SERAPHIC

Enterprise Browser Security: A New Way **to Protect** Users, Devices, and Data

White paper | January 2024

Seraphic Security

Table of Contents

Introduction: Browsers are the primary enterprise application

70% of organizations currently support hybrid work and such arrangements are possible only because many line-of-business applications are now delivered via Software-as-a-Service (SaaS).[1] As a result, 85% of workers do most or all their work in a web browser.

Web browsers are the de facto client for enterprise applications and services: a "super app" that is the hub of the modern digital workplace. The increasing popularity of web-based productivity suites like Google Workspace and Microsoft 365, coupled with a wide variety of smaller but equally important SaaS-based applications that range from collaboration to project management to payment processing, has made the browser an indispensable tool for the modern workforce. The fact that browsers are continuously executing external code can potentially create substantial risks but because browsers are the nexus for such a wide variety of threats, they are also an ideal location to consolidate the delivery of security capabilities.

A necessary precondition for a secure digital workplace is an uncompromised endpoint that is accessible only by an authorized user. If that requirement cannot be satisfied, then both the data at rest on the device and the data in use as it is accessed from the device are at risk. Such data may take the form of a user's credentials or session information (e.g., cookies or tokens), or it may be an organization's proprietary information. Creating such a secure digital workspace requires protecting against two distinctly different kinds of threats: those that can impact users and endpoints when they access untrusted resources—such as the public Internet—and those that can impact corporate resources when they are accessed by untrusted endpoints.

1 The Littler® Annual Employer Survey Report, May 2023 – https://www.littler.com/files/2023_littler_employer_survey_report.pdf

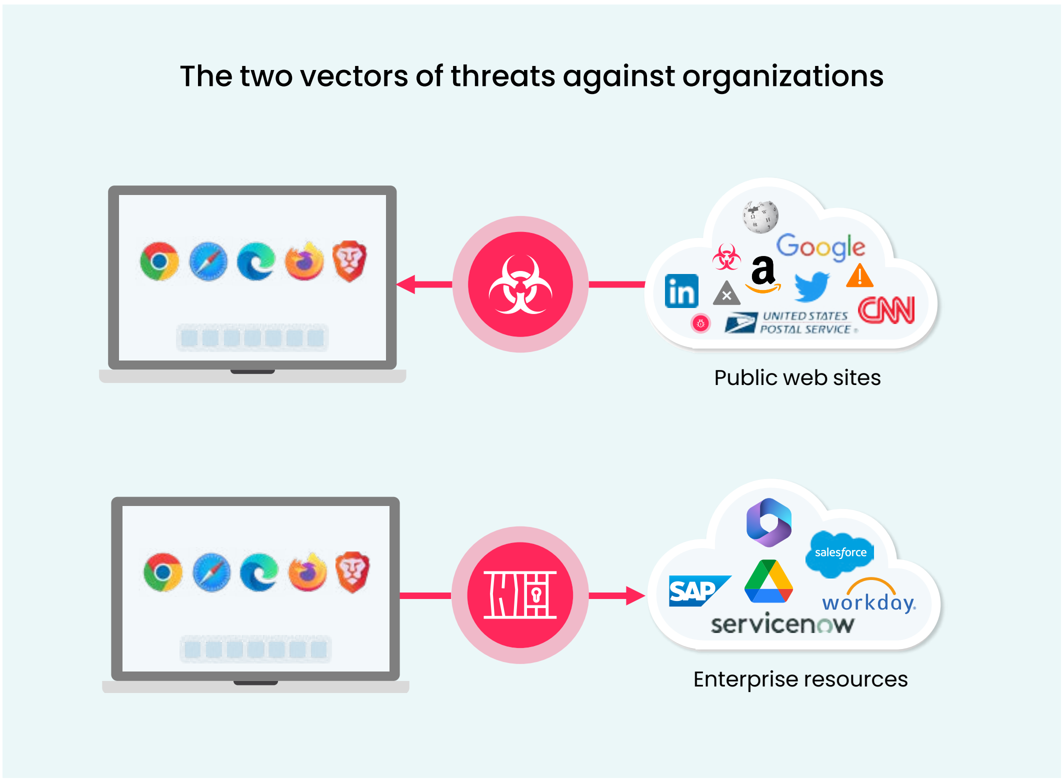

Figure 1 – Scenario 1: External attacks against trusted endpoints,
Scenario 2: Threats against corporate assets from untrusted endpoints

A new approach to protect the device, user, and corporate data

In the last few years multiple new approaches to protect the browser have emerged:

- **Local isolation** – opening the browser in a separate VM machine locally, ensuring that malicious code is isolated within the context of a virtual instance and does not affect the host device.
- **Remote isolation –** opening selected browser sessions in a remote browser, ensuring that a potentially malicious code stays isolated and doesn't affect the host device.

These methods have proven somewhat effective against specific types of attacks targeting the endpoint. However, these approaches are less effective when it comes to protect the user identity, and they do have a substantial price in form of degraded end-user experience.

- **Browser Extension –** providing browser governance, control and threat prevention by adding a browser extension, as part of the extension framework provided by different browser vendors. Browser extension adds local protection in the browser. However, browser extensions are constrained by the Extension APIs exposed by the browsers and, because of the restrictions the APIs impose, do not have visibility into all browser and user actions.

- **Dedicated browser** – replacing the existing browser with a different browser which is managed by the enterprise. This approach requires migration to a new browser. Users lose the benefit of the advanced features because building a derivative browser requires a significant investment both to maintain synchronization with the upstream project and to bridge the gap between the feature-functionality available in mainstream browsers from the dominant vendors. Even with a dedicated browser, the device is still exposed as the user can use other browsers which aren't controlled by the enterprise. Moreover, these browsers are forks of Chromium meaning that they inherit all of the bugs and vulnerabilities of Chromium, but they are also unable to improve security without making substantial modifications—modification which well-resourced organizations such as Google and Microsoft have, so far, been unable to make—to the bundle of highly complex components that comprise Chromium.

Seraphic takes a different approach, one that is not limited to a specific browser nor does it require a migration to a new browser, it is seamless to the end user, and it is also not limited by extension manifest. Seraphic is a JavaScript agent sitting in the heart of the browser, on top of the JavaScript Engine, providing it with runtime context not available in other solutions. This unique location makes it possible to apply innovative approaches—such as Moving Target Defense (MTD)—to exploit prevention and other real-time analysis techniques that don't depend on threat intelligence feeds and static Indicators of Attack (IoAs) Indicators of Compromise (IoCs) to stop sophisticated attacks such as spear phishing and HTML smuggling. The result is unparalleled protection for the device, user and data.

What is a browser agent?

Despite the substantial role of the browser in the delivery of modern IT, it is generally not considered a security tool. The Seraphic browser agent is executed directly by the browser engine (rather than interfacing with it via the Extension API) enabling it to deliver advanced enterprise security capabilities that other tools—such as OS agents, extensions, proxy, and dedicated browsers—cannot provide. The Seraphic Agent is native JavaScript meaning that it is not only browser agnostic, but it can be used anywhere there is a JavaScript runtime: apps based on Electron or Microsoft Edge WebView2, Office 365, NodeJS, JerryScript, and more. [2]

Protecting the device by protecting the browser

It is often assumed that tools such as Antivirus/Endpoint Protection Platforms (EPP) and Endpoint Detection and Response (EDR) can prevent compromise of the endpoint via the browser in the same way that they prevent compromise via other processes: by using hooks in the operating system kernel to monitor and interact with system calls and track process trees.

Unlike the majority of operating system processes, however, browsers possess an execution space that is invisible to external tools. Routine browser operations such as Document Object Model (DOM) and image rendering, or just-in-time compilation and execution of scripts—all of which can potentially be used to trigger vulnerabilities in browsers or gain unauthorized access to information browsers store—take place entirely within the browser process and are therefore invisible to traditional operating system-based approaches. By the time that malicious activity originating in the browser is identified by conventional endpoint tools, it is often too late.

Like any complex piece of software, browsers contain bugs, some of which manifest as security vulnerabilities. In fact, hundreds of high-severity browser vulnerabilities are discovered every year.

2 JerryScript is a lightweight JavaScript engine intended to run on constrained devices such as the microcontrollers used in Internet of Things (IoT) devices

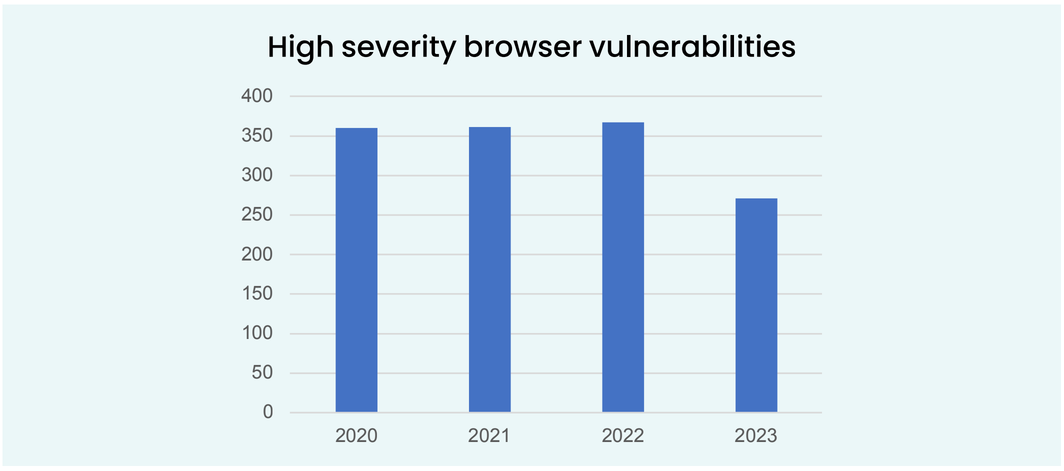

Figure 3 – *Number of browser vulnerabilities with a CVSS 3.0 score greater than 7, 2020–2023 (YTD); source: cvedetails.com*

Browsers are also the targets of a disproportionate number of the zero-day exploits discovered in the wild.

Because browsers—by design—access the public Internet and process untrusted code, it is much easier for adversaries to reach the attack surface created by these vulnerabilities. In order to stay safe, users must update their browsers. However, there is always a delay between the discovery of an exploited vulnerability and the development and release of a patch. Furthermore, patching may be delayed if there is no mechanism in place for organizations to enforce the updates (which usually involves a voluntary browser restart by the end user) or if organizations intentionally postpone browser updates during the compatibility testing they rely on to ensure business continuity. Until every browser is running a fully up-to-date version, organizations remain in the "patch gap" and exposed.

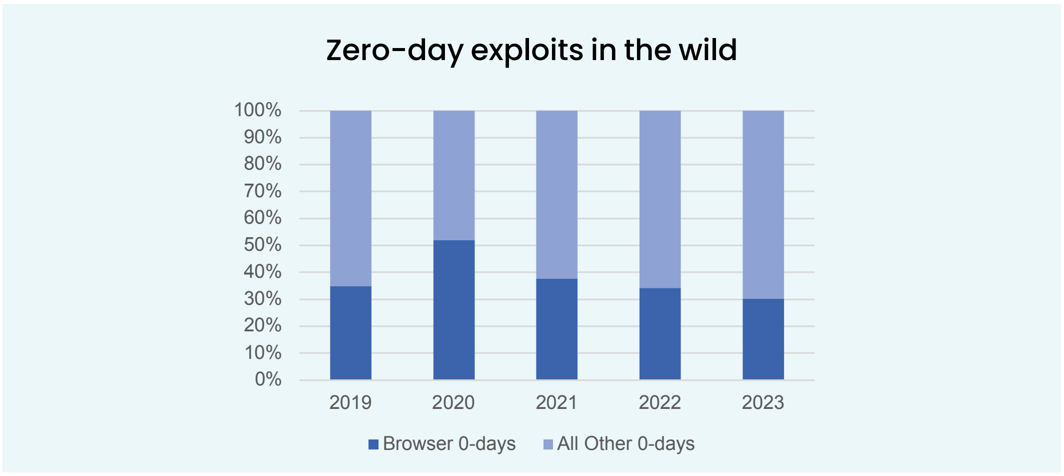

Figure 4 - *Browser 0-days vs. All Other 0-Days in the wild, 2020–2023 (YTD); source: Google Project Zero*

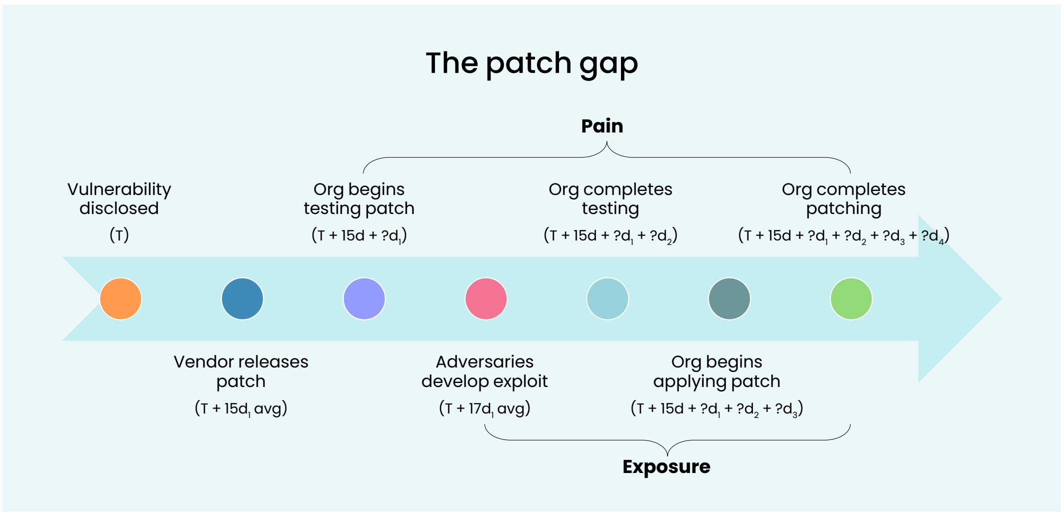

Figure 5 - *The "patch gap" timeline, which is often a minimum of two weeks but can vary widely based on procedures within organizations*

Although patch management does provide some measure of protection, it may be short-lived given the frequency of zero-day exploit discovery is increasing. This means that organizations who have just completed a patch cycle addressing one vulnerability may find themselves exposed to a new one and repeating the patching process in an endless loop.

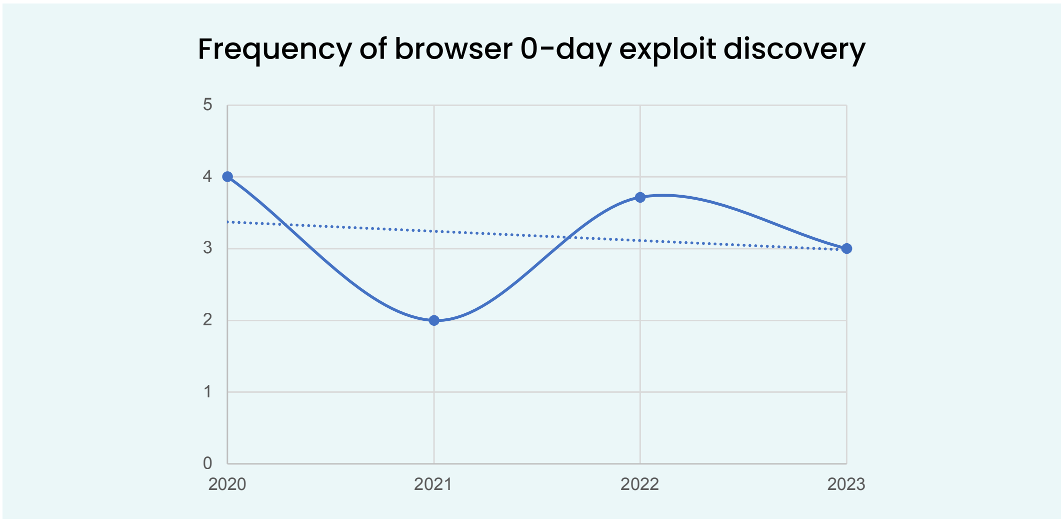

Figure 6 - *Average interval (in weeks) between discovery of browser 0-day exploits in the wild, 2020–2023 (YTD); source: Google Project Zero*

Browser exploitation is only one of the methods that can lead to device compromise. Web-based attacks such as Cross-Site Scripting (XSS) and "drive-by compromises", also known as HTML smuggling, can lead to the delivery and execution of malware. These attacks rely on the innate behavior of browsers rather than vulnerabilities and, in the last few years, the persistence and durability of malware families like GootLoader and SocGholish—which are often delivered through such attacks—have made it increasingly clear that they are capable of evading existing defenses and represent a viable path to compromise end-user devices.

To protect endpoints from compromise via the browser, a solution that uses a browser-based approach with characteristics similar to those of operating system-based solutions is necessary. Seraphic adds such security directly to any browser.

The Seraphic browser agent creates an abstraction layer between external code and the heart of the browser: the JavaScript Engine (JSE). Similar to Address Space Layout Randomization (ASLR), the Seraphic agent applies patented, dynamic randomization to the browser's execution environment which eliminates the deterministic nature of certain artifacts required for triggering of vulnerabilities and prevents exploitation without the need for any a priori knowledge of the underlying vulnerability or other detection techniques. Also, like ASLR, this protection is completely transparent to end users, does not impact performance, and does not alter the behavior of benign code. Seraphic does not depend on detection techniques, which are ineffective against new, novel, and unknown patterns which offers protection against zero-day and unpatched N-day exploits (i.e., Seraphic protects browsers during the "patch gap").

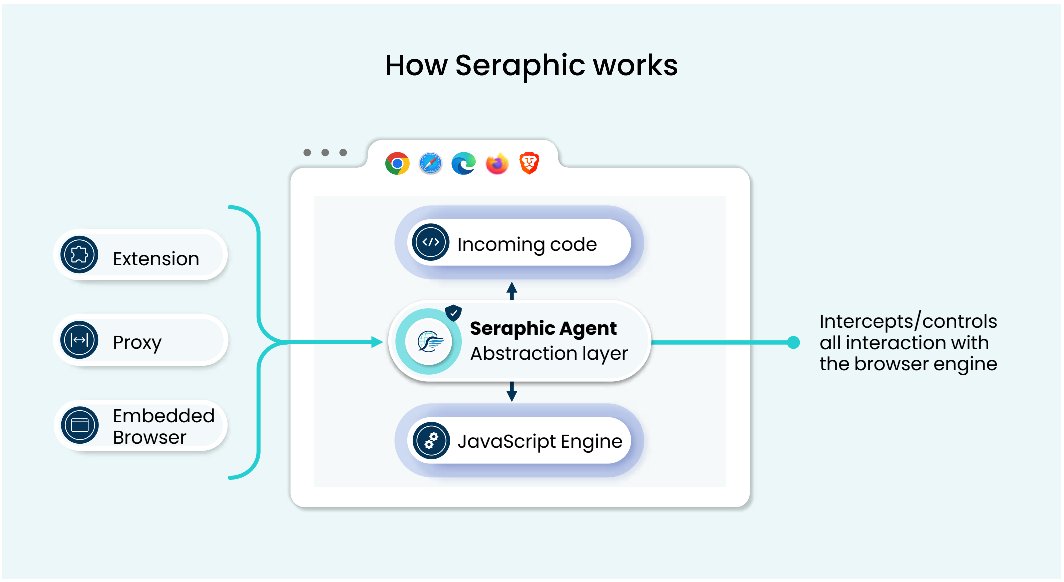

Figure 7 – *The Seraphic Agent can be injected by extension, proxy, or embedded in a dedicated browser. It creates an abstraction layer that can monitor, intercept, and control the interactions between incoming code and the JavaScript Engine.*

The unique location of the Seraphic agent also provides visibility and runtime context that enables unmatched detection and response capabilities within the browser itself, similar to those of EDR at the OS level. The Seraphic agent monitors and evaluates real-time browser telemetry and user context, enabling it to accurately detect and block other web-based attacks, including difficult-to-identify techniques that bypass traditional network- and endpoint-based technologies.

Seraphic Attack Protection in Action: SocGholish

SocGholish is a long-lived malware family that is frequently deployed to provide initial access and deliver additional payloads such as ransomware. It is often served from compromised websites with good reputations and—because it is JavaScript-based—it is trivial to obfuscate. Seraphic routinely detects and blocks SocGholish infection attempts that go undetected by other security tools. In the first nine months of 2023, Seraphic prevented an average of 100 SocGholish infections per month.

Comparison of attack protection capabilities

	Approach	**Effect**
Seraphic (Browser Agent)	Prevents exploitation of unknown and unpatched vulnerabilities	Protects during the "patch gap"
EPP/ EDR (OS Agent)	Detects known techniques/ methods	Limited protection against unknown exploits
Proxy	Requires HTTPS interception, deep packet inspection, and network sandboxing but is trivial to evade	No protection against unknown exploits
Dedicated Browser	Only detects known malware in dedicated browser; inherits Chromium vulnerabilities and doesn't protect endpoint from exploitation of other browsers	No protection against unknown/unpatched exploits Requires patching; has longer patch gap
Extension	Offers limited protection against known malware	No protection against unknown exploits

Table 1 - *Differences between the web-based attack protection capabilities of various security tools*

Protecting user credentials and sessions

Phishing and stolen credentials are not new or novel threats, but recently they have risen to become the most common methods of compromising an organization. There were record numbers of phishing attacks in three consecutive quarters in 2022 and stolen credentials were involved in nearly half of all breaches between late 2021 and late 2022.[3]

There are several reasons why these methods have grown so popular

1. **The range of potential targets is nearly universal.** Performing these attacks does not depend on any specific technology, device, platform, or operating system version.

2. **The barrier to entry is extremely low.** Like most conventional technology offerings, phishing campaigns and infrastructure can also be delivered "as-a-service" without any heavy up-front investments in activities like reverse engineering and exploit development that may be required for other forms of attack.

3. **The risk-to-reward ratio favors attackers.** they don't have to "burn" exploits and hope that they remain undetected throughout the attack. A successful attack allows them to impersonate an authorized user with all the attendant access to enterprise resources and without any special tooling. Once they've established a foothold, they can expand their attacks.

4. **The last line of defense is typically a person** and people can be easier to manipulate because they can be relied upon for their good-faith efforts to do their jobs, rather than depending on the existence of a specific flaw in technology or code.

The core of any successful phishing attempt is a sufficiently convincing or authentic-looking login page or web form. While considerable time and effort are expended to teach users what to look for, UI redressing techniques like

3 APWG Phishing Trends Activity Report, 4th Quarter 2022 – https://docs.apwg.org/reports/apwg_trends_report_q4_2022.pdf;

2023 Verizon Data Breach Investigations Report – https://www.verizon.com/business/resources/reports/dbir/2023/master-guide/

clickjacking and Browser-in-the-Browser (BitB) are making that training less and less effective. However, there are several other approaches that are being taken to address these attacks:

1. **Intelligence feeds** – Email gateways and secure web gateways use shared data from different sources to identify and strip or block malicious ULRs. Unfortunately, this only provides a partial solution. Phishing sites are very short-lived, lasting an average of 21 hours and the anti-phishing ecosystem usually takes about nine hours to determine that a site is malicious (and longer still for that verdict to be disseminated). That means that phishing sites are unclassified for almost 1/2 of their existence and, even then, over 1/3 of phishing site visitors reach those sites after they've been classified as malicious.

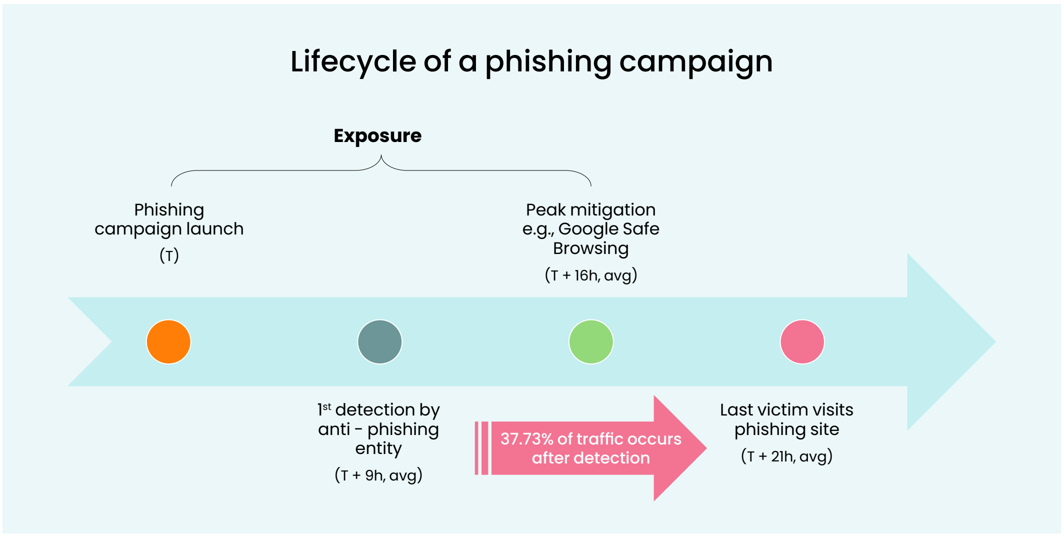
Lifecycle of a phishing campaign
Exposure
Phishing campaign launch (T)
Peak mitigation e.g., Google Safe Browsing (T + 16h, avg)
1st detection by anti - phishing entity (T + 9h, avg)
37.73% of traffic occurs after detection
Last victim visits phishing site (T + 21h, avg)

Figure 8 - *Phishing attack timeline: typical campaigns last less than 24 hours but take an average of 9 hours to be discovered. Over 1/3 of phishing site visits occur after the site has been discovered.*

2. **Static and dynamic analysis by automated web crawlers** – These solutions continuously analyze websites, trying to find indications of malicious intent. While these bots are successful at identifying some sites, adversaries now employ a wide variety of server- and client-side cloaking techniques—some as simple as CAPTCHAs—to attempt to ensure that the phishing page is shown only to the intended victim and innocuous pages are shown in any other scenario.

4 Oest, et al – Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale, https://www.usenix.org/system/files/sec20fall_oest_prepub.pdf

Seraphic provides multiple protections for user credentials and other authentication material:

1. As in the case of web-based attack detection described above, the Seraphic agent monitors and evaluates real-time browser telemetry and user context to identify risky sites and renders them in read-only mode so users cannot input their credentials or other sensitive information. Because Seraphic is directly monitoring browser and user activity in real time, it is highly resistant to all of the evasion techniques (e.g., CAPTCHAs) that confound other solutions. It does not depend on any threat intelligence offers protection from unclassified and otherwise unknown URLs.

2. Seraphic can also encrypt cached/long-lived authentication material such as session cookies and tokens, meaning that any attempt to steal them will yield unusable versions and stop any user impersonation or session hijacking.

Protection Against a Real-world Breach: Okta

In October 2023, the Okta support case management system was compromised by attackers who leveraged it to gain access to customer HTTP Archive (HAR) files containing session cookies of those customers' users. The attackers were then able to use those session cookies to gain unauthorized access to customer environments. Seraphic encrypts session cookies at rest in browser storage, as well as during the recording of HAR files. Had the session cookies been protected by Seraphic, they would have been useless to the attackers.

3. Seraphic can be configured to notify organizations of password reuse, enabling them to require password changes. Taking timely action on reused passwords makes organizations much more resistant to credential stuffing attacks.

Seraphic Attack Protection in Action: Reverse Proxy Phishing

Multi-Factor Authentication (MFA) has significantly reduced the impact of conventional phishing, causing adversaries to pivot to tools such as Evilginx and Modlishka that proxy requests directly to websites and capture the session cookies and/or tokens generated during successful authentication. Seraphic's ability to analyze page attributes at runtime and render the page in read-only mode enables the identification and prevention of these sophisticated attacks, safeguarding sensitive authentication material from theft and stopping attackers from being able to impersonate valid users.

Comparison of identity protection capabilities

	Approach	**Identity Protection**	**Credential Re-use Protection**
Seraphic (Browser Agent)	Evaluates page attributes in real time, does not require external data	Encrypts session cookies, blocks user input on unknown and known phishing sites	Alerts on credential re-use
EPP/ EDR (OS Agent)	Intelligence-based protection only	No protection for auth mat'l, only blocks known phishing sites	No protection against credential re-use
Proxy	Relies on database(s) of known phishing URLs	No protection for auth mat'l, only blocks known phishing sites	No protection against credential re-use
Dedicated Browser	Relies on database(s) of known phishing URLs	Only protects auth mat'l in dedicated browser , only blocks known phishing sites	No visibility for credential use outside dedicated browser
Extension	May be able to identify phishing sites but cannot prevent input	No protection for auth mat'l	No protection against credential re-use

Table 2 - *Differences between the identity protection capabilities of various security tools*

Securing corporate data

Browsers provide quick and efficient access to a wide variety of enterprise resources and data. Unlike the early days of the web, this data isn't only for download and use in external applications. Now the browser renders much of this data — documents, presentations, spreadsheets, or videos, etc.—directly and, importantly, also enables users to interact with it.

While this capability creates a tremendous amount of convenience, it also creates significant challenges to controlling how and where the data flows. Ensuring that data stays within the browser or remains confined to particular websites or applications is an increasingly difficult problem. There are two common approaches to dealing with this challenge:

1. Implementing Data Leakage Prevention (DLP) scanning and enforcement policies at a web gateway (typically a forward proxy). In organizations where every employee works from a corporate office, this can be effective because it is easy to ensure that all traffic traverses the proxy. However, these tools perform in-line inspection and enforcement, so they lack the ability to control basic user actions like copy/paste, print, and screen capture. The use of proxies also becomes a much more complicated problem when end users work remotely or if they're using personal devices because it is harder to force traffic through the proxy. There are also practical limits to the amount/depth of inspection that proxies can perform, given the constraints imposed by the availability of compute and network resources, creating the potential for significant performance impact to end users.

2. For SaaS applications, organizations may implement a Cloud Access Security Broker (CASB). CASBs can perform scanning and enforcement by operating as a reverse proxy, via an API integration, or a combination of the two. They enable administrators to prescribe the ways in which users interact with applications and data, but they also require traffic steering, may only have visibility into sanctioned applications (offering no protection against unsanctioned applications or "shadow IT"), and also lack control of user actions within the browser. Although some organizations may use Secure Web Gateways (SWGs i.e., forward proxies) in conjunction with their CASBs to monitor and control unsanctioned applications, this approach is generally ineffective for unmanaged endpoints because it is not possible to reliably force traffic through the proxy.

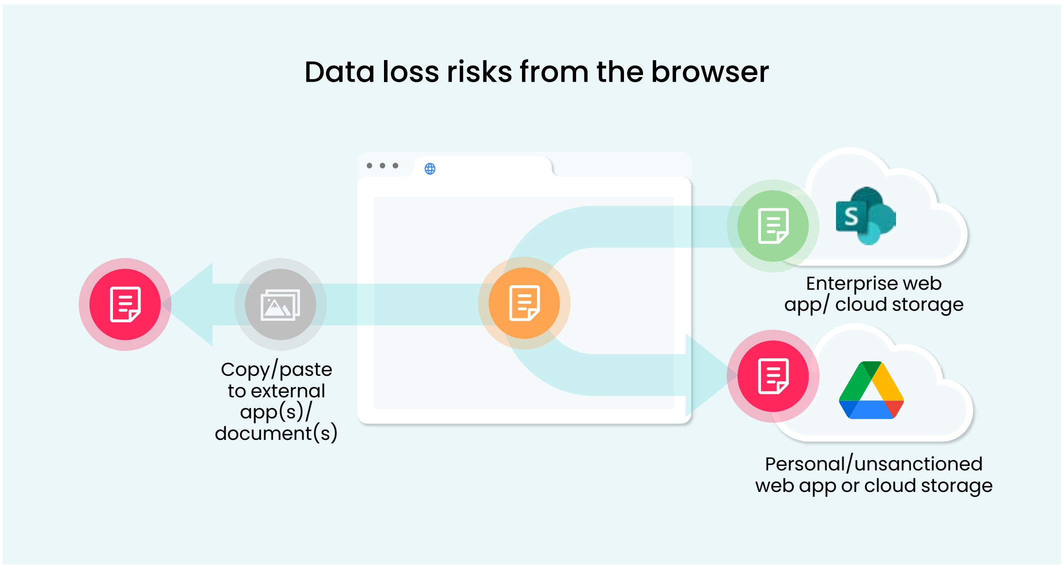
Data loss risks from the browser
Copy/paste to external app(s)/ document(s)
Enterprise web app/ cloud storage
Personal/unsanctioned web app or cloud storage

Figure 9 – *Browser data loss vectors: copy/paste from the browser into external applications and data "hair pinning" from a sanctioned service to an unsanctioned one.*

Because Seraphic operates in the browser, it protects sensitive data regardless of web destination. Seraphic uses a flexible policy framework that allows administrators to create granular definitions of sensitive data and tightly control how users interact with that data by:

1. Providing full control of all user actions, including copy/paste, print, manual data input, viewing page source, or opening developer tools.
2. Dynamically masking sensitive, so it is not displayed to the user.
3. Preventing the download of sensitive files to unmanaged devices while allowing access within a protected browser session, enabling users to work but stopping accidental or intentional data leakage.
4. Blocking screen capture or screen sharing of protected browser sessions.
5. Watermarking web pages and data with unique session identifiers so the source can be traced if data is leaked through a photograph of a screen.

The Seraphic agent performs all analysis and enforcement locally, without the need to upload anything to the cloud, providing a superior end-user experience and ensuring a company's data always remains within their environment and under their control.

Comparison of DLP capabilities

	Controls	Effect
Seraphic (Browser Agent)	Can control all user actions in browsers and desktop SaaS apps	Performs all scanning and protection on the client; masks sensitive data
EPP/ EDR (OS Agent)	Can only monitor local storage/ devices, no visibility/ control for SaaS and web	No protection for auth mat'l
Proxy	No control of client-side activity	Requires traffic-steering client
Dedicated Browser	Only restricts activity in dedicated browser	Only protects data in dedicated browser, no support for additional browsers or desktop SaaS apps
Extension	Controls constrained by capabilities available to extension APIs	Limited ability to control user actions or restrict data flow

Table 3 - *Differences between the web/SaaS DLP capabilities of various security tools*

Access control for web-based applications

The "webification" of enterprise applications is a decades-long phenomenon driven by a range of factors including cross-platform compatibility, cost-effectiveness, scalability, ease of maintenance and use, and more. While the cloud-based delivery of SaaS apps makes many enterprise resources accessible from anywhere, many organizations still have on-premises web applications that are accessible only from the corporate network.

As hybrid and remote work took hold, the use of Virtual Private Networks (VPNs) became necessary to ensure employees could continue to access resources essential to their jobs, even if they were offsite.

Although VPNs are intended to provide secure access to corporate networks, they may actually have the opposite effect:

1. Successful phishing or other credential compromise can enable adversaries to gain access to the corporate network with the same privileges as the user, and potentially allow them to move laterally and gain additional access.
2. VPN infrastructure creates additional attack surface and these systems have consistently been identified as having some of the most routinely exploited vulnerabilities that can give attackers access even if they don't possess valid credentials.
3. Their logging capabilities may not provide a clear picture of user activity, requiring SOC analysts or Incident Responders to correlate VPN logs with application logs to construct an audit trail.
4. VPN infrastructure doesn't provide any native DLP capabilities, meaning that additional tools must be deployed or integrated to protect data flowing out of the corporate datacenter.

There are multiple new technology categories attempting to make up for the deficiencies of legacy VPN:

1. Virtual Desktop Infrastructure (VDI) isolates the endpoint from direct access to the network, but requires its own costly and complex infrastructure, may not completely eliminate the need for VPN infrastructure, and may suffer from performance problems.
2. Zero Trust Network Access (ZTNA) products attempt to reduce the risk of unauthorized access and lateral movement through the use of strong authentication and application of the principle of least privilege (usually through default deny rules and micro segmentation). Secure Access Service Edge (SASE)/Security Service Edge (SSE) vendors offer "private access" options but those are effectively VPNs which terminate in vendor datacenters, requiring complex network peering between the vendor's datacenter(s) and the datacenter(s) belonging to the organization. SASE/SSE vendors may also be able to provide some DLP capabilities for on-premises apps, but implementation of the private access offering is a prerequisite.

Seraphic provides an efficient solution that addresses all these pain points—regardless of who the user is and what device the user is using to access corporate resources—by transforming any browser into both the remote access client and an access control enforcement point:

1. It can control access to both internal and SaaS web applications without the need for any complex and expensive infrastructure. The access is based on least privilege, taking the user, device and the network used into account.

2. It can be deployed in multiple ways, allowing organizations to support managed endpoints while also offering minimally invasive options for unmanaged endpoints. It does not require the installation of any additional security agents, host-checkers, or monitoring tools.

3. It provides full control and monitoring of all user activity in the browser, regardless of the site or application, the type of action, whether the traffic is encrypted or not, even if all of the activity is confined to the browser.

4. It allows organizations to control the flow of data by designating authorized destinations.

Attack Surface Reduction with Seraphic: Remote Access

A number of breaches have resulted from attacks on remote access infrastructure and vulnerabilities in those products are routinely listed as some of the most-exploited by authorities like the Cybersecurity and Infrastructure Security Agency (CISA). Seraphic enables browsers to become remote access clients without exposing infrastructure directly to the public Internet, thereby reducing the risks of providing access to on-premises applications.

Comparison of access control capabilities

	Approach	Effect
Seraphic (Browser Agent)	Transforms browsers and desktop SaaS apps into enforcement points	Identifies unsanctioned apps
EPP/ EDR (OS Agent)	No web or SaaS app access controls	No protection
Proxy	Requires traffic-steering client, also requires network peering for on-premises web applications	Requires log correlation to identify unsanctioned apps
Dedicated Browser	Requires all access to be through a single browser, no support for additional browsers or desktop SaaS clients	Only identifies unsanctioned apps if access via the dedicated browser
Extension	Depends on browser vendor APIs for identification and access control capabilities	May identify unsanctioned apps but may not be able to enforce access controls

Table 4 - *Differences between the access control capabilities of various security tools*

Conclusion: Better enterprise security begins with the browser

Despite its humble beginnings as a consumer application, the browser has evolved into a critical productivity tool for modern enterprises. The browser's central role also exposes it—and the organizations that depend on it—to a wide variety of risks. Because browsers are at the intersection of productivity and risk, they are also the ideal location to deploy the security capabilities required by modern enterprises. Seraphic provides a simple yet comprehensive solution to address these risks that is transparent to end users, enabling them to continue working from their mainstream browsers, while also giving organizations better protection for their users, endpoints, and data.

Glossary

Browser Agent – A browser agent is a security tool (similar to an operating system security agent) that is executed directly by the browser engine and "hooks" the browser engine, giving it complete visibility into browser operations and providing the most effective protection and policy enforcement.

Browser Agents should not be confused with Browser Extensions, which are add-ons that interact with the browser using a specific API.

Browser Isolation – A generic term encompassing different techniques for keeping the execution of a web browser and the code it renders/executes separate from a host operating system. There are two basic categories of browser isolation: Local Browser Isolation and Remote Browser Isolation.

- **Local Browser Isolation (LBI)** – The process of executing a browser in a dedicated virtual machine (VM) on an end user's device. While this approach can contain the damage of certain types of attacks (e.g., browser exploits) but are ineffective against others (e.g., phishing and session cookie/token theft). Maintaining isolation between the between the browser and the host OS can also negatively impact the user experience (e.g., by disrupting file downloads and taxing system resources).

- **Remote Browser Isolation (RBI)** ¬ The process of executing a browser on virtualized server infrastructure (e.g., in a VM or container). Remote browser isolation can use either "pixel pushing" (which streams bitmaps similar to the way in which many virtual desktops are rendered) or Document Object Model (DOM) Reconstruction (which renders/executes the web content and strips the dynamic elements). In addition to the limitations of LBI, RBI frequently creates application compatibility problems and is also affected by network congestion.

Electron Framework – A framework designed to enable the creation of desktop applications using web technologies such as HTML, CSS, and JavaScript. Electron renders UI elements using Chromium and makes building cross-platform applications easier because it doesn't require developers to maintain separate codebases for different platforms.

The use of Chromium means that apps built using Electron are effectively additional web browsers and, like all Chromium derivatives, Electron apps will inherit Chromium bugs and vulnerabilities. They may also have a longer "patch gap" because the framework must be updated with the new version of Chromium and then app developers must update the framework used by their app.

Enterprise Browser – An emerging product category of web browsers and—in some analysts' definitions—browser add-ons focusing on the security and productivity needs of enterprises rather than the commercial objectives of large browser vendors or the feature/functionality needs of consumers.

Virtually all modern Enterprise Browsers are derivatives of Chromium and therefore inherit all of its bugs and vulnerabilities. They may have longer patch gaps than mainstream browsers due to the independent development pipelines that are gated by the release of the patch from the upstream code repository.

Enterprise Browsers may also struggle to provide additional security capabilities because they do not (and cannot) fundamentally alter the behavior of the core components such as the Document Object Model (DOM) Renderer and JavaScript Engine (JSE).

Enterprise Browser Extension – A component for customizing the capabilities of a web browser to include enterprise-focused management and security capabilities. The ability of such extensions to implement these capabilities is constrained by the Application Programming Interfaces (APIs) exposed by individual browser vendors and cannot be applied to all browser operations (e.g., extensions cannot examine HTTP POST request) making them ineffective for providing protection against certain attacks or enforcing certain kinds of policies.

Microsoft Edge WebView2 Runtime – An alternative to the Electron Framework for building desktop applications using web technologies. It can be bundled and redistributed with applications or installed as a standalone component. Because it is based on Microsoft Edge, it is also a Chromium derivative and therefore inherits the bugs and vulnerabilities. Also, like Electron-based apps, the patch gap may be longer than mainstream browsers due to the distribution and update methods implemented by the apps that use the framework.

Moving Target Defense (MTD) – dynamically changing system dimensions to increase uncertainty and complexity for attackers. MTD can be useful for disrupting exploitation of vulnerabilities by "breaking" the determinism on which attackers depend. There are very few real-world implementations of MTD but one example is Address Space Layout Randomization (ASLR) which randomizes the location of process memory in order to prevent attackers from being able to reliably locate the memory addresses of vulnerable functions.