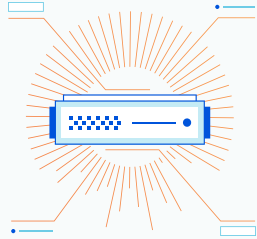

Common browser isolation challenges, and how to overcome them

The intersection of Internet browsing and Zero Trust security

INDEX

Introduction	3
Challenges of common browser isolation strategies	3
Streaming screen captures of browsing activity from the cloud	4
Breaking down websites in the cloud and stripping out malicious code	5
Isolating browsing activity in an on-device virtual machine	6
These immediate challenges create additional ones	7
A better approach to remote browser isolation	7
How Cloudflare makes remote browsing cost-effective and less disruptive of end-user	8
Network Vector Rendering delivers better end-user experiences and closes security gaps an	9
Learn more and get started	9



Introduction

IT and security teams have good reason not to trust the public Internet. Phishing and malware accounted for 39% of all data breaches in 2020, [a Verizon report found](#). And a [Forrester Consulting study commissioned by Cloudflare](#) found that in 2020, 61% of companies with more than 1000 employees experienced an increase in phishing attacks compared to previous years.

Every IT and security professional wants to keep their organization from being included in these statistics. Specifically, they want to:

- **Block malware and phishing**, which constantly shift tactics to evade detection.
- **Stop data loss in general**, whether via infected devices or user interactions.
- **Get better visibility into employee Internet browsing**, in order to understand their organization's specific threat landscape and respond faster to breaches that do occur.

These use cases are important elements of **Zero Trust security**, in which Internet properties and code should not be implicitly trusted — and must therefore be securely processed at the moment of user interaction.

In order to accomplish these goals, some organizations are turning to browser isolation, in which employee Internet browsing is kept separate from local networks and infrastructure. Implemented thoroughly and efficiently, browser isolation shows potential to be the single most powerful way to mitigate attacks coming from the Internet. Unfortunately, it has so far remained a niche technology because of a number of challenges — including **high costs, poor browsing experiences, logistical management hurdles, and security gaps**.

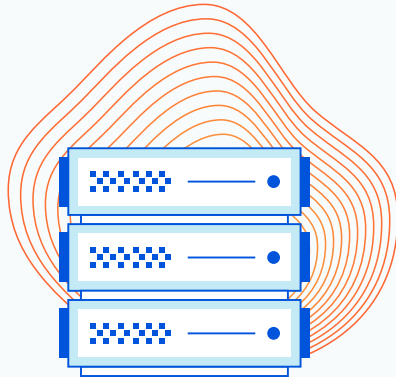
This paper describes those challenges in detail in order to help security and IT teams better understand their Internet browsing security needs. It also describes a method for overcoming these challenges — and, finally, explains how Cloudflare has integrated this method into its global network.

Challenges of common browser isolation strategies

Malware, phishing, and data loss affect organizations in every industry and field. Browser isolation is most commonly deployed as a solution to the first two threat types, bolstering the blacklist, file-matching and behavioral approaches used by secure web gateways.

In practice, though, browser isolation often falls short of this goal.

Why is this the case? Consider the limitations of the most common browser isolation methods:



Streaming screen captures of browsing activity from the cloud

This approach, which is sometimes called ‘pixel-pushing,’ captures events in the end-user’s browser and transmits them to a cloud-hosted remote browser, which actually performs browsing actions and transmits a sequence of pixel images of the remote browser window back to the end-user’s. In this way, any malicious code — whether it is downloaded automatically or by a user’s deliberate action — is kept separate from the end-user’s device. And potential phishing sites — such as pages with username/password form fields — may be displayed in a read-only fashion or with a warning message added.

This approach does a good job of isolating endpoint devices from malware and phishing. However, it presents problems like:



End-user latency: When remote browser isolation is hosted in the public cloud — or on a geographically limited private network — end-users may experience latency when they are physically distant from browser isolation data centers. This problem compounds when end-user traffic passes through other security tools — such as a secure web gateway — that are not hosted in the same data centers, or that require multiple ‘passes’ through inefficiently-architected containers.



High costs: Continuously encoding video streams of remote webpages to end-user endpoint devices is very costly from a computational perspective. It also requires significant bandwidth, even when highly optimized. These costs are typically passed along to customers.



Security gaps: Since ‘pixel-pushing’ often causes poor end-user experiences, many organizations only require its use in teams with access to especially sensitive data — such as finance, human resources, or company executives. The organization may also apply remote browsing only to a small percentage of web pages seen as especially risky. Either way, the organization will remain exposed — whether via unprotected employees, or via ‘trustworthy’ sites that have become compromised.



High bandwidth needs: Image streaming is bandwidth-hungry, which can overburden network infrastructure and negatively impact end-user experience. In addition, pixel density increases exponentially with resolution, which means remote browser sessions (particularly fonts) on HiDPI devices can appear fuzzy or out of focus.

Breaking down websites in the cloud and stripping out malicious code

This method is often referred to as DOM manipulation. In frontend development, the DOM, or Document Object Model, is the data representation of the objects that comprise the structure and content of a webpage. In DOM manipulation, a cloud-hosted remote browser examines a webpage's HTML, CSS, and other elements, and attempts to eliminate active code such as Javascript, known exploits, and other potentially malicious content. The remote browser then forwards this code to the end-user's browser, which uses it to reconstruct a 'clean' version of the page. In addition, as in "pixel-pushing," DOM manipulation may also flag certain pages as phishing risks.

Since DOM manipulation only transfers website code, rather than a full stream of the browsing experience, it requires less bandwidth and can lead to faster end-user experiences.

However, it presents problems like:



End-user latency: As with 'pixel-pushing,' if DOM manipulation browser isolation operates in the public cloud — or on a geographically limited private network — end-users may still experience latency when origin servers are too far away, or when browser isolation and other security tools are hosted in different data centers.



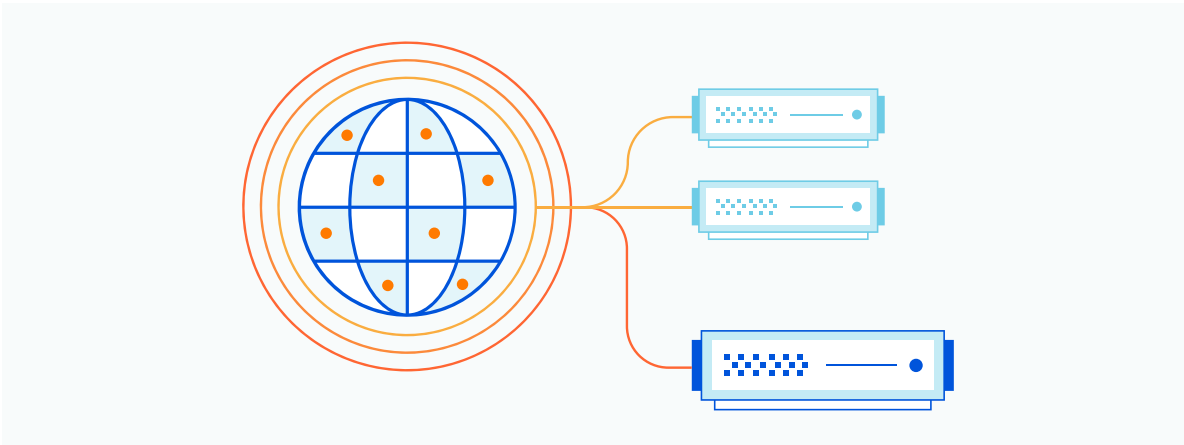
Breaking website experiences: Inevitably, attempting to remove malicious active code — as well as reconstructing HTML and CSS, and re-building uncommon site architectures — results in broken pages that do not render properly or at all. Furthermore, websites that work today may not work tomorrow, since site publishers may make daily changes that break DOM reconstruction functionality. DOM manipulation even struggles to support common enterprise-wide services like Google G Suite or Microsoft Office 365. The result is an infinite tail of issues requiring significant IT resources in an endless game of whack-a-mole.



High costs: Some DOM manipulation services are hosted in third-party public cloud infrastructure, which incurs additional costs that are typically passed along to customers. Either way, the organization will remain exposed — whether via unprotected employees, or via 'trustworthy' sites that have become compromised



Security gaps: As with 'pixel-pushing,' DOM manipulation's unreliable end-user experience often means organizations only use it sporadically. In addition, though DOM manipulation is a form of browser isolation, it still sends untrusted third-party code to endpoint devices. If the service fails to identify malicious code — a constant risk given the ever-evolving threat landscape — endpoint devices could still succumb.



Isolating browsing activity in an on-device virtual machine

In this approach, software installed on an endpoint device creates a virtual machine that is isolated from the rest of the device's operating system. All browsing activity takes place on this virtual machine, so any downloaded malware cannot infect the rest of the device. In addition, like other approaches, the software may flag certain pages as phishing risks.

Unfortunately, this approach's problems include:



High CPU and RAM demands: Running a separate virtual machine can slow down many personal computers, resulting in slow browsing experiences for end-users.



Endpoint management difficulties: Browser isolation via local software requires IT teams to install and update said software on every endpoint device — a tall order for large organizations. This logistical complexity gets even worse when organizations have large numbers of remote workers, or if they employ third-party contractors who do not use corporate-issued devices.



Mobile compatibility issues: On-device virtual machine-based browser isolation requires operating-system specific implementations. Mobile devices are often unsupported.



Isolation failures: Local browser isolation services periodically experience vulnerabilities that allow malicious code to access the main operating system. In these circumstances, IT teams or end-users must manually install patches and updates. But if the patch isn't installed properly, the endpoint device may remain vulnerable.



Poor end-user experience: Virtual machine based implementations often require end-users to use separate "virtualized" browsers, windows or desktops. This requires training and creates IT support burdens.

These immediate challenges create additional ones

Immediate consequences like high costs, poor end-user experiences, and endpoint management difficulties are not the only challenges of common browser isolation approaches.







Another is the long-term consequences of device compromise. When the end-users log into sensitive applications via a compromised device, malware may access that application's data without the user being aware.

In addition, all of these browser isolation approaches struggle to prevent certain types of data loss. Internal actors may still be able to upload and send sensitive information via email, or to enter it into online forms.

A better approach to remote browser isolation

Organizations still want to adopt remote browsing despite the problems mentioned in the previous section.

Fortunately, certain technologies can help:

Problem	Solution
 Latency	Use of a large edge network: Rather than hosting browser isolation in a limited number of public cloud data centers, do so on a global edge network that is close to end-users anywhere. Furthermore, use browser isolation software that lives in the same data centers as secure web gateways and other security tools.
 High bandwidth requirements	No pixel pushing: Streaming images of remote browser activity is impractical to implement for larger enterprises, both from cost and user experience perspectives.
 Breaking website experiences	Use native browser technology: Remote browsers that use technology already built into common endpoint device browsing apps are more reliable at reconstructing all kinds of sites accurately.
 High compute costs	Next-generation cloud computing: Avoid remote browser isolation hosted in the public cloud. And use efficient serverless computing techniques that improve on virtualization and containerization by eliminating the orchestration and management of underlying server resources, in order to use those resources more effectively.
 Security gaps	Use native browser technology: Rather than trying to decide which code to send or block, native browsing technology can avoid sending code altogether. Rather, it is able to only send the last step in the rendering process that draws the page.
 Endpoint-specific difficulties	No local browser isolation: Isolating browsing activity on endpoint devices is too slow and difficult to manage, making the approach entirely outdated.

To see how these technologies work in practice, consider the example of Cloudflare Browser Isolation, which is natively-integrated with our Zero Trust platform.

How Cloudflare makes remote browsing cost-effective and less disruptive of end-user

At a high level, Cloudflare avoids endpoint-specific difficulties simply by running on a global edge network rather than endpoint devices. More specifically, Cloudflare's technology removes the other problems as well:

Problem	Solution	Cloudflare Implementation
 Latency	Use of a large edge network	Remote browser isolation takes place in every data center in Cloudflare's edge network, which spans over 200 cities in 100 countries and sits within 100 milliseconds of 95% of the world's internet connected population. This same infrastructure delivers ultra-low latency global DNS and CDN services. In addition, our remote browsing seamlessly interacts with our forward proxy to apply all other filters (such as blocking part of a page) and inspections without requiring multiple passes and hopping between disparate point solutions.
 High bandwidth requirements	Native browser technology	Cloudflare's Network Vector Rendering technology (learn more below) transmits draw commands rather than pixel images or "scrubbed" code. This method requires a fraction of the bandwidth consumed by normal browsing or DOM manipulation — let alone 'pixel-pushing.'
 Breaking website experiences	Native browser technology	Cloudflare's remote browser is based on the open-source Chromium engine, on which Google Chrome and twenty-one other common browsers are built. Significant ongoing investment in the Chromium engine ensures the highest levels of website compatibility. In addition, since Cloudflare's Network Vector Rendering technology transmits draw commands rather than "cleaned" code, it ensures that even the most complex web pages don't break.
 High compute costs	Next-generation cloud computing	Since Cloudflare browser isolation operates on our own network, we do not have to pass along public cloud costs to customers. And since we are able to orchestrate and manage server resources very efficiently, we avoid the seconds-long cold starts that frequently affect applications hosted on the public cloud.
 Security gaps	Native browser technology	By transmitting lightweight vector draw commands — rather than any original website code — Cloudflare eliminates the risk of untrusted code running on the endpoint device. Undetected malware only compromises the remote browser without affecting the endpoint. And since Cloudflare provides strong end-user experiences, companies can apply remote browsing to less-risky use cases that may otherwise go unprotected.
	Granular control of end-user behavior	Data loss prevention tools typically protect data while it is in transit over the network, either by allowing or blocking transmission. Cloudflare Browser Isolation will give administrators granular control over: <ul style="list-style-type: none"> • Copy/paste/print permissions • Upload/download actions • Keyboard activity in general • Form input permissions • Where downloaded files are stored* *Coming soon

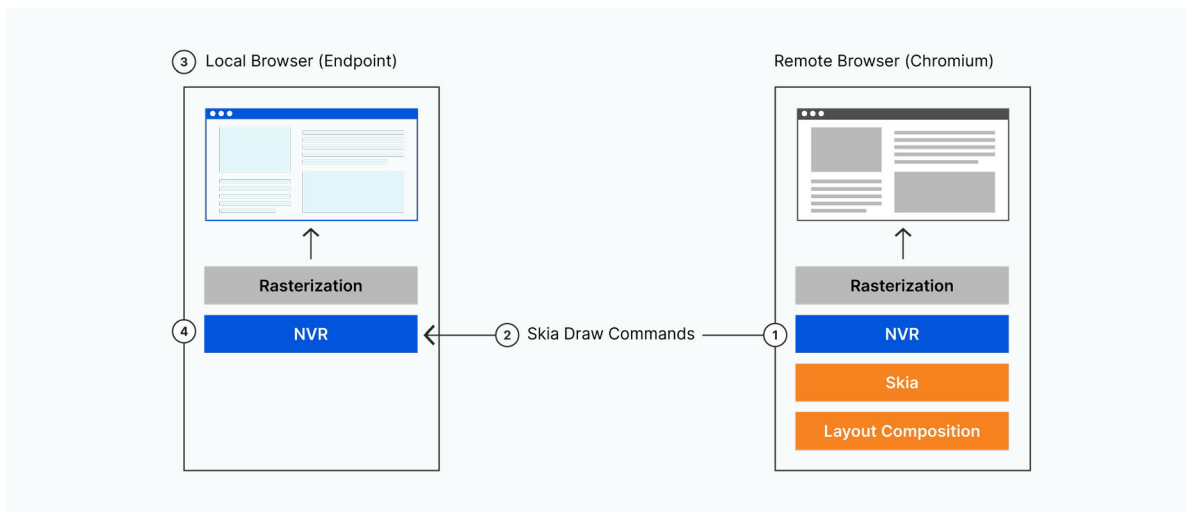
Read on for a deeper dive into the NVR technique described above:

Network Vector Rendering delivers better end-user experiences and closes security gaps

As mentioned above, Cloudflare’s remote browser is based on Chromium. A key architectural feature of the Chromium browser is its use of [Skia](#) — a widely-used cross-platform graphics engine for Android, Google Chrome, Chrome OS, Mozilla Firefox, and many other browsers. All HTML5-compliant browsers can render Skia. Everything visible in a Chromium browser window is rendered through the Skia rendering layer. This includes application window UI such as menus — but more importantly, the entire contents of the webpage window are rendered through Skia. Cloudflare can even isolate downloaded files and move them to various locations per end-user needs.

Cloudflare’s Network Vector Rendering (NVR) technology intercepts the remote Chromium browser’s Skia draw commands, tokenizes and compresses them, then encrypts and transmits them across the wire to any HTML5 compliant web browser running locally on the endpoint desktop or mobile device. The Skia API commands captured by NVR are pre-rasterization which means they are highly compact. And since Skia is so widespread, Cloudflare’s remote browsing works on any modern web browser.

Network Vector Rendering is also more secure. As mentioned above, since Cloudflare delivers draw commands rather than actual website code to endpoint devices, the underlying data transport is not an attack vector.



Learn more and get started

From the day Cloudflare started, our mission has been to help build a better Internet and democratise the technologies that were only previously accessible to large companies with sophisticated networks, dedicated IT teams, and massive budgets. Better remote browser isolation is an important part of that mission.

By making browser isolation more cost effective — and by providing exceptional end-user experiences — we hope more organizations will be able to experience the technology’s true value. Like a not-too-distant past when HTTPS encryption was reserved for “sensitive” login pages and ecommerce checkouts, we believe that trusting arbitrary website code will seem just as archaic as creating the new paradigm of Zero Trust browsing.

To learn more about Cloudflare Browser Isolation, and how it can help you achieve Zero Trust browsing, visit <https://www.cloudflare.com/products/zero-trust/browser-isolation/>



© 2022 Cloudflare Inc. All rights reserved. The Cloudflare logo is a trademark of Cloudflare. All other company and product names may be trademarks of the respective companies with which they are associated.