Avoiding Zombies in Archival Replay Using ServiceWorker

Sawood Alam, Mat Kelly, Michele C. Weigle, and Michael L. Nelson
Department of Computer Science, Old Dominion University
Norfolk, Virginia, USA - 23529
{salam,mkelly,mweigle,mln}@cs.odu.edu

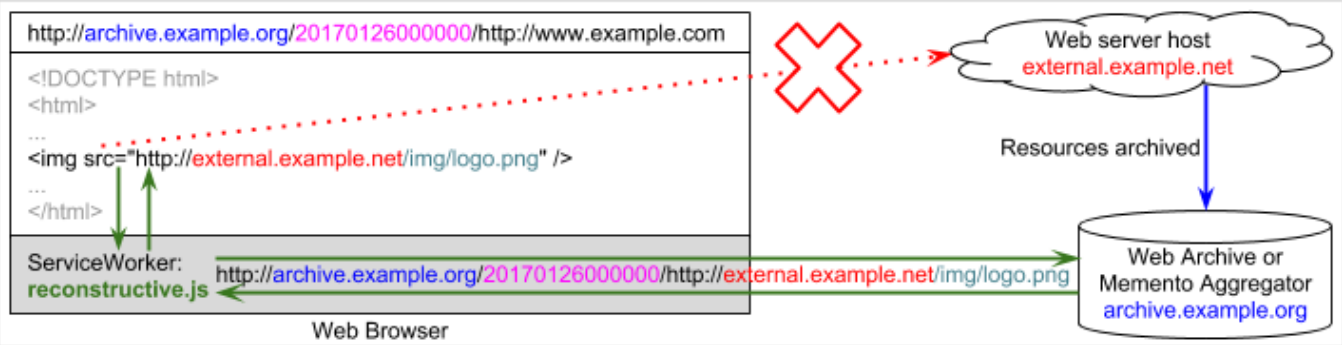

Figure 1: ServiceWorker reconstructive.js intercepts a zombie resource and reroutes to its archived copy.

A *Composite Memento* is an archived representation of a web page with all the page requisites such as images and stylesheets. All embedded resources have their own URIs, hence, they are archived independently. For a meaningful archival replay, it is important to load all the page requisites from the archive within the temporal neighborhood of the base HTML page. To achieve this goal, archival replay systems try to rewrite all the resource references to appropriate archived versions before serving HTML, CSS, or JS. However, an effective server-side URL rewriting is difficult when URLs are generated dynamically using JavaScript. A failure of correct URL rewriting might yield an invalid/unintended URI or resolve to a live resource. Such live resources, leaking in a composite memento, are called "*zombies*".

ServiceWorker (SW) is a new client-side web API [2] that can be used to intercept all the network requests for embedded resources originating from web pages in its scope. We use SW API to reconstruct composite mementos from the originally captured data without any URL rewriting. By intercepting requests on the client-side, we are essentially rerouting instead of rewriting. Rerouting is an effective mechanism to block zombies, as it takes effect when the user-agent resolves a reference. Figure 1 illustrates how our SW implementation, `reconstructive.js`[1], intercepts a live leakage and reroutes it correctly to the corresponding archived copy.

To evaluate the archival replay reconstruction quality we created the Archival Capture Replay Test Suite (ACRTS)[2] with different scenarios of how a web page might initiate a network request. We archived ACRTS and saved the resulting Web ARChive (WARC) file. We then changed the live ACRTS site in a way that all the resource references remained the same, but their content was changed. Using various replay systems we loaded the archived ACRTS from the stored WARC file. Depending on how effective the replay system is, it might load resources from the archive (☑), leak from the live site (☒), or not load at all (☐). The latter might happen either because the requested resource was not present in the archive or the replay system resolved the location incorrectly. Table 1 shows how well each of the listed archival replay systems reconstructs a composite memento when resource requests are originated from different conditions. A more extensive description and evaluation of this work is published in Alam et al. [1].

[1]https://github.com/oduwsdl/reconstructive
[2]https://ibnesayeed.github.io/acrts/

Table 1: URL Rewriting/Rerouting Results in Different Archival Replay Systems (A: OpenWayback, B: PyWB, C: Memento Reconstruct, D: Memento for Chrome, and E: Reconstructive)

Resource Loading Scenarios	ABCDE
Relative path	☑☑☑☑☑
Absolute rooted path	☑☑☑☑☑
Absolute local URL	☑☑☑☑☑
Absolute external URL	☑☑☑☑☑
External resource from an external iframe	☑☑☑☑☑
Loaded by an inline CSS	☑☑☑☑☑
Loaded by a CSS file	☑☑☑☑☑
Loaded by CSS @font-face	☑☑☐☑☑
Loaded by image srcset	☑☑☑☑☑
Added by an inline JS on page load	☒☑☑☒☑
Added by an inline JS on page scroll	☒☑☐☒☑
Added by an inline JS on click	☒☑☑☒☑
Added by a JS file	☒☑☑☒☑
Added by an Ajax request	☒☑☐☒☑

REFERENCES

[1] Sawood Alam, Mat Kelly, Michele C. Weigle, and Michael L. Nelson. 2017. Client-side Reconstruction of Composite Mementos Using ServiceWorker. In *Proceedings of the 17th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '17)*.
[2] Alex Russell, Jungkee Song, and Jake Archibald. 2015. Service Workers. https://www.w3.org/TR/service-workers/