Progressive Web Apps are advertised to have the capabilities of an installed application and the ease/reach of websites. If this is the next thing in our application evolution, what does this mean for security testing? This article gives some details of specific tests that a security assessor can incorporate into his/her toolset.

What are PWA?

One of my contacts, who I shall call "A", has been an advocate of using Progressive Web Apps (PWA) over native Mobile Applications. He has been sharing links via WhatsApp such as a Medium blog post on PWA's on iOS (2 years ago) to this post from the NotebookCheck website (this year). Sites such as PWA Stats advertise that moving to PWA led to more revenue, conversions and active users. This led me to wonder – will PWA overtake native mobile applications? Is this the new thing to learn for pentest?

Progressive Web Applications rely on a new generation of web browser APIs to grant web applications the capabilities formerly in the domain of installed applications. These capabilities build on relatively new standards of web assembly and modern browser APIs (e.g. Web Share API). While native applications had such capabilities, they require users to download for every single update to the application (taking up bandwidth and disk space). In addition, there is the problem of maintaining two code bases (with at least two teams to maintain them).

Other benefits that PWAs have include:

- not needing to go through the process of getting approved on different app stores
- responsiveness and work with many different screen sizes
- smooth, fast and lightweight
- work offline, unlike legacy web applications
- discoverable via search engines (which have a lot larger audience than app stores)

Security considerations

There are some improvements in security as a PWA and more opportunities for security testers to find vulnerabilities. PWA has improvements such as requiring Secure Context as a feature (e.g. communicate only using encrypted TLS). Requiring Secure Context means that the use of an authenticated and encrypted channel is required – since service workers (a core component of PWA) will not work without it.

Building an app as a PWA does bring about several security drawbacks compared to mobile applications and I will list three of them below.

(1) Missing mobile application protections

There are security controls in mobile applications that are missing in a PWA. One of them is the TLS certificate pinning. Certificate pinning allows the mobile application to only trust a certain public key in TLS/SSL connections; such a mobile application will not trust other public keys, even if signed by a certificate authority trusted by the operating system. For more information, you can refer to the “Certificate and Public Key Pinning” OWASP technical guide.

(2) Missing manifest leading to HTML injection

Web app manifest is a JSON file to inform the browser about the application, how the PWA should appear when “installed” on the device (e.g. app icon) and other behaviours. When no manifest is enabled, an attacker that spots a HTML injection vulnerability (due to lack of output encoding and input validation of JavaScript) can exploit it by linking a malicious manifest and then submitting the injected payload. Compare this behaviour with the default disabled JavaScript behaviour of Android.

(3) Threat of malicious service workers

PWAs face the threat of malicious service workers browsing the web application as a victim. There are two ways of doing this. The first is to exploit existing Cross-Site Scripting and file upload vulnerabilities to upload a malicious service worker. The service worker can be controlled via the Shadow Worker C2 and proxy framework. The second is to trick the user to run malicious scripts.

One interesting case was submitted to HackerOne is the hijack of Augur (prediction platform built on Ethereum). In this case, the dormant malicious service worker could be installed prior to installation of Augur. After Augur is installed, network traffic between Augur client and the server could be modified by the malicious service worker at the attacker’s bidding.

Conclusion

No matter what platform is desired by businesses, infosec teams will be expected to be able to secure them. For security assessors, this means understanding the various threats and recommending appropriate controls for them.

References

I. https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps
II. https://web.dev/what-are-pwas/
III. https://medium.com/awebdeveloper/pwa-is-future-of-mobile-how-to-tame-it-855dd42df0ec
IV. https://yoast.com/what-is-a-progressive-web-app-pwa/
V. https://blog.nviso.eu/2020/01/16/deep-dive-into-the-security-of-progressive-web-apps/
VI. https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts/features_restricted_to_secure_contexts
VII. https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning
VIII. https://shadow-workers.github.io/