Cloudflare + Remote Browser Isolation

01/07/2020

Darren Remington

This post is also available in □□□□, □□□, Deutsch, Français and Español, .

Cloudflare announced today that it has purchased S2 Systems Corporation, a Seattle-area startup that has built an innovative remote browser isolation solution unlike any other currently in the market. The majority of endpoint compromises involve web browsers — by putting space between users' devices and where web code executes, browser isolation makes endpoints substantially more secure. In this blog post, I'll discuss what browser isolation is, why it is important, how the S2 Systems cloud browser works, and how it fits with Cloudflare's mission to help build a better Internet.

What's wrong with web browsing?

It's been more than 30 years since Tim Berners-Lee wrote the project proposal defining the technology underlying what we now call the world wide web. What Berners-Lee envisioned as being useful for "*several thousand people, many of them very creative, all working toward common goals*"[1] has grown to become a fundamental part of commerce, business, the global economy, and an integral part of society used by more than 58% of the world's population[2].

The world wide web and web browsers have unequivocally become the platform for much of the productive work (and play) people do every day. However, as the pervasiveness of the web grew, so did opportunities for bad actors. Hardly a day passes without a major new cybersecurity breach in the news. Several

contributing factors have helped propel cybercrime to unprecedented levels: the commercialization of hacking tools, the emergence of malware-as-a-service, the presence of well-financed nation states and organized crime, and the development of cryptocurrencies which enable malicious actors of all stripes to anonymously monetize their activities.

The vast majority of security breaches originate from the web. This should not be surprising. Although modern web browsers are remarkable, many fundamental architectural decisions were made in the 1990's before concepts like *security*, *privacy*, *corporate oversight*, and *compliance* were issues or even considerations. Core web browsing functionality (including the entire underlying WWW architecture) was designed and built for a different era and circumstances.

In today's world, several web browsing assumptions are outdated or even dangerous. Web browsers and the underlying server technologies encompass an extensive – and growing – list of complex interrelated technologies. These technologies are constantly in flux, driven by vibrant open source communities, content publishers, search engines, advertisers, and competition between browser companies. As a result of this underlying complexity, web browsers have become primary attack vectors.

The very structure and underlying technologies that power the web are inherently difficult to secure. Some browser vulnerabilities result from *illegitimate use of legitimate functionality*: enabling browsers to download files and documents is good, but allowing downloading of files infected with malware is bad; dynamic loading of content across multiple sites within a single webpage is good, but cross-site scripting is bad; enabling an extensive advertising ecosystem is good, but the inability to detect hijacked links or malicious redirects to malware or phishing sites is bad; etc.

Enterprise Browsing Issues

Enterprises have additional challenges with traditional browsers.

Paradoxically, IT departments have the least amount of control over the most ubiquitous app in the enterprise – the web browser. The most common complaints about web browsers from enterprise security and IT professionals are:

1. **Security** (obviously). The public internet is a constant source of security breaches and the problem is growing given an 11x escalation in attacks since 2016 (Meeker[3]). Costs of detection and remediation are escalating and the reputational damage and financial losses for breaches can be substantial.
2. **Control**. IT departments have little visibility into user activity and limited ability to leverage content disarm and reconstruction (CDR) and data loss prevention (DLP) mechanisms including when, where, or who downloaded/upload files.
3. **Compliance**. The inability to control data and activity across geographies or capture required audit telemetry to meet increasingly strict regulatory requirements. This results in significant exposure to penalties and fines.

Given vulnerabilities exposed through everyday user activities such as email and web browsing, some organizations attempt to restrict these activities. As both are legitimate and critical business functions, efforts to limit or curtail web browser use inevitably fail or have a substantive negative impact on business productivity and employee morale.

Current approaches to mitigating security issues inherent in browsing the web are largely based on *signature* technology for data files and executables, and lists of known good/bad URLs and DNS addresses. The challenge with these approaches is the difficulty of keeping current with known attacks (file signatures, URLs and DNS addresses) and their inherent vulnerability to zero-day attacks. Hackers have devised automated tools to defeat signature-based approaches (e.g. generating hordes of files with unknown signatures) and create millions of transient websites in order to defeat URL/DNS blocklists.

While these approaches certainly prevent some attacks, the growing number of incidents and severity of security breaches clearly indicate more effective

alternatives are needed.

What is browser isolation?

The core concept behind browser isolation is security-through-physical-isolation to create a “gap” between a user’s web browser and the endpoint device thereby protecting the device (and the enterprise network) from exploits and attacks. Unlike secure web gateways, antivirus software, or firewalls which rely on known threat patterns or signatures, this is a zero-trust approach.

There are two primary browser isolation architectures: (1) client-based *local* isolation and (2) *remote* isolation.

Local browser isolation attempts to isolate a browser running on a local endpoint using app-level or OS-level sandboxing. In addition to leaving the endpoint at risk when there is an isolation failure, these systems require significant endpoint resources (memory + compute), tend to be brittle, and are difficult for IT to manage as they depend on support from specific hardware and software components.

Further, local browser isolation does nothing to address the control and compliance issues mentioned above.

Remote browser isolation (RBI) protects the endpoint by moving the browser to a remote service in the cloud or to a separate on-premises server within the enterprise network:

- On-premises isolation simply relocates the risk from the endpoint to another location within the enterprise without actually eliminating the risk.
- Cloud-based remote browsing isolates the end-user device and the enterprise’s network while fully enabling IT control and compliance solutions.

Given the inherent advantages, most browser isolation solutions – including S2 Systems – leverage cloud-based remote isolation. Properly implemented, remote

browser isolation can protect the organization from browser exploits, plug-ins, zero-day vulnerabilities, malware and other attacks embedded in web content.

How does Remote Browser Isolation (RBI) work?

In a typical cloud-based RBI system (the blue-dashed box ❶ below), individual remote browsers ❷ are run in the cloud as disposable containerized instances – typically, one instance per user. The remote browser sends the rendered contents of a web page to the user endpoint device ❹ using a specific protocol and data format ❸. Actions by the user, such as keystrokes, mouse and scroll commands, are sent back to the isolation service over a secure encrypted channel where they are processed by the remote browser and any resulting changes to the remote browser webpage are sent back to the endpoint device.

In effect, the endpoint device is “remote controlling” the cloud browser. Some RBI systems use proprietary clients installed on the local endpoint while others leverage existing HTML5-compatible browsers on the endpoint and are considered ‘clientless.’

Data breaches that occur in the remote browser are isolated from the local endpoint and enterprise network. Every remote browser instance is treated as if compromised and terminated after each session. New browser sessions start with a fresh instance. Obviously, the RBI service must prevent browser breaches from leaking outside the browser containers to the service itself. Most RBI systems provide remote file viewers negating the need to download files but also have the ability to inspect files for malware before allowing them to be downloaded.

A critical component in the above architecture is the specific remoting technology employed by the cloud RBI service. The remoting technology has a significant impact on the operating cost and scalability of the RBI service, website fidelity and compatibility, bandwidth requirements, endpoint hardware/software requirements

and even the user experience. Remoting technology also determines the effective level of security provided by the RBI system.

All current cloud RBI systems employ one of two remoting technologies:

(1) **Pixel pushing** is a video-based approach which captures pixel images of the remote browser ‘window’ and transmits a sequence of images to the client endpoint browser or proprietary client. This is similar to how remote desktop and VNC systems work. Although considered to be relatively secure, there are several inherent challenges with this approach:

- Continuously encoding and transmitting video streams of remote webpages to user endpoint devices is very costly. Scaling this approach to millions of users is financially prohibitive and logistically complex.
- Requires significant bandwidth. Even when highly optimized, pushing pixels is bandwidth intensive.
- Unavoidable latency results in an unsatisfactory user experience. These systems tend to be slow and generate a lot of user complaints.
- Mobile support is degraded by high bandwidth requirements compounded by inconsistent connectivity.
- HiDPI displays may render at lower resolutions. Pixel density increases exponentially with resolution which means remote browser sessions (particularly fonts) on HiDPI devices can appear fuzzy or out of focus.

(2) **DOM reconstruction** emerged as a response to the shortcomings of pixel pushing. DOM reconstruction attempts to clean webpage HTML, CSS, etc. before forwarding the content to the local endpoint browser. The underlying HTML, CSS, etc., are *reconstructed* in an attempt to eliminate active code, known exploits, and other potentially malicious content. While addressing the latency, operational cost, and user experience issues of pixel pushing, it introduces two significant new issues:

- Security. The underlying technologies – HTML, CSS, web fonts, etc. – are the attack vectors hackers leverage to breach endpoints. Attempting to remove malicious content or code is like washing mosquitos: you can attempt to clean them, but they remain inherent carriers of dangerous and malicious material. It is impossible to identify, in advance, all the means of exploiting these technologies even through an RBI system.
- Website fidelity. Inevitably, attempting to remove malicious active code, reconstructing HTML, CSS and other aspects of modern websites results in broken pages that don't render properly or don't render at all. Websites that work today may not work tomorrow as site publishers make daily changes that may break DOM reconstruction functionality. The result is an infinite tail of issues requiring significant resources in an endless game of whack-a-mole. Some RBI solutions struggle to support common enterprise-wide services like Google G Suite or Microsoft Office 365 even as malware laden web email continues to be a significant source of breaches.

Customers are left to choose between a secure solution with a bad user experience and high operating costs, or a faster, much less secure solution that breaks websites. These tradeoffs have driven some RBI providers to implement both remoting technologies into their products. However, this leaves customers to pick their poison without addressing the fundamental issues.

Given the significant tradeoffs in RBI systems today, one common optimization for current customers is to deploy remote browsing capabilities to only the most vulnerable users in an organization such as high-risk executives, finance, business development, or HR employees. Like vaccinating half the pupils in a classroom, this results in a false sense of security that does little to protect the larger organization.

Unfortunately, the largest "gap" created by current remote browser isolation systems is the void between the potential of the underlying isolation concept and

the implementation reality of currently available RBI systems.

S2 Systems Remote Browser Isolation

S2 Systems remote browser isolation is a fundamentally different approach based on S2-patented technology called Network Vector Rendering (NVR).

The S2 remote browser is based on the open-source Chromium engine on which Google Chrome is built. In addition to powering Google Chrome which has a ~70% market share[3], Chromium powers twenty-one other web browsers including the new Microsoft Edge browser.[4] As a result, significant ongoing investment in the Chromium engine ensures the highest levels of website support, compatibility and a continuous stream of improvements.

A key architectural feature of the Chromium browser is its use of the Skia graphics library. Skia is a widely-used cross-platform graphics engine for Android, Google Chrome, Chrome OS, Mozilla Firefox, Firefox OS, FitbitOS, Flutter, the Electron application framework and many other products. Like Chromium, the pervasiveness of Skia ensures ongoing broad hardware and platform support.

Skia code fragment

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. Chromium compositing, layout and rendering are extremely complex with multiple parallel paths optimized for different content types, device contexts, etc. The following figure is an egregious simplification for illustration purposes of how S2 works (apologies to Chromium experts):

S2 Systems NVR technology intercepts the remote Chromium browser's Skia draw commands ❶, tokenizes and compresses them, then encrypts and transmits them across the wire ❷ to any HTML5 compliant web browser ❸ (Chrome,

Firefox, Safari, etc.) running locally on the user endpoint desktop or mobile device. The Skia API commands captured by NVR are pre-rasterization which means they are highly compact.

On first use, the S2 RBI service transparently pushes an NVR WebAssembly (Wasm) library ❹ to the local HTML5 web browser on the endpoint device where it is cached for subsequent use. The NVR Wasm code contains an embedded Skia library and the necessary code to unpack, decrypt and "replay" the Skia draw commands from the remote RBI server to the local browser window. A WebAssembly's ability to "*execute at native speed by taking advantage of common hardware capabilities available on a wide range of platforms*"[5] results in near-native drawing performance.

The S2 remote browser isolation service uses headless Chromium-based browsers in the cloud, transparently intercepts draw layer output, transmits the draw commands efficiency and securely over the web, and redraws them in the windows of local HTML5 browsers. This architecture has a number of technical advantages:

(1) Security: the underlying data transport is not an existing attack vector and customers aren't forced to make a tradeoff between security and performance.

(2) Website compatibility: there are no website compatibility issues nor long tail chasing evolving web technologies or emerging vulnerabilities.

(3) Performance: the system is very fast, typically faster than local browsing (subject of a future blog post).

(4) Transparent user experience: S2 remote browsing feels like native browsing; users are generally unaware when they are browsing remotely.

(5) Requires less bandwidth than local browsing for most websites. Enables advanced caching and other proprietary optimizations unique to web browsers and the nature of web content and technologies.

(6) Clientless: leverages existing HTML5 compatible browsers already installed on user endpoint desktop and mobile devices.

(7) Cost-effective scalability: although the details are beyond the scope of this post, the S2 backend and NVR technology have substantially lower operating costs than existing RBI technologies. Operating costs translate directly to customer costs. The S2 system was designed to make deployment to an entire enterprise and not just targeted users (aka: vaccinating half the class) both feasible and attractive for customers.

(8) RBI-as-a-platform: enables implementation of related/adjacent services such as DLP, content disarm & reconstruction (CDR), phishing detection and prevention, etc.

S2 Systems Remote Browser Isolation Service and underlying NVR technology eliminates the disconnect between the conceptual potential and promise of browser isolation and the unsatisfying reality of current RBI technologies.

Cloudflare + S2 Systems Remote Browser Isolation

Cloudflare's global cloud platform is uniquely suited to remote browsing isolation. Seamless integration with our cloud-native performance, reliability and advanced security products and services provides powerful capabilities for our customers.

Our Cloudflare Workers architecture enables edge computing in 200 cities in more than 90 countries and will put a remote browser within 100 milliseconds of 99% of the Internet-connected population in the developed world. With more than 20 million Internet properties directly connected to our network, Cloudflare remote browser isolation will benefit from locally cached data and builds on the impressive connectivity and performance of our network. Our Argo Smart Routing capability leverages our communications backbone to route traffic across faster and more reliable network paths resulting in an average 30% faster access to web assets.

Once it has been integrated with our Cloudflare for Teams suite of advanced security products, remote browser isolation will provide protection from browser exploits, zero-day vulnerabilities, malware and other attacks embedded in web content. Enterprises will be able to secure the browsers of all employees without having to make trade-offs between security and user experience. The service will enable IT control of browser-conveyed enterprise data and compliance oversight. Seamless integration across our products and services will enable users and enterprises to browse the web without fear or consequence.

Cloudflare's mission is to help build a better Internet. This means protecting users and enterprises as they work and play on the Internet; it means making Internet access fast, reliable and transparent. Reimagining and modernizing how web browsing works is an important part of helping build a better Internet.

[1] https://www.w3.org/History/1989/proposal.html

[2] "Internet World Stats,"https://www.internetworldstats.com/, retrieved 12/21/2019.

[3] "Kleiner Perkins 2018 Internet Trends", Mary Meeker.

[4] https://www.statista.com/statistics/544400/market-share-of-internet-browsers-desktop/, retrieved December 21, 2019

[5] https://en.wikipedia.org/wiki/Chromium_(web_browser), retrieved December 29, 2019

[6] https://webassembly.org/, retrieved December 30, 2019

We protect entire corporate networks, help customers build Internet-scale applications efficiently, accelerate any website or Internet application, ward off

DDoS attacks, keep hackers at bay, and can help you on your journey to Zero Trust.

Visit 1.1.1.1 from any device to get started with our free app that makes your Internet faster and safer.

To learn more about our mission to help build a better Internet, start here. If you're looking for a new career direction, check out our open positions.

ON AIR | **CLOUDFLARE TV**

Developer Speaker Series: Taylorism has entered the ChatGPT with Heidi Waterhouse

Tune In

Product News | Cloudflare Access | Cloudflare Zero Trust | Security | VPN

Follow on Twitter

Cloudflare | Cloudflare

RELATED POSTS

March 17, 2022 12:59PM

Clientless Web Isolation is now generally available

Today, we're excited to announce that Clientless Web Isolation is generally available...

By Tim Obezuk

Security Week, Remote Browser Isolation, RBI, Clientless, CASB

April 20, 2021 2:00PM

Start building your own private network on Cloudflare today

Starting today, your team can build a private network on Cloudflare's network....

By Sam Rhea

Developer Week, Developers, Cloudflare Access, Zero Trust, VPN

August 17, 2021 1:59PM

6 New Ways to Validate Device Posture

Cloudflare for Teams adds additional posture capabilities to better protect Access backed applications...

By Kyle Krum

Cloudflare Access, Cloudflare Gateway, WARP, Cloudflare Zero Trust, Product News

April 15, 2021 2:00PM

A Zero Trust terminal in your web browser

Starting today, your team can use that same platform to seamlessly connect to non-HTTP resources from inside of a browser with the same level of Zero Trust control available in web applications....

By Sam Rhea

Developer Week, Developers, Zero Trust, SSH, Cloudflare Access