Workerounds - Categorizing Service Worker Attacks and Mitigations

Team Students: Karthika Subramani (UGA), Jordan Jueckstock (NCSU) | Professors: Roberto Perdisci (UGA, GATech), Alexandros Kaprevalos (NCSU)

Introduction

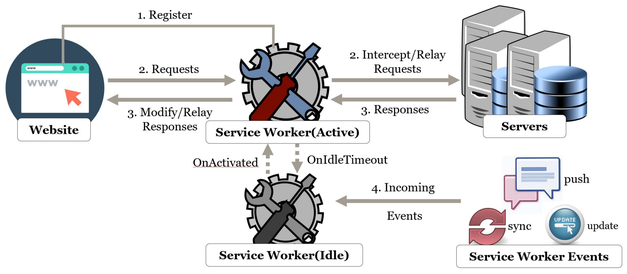
Service Workers (SWs) are a powerful feature at the core of Progressive Web Apps that can continue to function when a user's device is offline and have access to device sensors and capabilities previously accessible only by native applications. Researchers have found ways in which SWs may be abused to achieve different malicious purposes. In this project, we reproduce and analyze known attack vectors related to SWs and explore new abuse paths that have not previously been considered. We systematize the attacks into different categories, discuss unmitigated open SW security problems and propose dynamic defenses to further improve SW security.

Problem Statement

Because SWs are a powerful feature, browser developers are mindful of potential security risks that come with them and have implemented certain security policies to limit SW abuse. However, researchers have proven that SWs can still be abused to build a web-based botnet, launch DDoS attacks, cryptomining, XSS and side-channel attacks and Social Engineering attacks such as phishing and malvertising. To successfully defend against existing unmitigated attacks and future attacks, we systematically categorize known and new attack vectors and propose defenses to improve SW security.

Service Worker & its Policies

In practice, a SW is a JavaScript Worker script with the following high-level properties: (i) it is installed by a web application (ii) after installation, the SW can act as a proxy for network requests issued by its web application (iii) it is an event-driven process that runs in the background and can be activated by events such as push notifications or fetch requests even if the web application is not active in the browser.
An overview of SW life cycle



Browsers enforce the following major policies for SW security:

1. Only secure origins (HTTPS sites) can register SWs.
2. The JavaScript file containing SW code must be hosted under the same origin as the website that registers the SW.
3. A SW should be terminated if the SW code has been idle for more than 30 seconds or if an event takes more than 5 minutes to process.
4. Push notifications should trigger a user-visible notification if the SW does not explicitly issue one.
5. The use of some APIs (e.g., Periodic Background Sync) should be restricted by permissions that must be granted by the browser (not necessarily via a direct UI request to the user).

SW Abuse Categories

First, we group the attacks discovered via our literature review, as well as new attacks found, into different categories based on the root SW features that make them possible. The chart below displays the SW abuse categories. The attacks under each category and the corresponding APIs they exploit. Further, it also contains detailed information on the major browser versions that are affected by each attacks and the versions the attacks were mitigated. As can be seen, despite the existing SW security policies, a number of attacks were still possible and are largely unmitigated. Further, we create a testbed of new attacks and known attacks by reproducing them that can be used to check if a browser is vulnerable to these attacks.

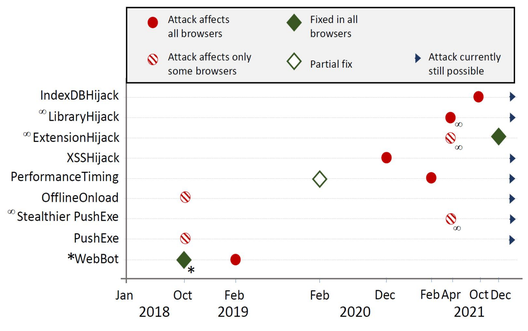
		Abuse Vectors									Browsers				
		Push API	Notification API	Sync API	Performance API	Update API	ImportScripts	*iframe* inclusion	3rd-party code	IndexDB API	Chrome	Firefox	Edge	Safari	Opera
Continuous Execution	WebBot			✓		✓					● v69.0 ○ v70.0	● v57.0 ◐ v60.0	- ○ v80.0	✡	● v56.0 ○ v57.0
	PushExe	✓									✡	● v59.0 ⊠	✡	✡	✡
	Stealthier PushExe [New]	✓	✓								● v85.0 ⊠	✡	● v.85.0 ⊠	✡	● v71.0 ⊠
Side Channel	Offline Onload							✓			✡	● v59.0 ⊠	✡	● v11.1 ⊠	✡
	Performance Timing1				✓			✓			● v79.0 ○ v83.0	● v72.0 ◐ v73.0	● v79.0 ○ v83.0	● v12.1 ◐ v14.0	● v66.0 ○ v69.0
	Performance Timing2				✓			✓			● v79.0 ⊠	● v72.0 ⊠	● v79.0 ⊠	● v12.1 ⊠	● v66.0 ⊠
Hijacking	XSS						✓		✓		⊠	⊠	⊠	⊠	⊠
	Extension Hijack [New]								✓		✡	● v82.0 ○ v95.0	✡	✡	✡
	Library Hijack [New]						✓		✓		⊠	⊠	⊠	⊠	⊠
	IndexDB Hijack						✓		✓	✓	● v55.0 ⊠	● v52.0 ⊠	● v80.0 ⊠	● v11.1 ⊠	● v42.0 ⊠
Push API & Notifications Abuse	Malvertising [New]	✓	✓								☑	☑	☑	☑	☑
	Phishing	✓	✓								☑	☑	☑	☑	☑
	Stalkerware	✓									☑	☑	☑	☑	☑

Legend: (●) first attack impact; (○) fix released; (◐) partial fix released; (⊠) no fix released; (☑) possible if notifications are supported; (✡) attack not possible

Existing Mitigations

To understand when the attacks were discovered and if/when any mitigation was employed, we create a timeline of the attack discovery and mitigation in the shown figure. Although certain additional mitigations were introduced to curb specific issues, they do not address the source of the issue and still leads to more open problems as discussed in the next section. In addition to the browser policies, the following mitigations were implemented

- **Termination Delay Limits:** This was introduced to stop the WebBot attack and prevents SWs from delaying their own termination on self-update.
- **Notification UI changes:** In lieu of web notifications abuse, browsers changed the UI for requesting notifications making it less intrusive.
- **Default Notifications :** Some browsers display notifications with default message in case the SW was activated in the background and no notification was displayed to the use explicitly.
- **Event Signaling :** In order to prevent side-channel attacks that leverage the invocation of page events, more standard approach was used.
- **Site Isolation :** This is to ensure that 3rd party iframes have limited access to the main frame's data with respect to SWs.



This work will be published in Euro S&P,2022. Source Code : https://github.com/karthikaS03/SW_Sec_Project

SW Open Problems

Based on our study of SW and its abuse, we highlight open problems that are yet to be addressed and propose mitigations that could be incorporated in browsers.

- **Limiting SW Execution:** One of the major problems is to limit SWs from running in the background for longer duration while its web application is inactive. Therefore, we propose to dynamically monitor SWs and their usage and apply heuristics based on real-time measurements to terminate their abuse automatically.
- **Limiting Malicious SW Permissions:** Adversaries could use the recommended practice of 'Double Permission' and Social Engineering approaches to render 'Quiet Notification' futile. Therefore, we propose that notifications are monitored by browsers in real-time and alert the users or browsers of their abuse to warrant further actions.
- **Limiting 3rd Party Code:** Considering that most websites leverage 3rd party services and grant them access to their SWs unknowing of the extent of their access, additional browser policies need to be mandated to make developers aware of the danger of including 3rd party SW scripts without any restrictions.

SW Behavior in-the-wild

We instrument Chromium browser to collect and record all SW related data. Then, we use this instrumented browser to crawl top websites as per Alexa ranking to monitor their behavior and arrive at reasonable thresholds that can be used to detect and limit malicious SW behavior in real-time. The table below shows a number of measurement for 90% of the websites in different bands of ranking.

Event Count	No. of SW Origins	#SW Origins above threshold value 90%						
		Threshold Value	B-1	B-2	B-3	B-4	B-5	B-6
Push Count per Hour	518	14	0	1	5	11	10	22
Push Count per Day	518	38	1	2	5	13	9	43
Third Party Fetch Count per Activation	416	1	12	10	13	35	30	44
SW Execution Time Per Activation	761	3	2	1	6	16	16	38
SW Execution Time Per Day	761	64	1	2	7	9	10	33

SW Browser Policies

To demonstrate that implementing the new policies proposed is possible with reasonable effort, we implement a proof-of-concept in the Chromium browser of some of those policies. For example, we successfully monitor SW execution lifetime and generate alerts when it exceeds a certain threshold.

Acknowledgements

This material is based in part upon work supported by the National Science Foundation (NSF) under grants No. CNS-2126641 and CNS-2047260.

References

- P. Papadopoulos, P. Ilia, M. Polychronakis, E. Markatos, S. Ioannidis, and G. Vasiliadis, "Master of web puppets: Abusing web browsers for persistent and stealthy computation," ArXiv, vol. abs/1810.00464, 2019
- J. Lee, H. Kim, J. Park, I. Shin, and S. Son, "Pride and prejudice in progressive web apps: Abusing native app-like features in web applications," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018.
- P. Chinprutthiwong, R. Vardhan, G. Yang, and G. Gu, "Security study of service worker cross-site scripting," in Annual Computer Security Applications Conference, ser. ACSAC '20. New York, NY, USA: Association for Computing Machinery, 2020.
- S. Karami, P. Ilia, and J. Polakis, "Awakening the web's sleeper agents: Misusing service workers for privacy leakage," in Network and Distributed System Security Symposium (NDSS), 2021
- K. Subramani, X. Yuan, O. Setayeshfar, P. Vadrevu, K. H. Lee, and R. Perdisci, "When push comes to ads: Measuring the rise of (malicious) push advertising," in Proceedings of the ACM Internet Measurement Conference, ser. IMC '20.
- Chinprutthiwong, R. Vardhan, G. Yang, Y. Zhang, and G. Gu, "The service worker hiding in your browser: The next web attack target?" in 24th International Symposium on Research in Attacks, Intrusions and Defenses, ser. RAID '21.
- Watanabe, E. Shioji, M. Akiyama, and T. Mori, "Melting pot of origins: Compromising the intermediary web services that rehost websites," in NDSS, 2020.

National Science Foundation