What’s Up With Site Isolation

With Special Guest Charlie

Today, we're talking about Site Isolation with special guest Charlie. Charlie is responsible for making it happen, and has worked all over the navigation code as well, making sure it all works and works securely.

Site isolation

Site isolation is a way to use Chrome's sandbox to try to protect websites from each other. It's a way to improve the browser security model.

Sandbox

Sandbox is a mechanism that tries to keep web pages contained within the renderer process even if something goes wrong. So if attackers find a bug to exploit, it should still be hard for them to get out and install malware on your computer or do things outside the renderer process.

Concerns about renderer process model

Renderer processes are highly susceptible to attacks because they execute code fetched from untrusted websites, often within a sandboxed environment. Attackers aim to find and exploit bugs in this code, and most vulnerabilities are likely to be found in the renderer process, giving attackers more control. All software has bugs, and attackers attempt to exploit these bugs to achieve their goals. For instance, they might trigger a parsing error in the web browser's code using input such as HTML or JavaScript to make it behave incorrectly, which can be advantageous to them.

Regarding the potential consequences of an attack, in the past, when browsers were less secure, a successful attack could lead to the installation of malware on a user's computer. This compromise could grant attackers access to local files and sensitive network resources, like bank accounts, making it a significant concern that browser security measures aim to mitigate.

Site isolation mechanism

Site isolation in Chrome aims to enhance security by ensuring that each renderer process can only load pages from one website. This means that when you visit different websites, they are loaded in separate processes to prevent them from interfering with each other. One significant

aspect of site isolation is isolating cross-site iframes, which are now executed in separate processes. This architectural change in the browser helps protect websites from each other and minimizes the risk of various types of attacks, such as malware installation and unauthorized access to the file system or network.

iframe, site isolation and origin isolation

An iframe is essentially a web page embedded within another web page, such as an ad or a YouTube video. These iframes can originate from different sources than the top-level page you're viewing, which gives them a separate security context.

A site comprises a registered domain name plus a scheme, like "https://example.com." However, within a site, there can be multiple origins, especially in the case of subdomains like "foo.example.com" and "bar.example.com." Web security is primarily concerned with origins, and normally, origins within the same site should not be able to access each other. Some legacy web APIs, like the ability to modify "document.domain," which allow interactions between different origins within the same site. Since it's challenging to predict in advance whether these interactions will occur, everything from a site is placed in the same process for security reasons. However, there is ongoing work to potentially move away from this practice and focus on isolating origins instead.

Browser security model vs. web security model.

They are related but distinct concepts. The web security model pertains to what web pages, in general, are allowed to do. It encompasses considerations such as what web pages can access from other websites or origins and what permissions they have for accessing resources like cameras and microphones.
On the other hand, the browser security model focuses on how these web security principles are implemented and enforced within the browser. It encompasses measures such as sandboxing, multi-process architecture, and site isolation. The browser security model's goal is not only to enforce the web security model but also to provide additional layers of defense to make it more challenging for attackers to achieve their objectives, even in the presence of software bugs.

Security context

A security context is essentially the environment where code is executed. In the web context, this could be an HTML document or a worker like a service worker. The security principle, on the other hand, is typically associated with an origin in the web. For example, if you have an HTML document from "example.com," it runs in a web page in the browser and is considered to have its security context. An ad from a different origin would be in a different security context. Two different documents with the same security principle might still be able to access each other. They could even be running in separate processes but still have access to the same cookies, local storage, and resources on disk. The key point is that the security principle represents the entity with access to certain resources or data.

Site isolation and navigation

Navigation refers to the process of moving from one web page to another, and it may involve a change in the security context or security principle. Navigation is complex and involves various tasks, such as fetching web pages from the network, handling unload events for safe navigation, and ensuring precise timing for transitioning to the new process. These navigation challenges align with the broader objectives of site isolation and the overall process model in web browsers.

Process model and multiple process structure

The process model for a web browser involves determining which processes are responsible for handling various web pages, primarily focusing on renderer processes and web pages. In the absence of site isolation, a new tab typically gets its dedicated renderer process. However, web pages visited within the same tab usually share the same process. Site isolation introduces a stricter process model, ensuring that web pages from different websites are never placed in the same renderer process. This process model imposes constraints on how navigation functions. The multi-process architecture is a foundational aspect of modern web browsers. It enables the separation of browser code, renderer code, plug-in code, and various utilities into different processes. This architecture enhances security and stability.

Why navigation is so hard

Navigation involves various types of transitions and corner cases. Navigation can include redirecting to different websites, downloading content, or staying within the same document, each with its unique properties. Managing these aspects while ensuring consistent back-forward history is challenging. Features like back-forward cache, pre-rendering, and navigation APIs enhance web developer experiences but also introduce new complexities. Site isolation's

difficulty stems from the fact that navigation can occur in any frame within a web page. Previously, iframes were handled entirely within the renderer process. To implement site isolation, they needed to run iframes from different origins in separate renderer processes.

Sandbox escape and renderer compromise

A "sandbox escape" refers to an attacker exploiting a bug to elevate their privileges to affect the browser process or exit the browser process and access the operating system. It is a more severe scenario than a "site isolation bypass," which allows an attacker to gain access to another website's data or manipulate permissions. "Renderer compromise" is where an attacker exploits a bug within the renderer process itself. Despite having the ability to run native code within the renderer process, the sandbox and site isolation are designed to contain such compromises, preventing attackers from installing malware or accessing other site's data.

Other work on site isolation

Since the initial launch of site isolation, several improvements have been made. Initially, the focus was on mitigating Spectre-related attacks, and compromise renderer defenses to prevent attackers from escalating their privileges within the renderer process. Efforts were made to make site isolation work on Android, although resource constraints limited the isolation of all websites from each other. Extensions were isolated not only from web pages but also from each other. Possible future changes could be related to strict origin isolation and document.domain deprecation.

Defensive and offensive security

Offensive security involves finding vulnerabilities and exploiting them, while defensive security is about protecting systems and fixing vulnerabilities. A metaphor of "layers of Swiss cheese" is used to describe the security landscape. Each layer represents a defense mechanism, but no layer is perfect, and vulnerabilities can emerge when holes in different layers align. Site isolation is presented as a layer that aims to create a boundary between compromised renderer processes and sensitive data.