Site Isolation

Site Isolation is **a security mechanism implemented in modern web browsers to provide stronger isolation between different websites by running each site in its own process**. This prevents one site from accessing or interfering with the data of another site, even in the presence of browser vulnerabilities like speculative execution attacks (e.g., Spectre).

1. How Site Isolation Works

- Traditional Model
  - In older browser architectures, multiple websites could share the same process for rendering.
  - Shared processes could lead to cross-origin data leaks if vulnerabilities were exploited.
- Site Isolation Model
  - **Each website (or "origin") is rendered in a separate process**.
  - Process boundaries prevent one site from accessing another site's data, even if the browser's renderer is compromised.

Key Features

1. **Process Separation**

- Each domain or origin gets its own dedicated process.
- Example: https://example.com and https://another.com run in different processes.

2. **Cross-Origin Data Protection**

- Data from one origin (cookies, DOM objects, etc.) cannot be accessed by another origin.

3. **Memory and Cache Isolation**

- Separate memory spaces for each process prevent leakage of sensitive data between tabs.

2. Benefits of Site Isolation

- **Mitigates Speculative Execution Attacks**
  - Prevents attacks like Spectre from leaking sensitive data by ensuring data resides in isolated processes.
- **Stronger Sandbox**
  - Each process operates in a stricter sandbox, making it harder for attackers to escalate privileges.
- **Cross-Site Scripting (XSS) Containment**
  - Even if one site is compromised via XSS, the attack is confined to that process and cannot affect other sites.
- **Enhanced Privacy**
  - Prevents one site from snooping on another's data or cookies.

3. Limitations of Site Isolation

- Increased Resource Usage

    - Each process consumes memory and CPU resources, leading to higher resource utilization compared to shared processes.
- **Not a Complete Security Solution**
    - Does not protect against all types of web-based attacks (e.g., phishing or drive-by downloads).
- Implementation Complexity
    - Requires careful management of inter-process communication to ensure seamless user experience.

4. Site Isolation in Modern Browsers

a. Google Chrome

- Chrome introduced Site Isolation as a key defense against speculative execution attacks (e.g., Spectre).
- Fully enabled by default since Chrome 67 for desktop and Chrome 77 for Android.

Command-Line Options

- Force-enable Site Isolation:

```
chrome.exe --site-per-process
```

b. Mozilla Firefox

- Firefox uses a similar feature called Fission to isolate websites in separate processes.
- Still under phased deployment as of late 2024, with additional tuning for performance.

c. Microsoft Edge

- Built on the Chromium engine, Edge inherits Site Isolation features from Chromium.

d. Apple Safari

- **Uses a process-per-tab model** but does not implement full site isolation as in Chromium-based browsers.

5. Attacks Mitigated by Site Isolation

- **Spectre and Meltdown**
    - Prevents malicious scripts on one site from stealing sensitive data from another site in the same browser session.
- **Cross-Origin Information Leakage**
    - Blocks unauthorized access to cookies, session tokens, and other sensitive data.
- **Universal XSS (UXSS)**
    - Confines XSS vulnerabilities to a single site, reducing their impact.

6. Enabling and Testing Site Isolation

a. Chrome Example

1. Open Chrome and go to chrome://flags.
2. Search for “Strict Site Isolation”.
3. Enable the flag and restart the browser.

b. Testing Isolation

- Use online tools like Google’s Spectre Test
    - Confirm that Site Isolation prevents speculative execution leaks.

c. Debugging Site Isolation

- Chrome Debugging Command

```
chrome.exe --site-per-process --disable-features=IsolateOrigins
```

7. Best Practices for Developers

- Avoid Relying on Same-Origin Policies Alone
    - Use additional security headers like Content-Security-Policy (CSP) and Strict-Transport-Security (HSTS).
- Reduce Shared Resources
    - Minimize shared objects between origins to benefit fully from isolation.
- Test Compatibility
    - Ensure applications are compatible with site isolation policies.

8. Summary

Aspect	Details
What It Is	Isolates websites in separate browser processes for enhanced security.
Primary Goal	Protect against cross-origin attacks and speculative execution vulnerabilities.
Key Benefits	Data protection, stronger sandboxing, and containment of vulnerabilities.
Limitations	Higher resource usage, not a defense against all web threats.
Implementation	Supported by Chrome, Edge, Firefox (Fission), with varying levels in Safari.

Site Isolation is a critical browser security feature that strengthens defenses against modern threats like speculative execution attacks and cross-origin data leakage. By isolating each site into its own process, browsers achieve better data protection and containment. While it increases resource consumption, the trade-off is worthwhile for environments requiring strong security, especially in enterprise or cloud applications.