by third-party tools.

Tightly integrated for complete protection	Configurable from one dashboard	Deployed on Cloudflare's fast global edge network

REV:BDES-2953.2022FEB24