November 2021

Web Workers for Software Attestation

Will Drewry

Radoslav Vasilev

Web Workers for Software Attestation

ABSTRACT

Theft of browser authentication cookies is a serious security problem. Cookies stolen, e.g., by copying from disk and transmitting to another machine the cookie jar of a web browser, can grant substantial, unauthorized account access to the thief. This disclosure describes techniques to protect cookies by creating a web platform based isolated service for an authentication domain. The isolated service creates an ongoing measurement of the on-origin actions taken to provide an attestation that the browser session in use is the one that the server has been interacting with. The isolated service runs as part of the browser using web workers, which is tamperproof from normal web platform APIs and does not depend on on-disk data. The techniques provide a generic mechanism for binding user sessions to a given browser without relying on explicit device identity in cryptographically deprived environments. By avoiding disk persistence of cookies, the techniques thwart both web and memory attacks.

KEYWORDS

- Browser cookie
- One-time cookie
- Cross-site request forgery (XSRF)
- WebAuthn
- User authentication
- Service worker
- Web worker
- Browser session
- Session authentication
- One time password (OTP)
- Hash-based message authentication code (HMAC)
- HMAC-based OTP
- Time-based OTP

BACKGROUND

Web browsers store cookies to enable website access and reloading by the browser user across process restarts. Cookies are stored on a backing store such as a disk or other storage device. With such a setup, if a malicious actor is able to execute malware in the user context, it is possible for the cookie to be collected and used on other devices controlled by the malicious actor. Cookie theft can be performed by binary malware that simply copies the browser cookie jar from the disk or other storage device. Since cookies are bearer tokens for web application use, access to a user's cookies can grant substantial and unauthorized power of access (login) to the malicious actor who steals the cookie. The harmful effects of unauthorized access can include account theft, data loss, etc.

Various web services use the browser to store cookies to provide long-lived credentials relating to an identifiable session, authenticated or not. Depending on their usage, cookies may last only for the life of the browser task or may be written to disk. Regardless, updates to a cookie, and therefore session information, require the browser to visit, reload, or continue on the web origin that created the cookie.

Risk heuristics can be deployed to detect cookie misuse and upon detection, the user can be requested to re-authenticate with a knowledge factor or other shared secret. However, such re-authentication procedures can be costly and cumbersome. Detection of cookie theft in (near) real time is challenging since it depends on effectively binding a cookie to the device of issuance and being able to validate proof of binding on subsequent transactions.

Cookies (or other) data on disk can be protected by having a separate, trusted-service store such that access via the filesystem is infeasible by normal users or by the browser user session. Alternatively, cookie data can be protected from theft by using encryption services

provided by the operating system (OS), by using obfuscation tactics, etc. Attestation techniques, used in content protection, game consoles, browsers, OS, etc., assert either information about the state of the operating system or assert that specific cryptographic key matter is guaranteed to be protected by specified policy compliant software and hardware configurations.

A web worker (or service worker) is a script that runs in the background of a browser. Web workers are separate from a web page and enable features that do not need a web page or user interaction.

DESCRIPTION

This disclosure describes techniques to protect cookie data by creating a web platform-based isolated service for an authentication domain (e.g., the accounts domain of web services). The techniques are implemented with user permission.

With user permission, the isolated service (also known as service worker, which runs in a protected space of the browser) creates an ongoing measurement of the on-origin actions taken by the user to provide attestation that the browser session in use is the same one that the server has been interacting with. The isolated service runs as part of the browser using web workers, which cannot be tampered with using normal web platform application programming interfaces (APIs) and which do not depend on on-disk data.

Per the techniques, a service worker is instantiated for each web service or website. The service worker is seeded with user and session-specific data, with specific user permission. This data and subsequent site-specific events are used to compute rolling datasets. The service worker presents an interface that enables the initial seeding and future data mix-ins, as well as queries against its data that yield ephemeral attestation. For example, the worker can run a hash-based message authentication code (HMAC) or time-based one-time password procedure. Additionally,

if the site uses the service worker as a network proxy, all requests can be annotated with an assertion.

Cookie theft is detected by creating a session-bound attestation mechanism that isn't solely dependent on other symmetric secrets, like device fingerprinting. Even if the service worker is initiated with symmetric secrets, the longevity of the binding and continual advancement of the session state increases the burden for cookie theft/reuse to proceed undetected. Indeed, even a fork in the worker state can become apparent if the server tracks its session cookie and its session service worker states in tandem. In such a case, cross-site request forgery (XSRF) protections are also provided, where the service worker is queried for an XSRF token variant of its data (versus appended as a proxy).

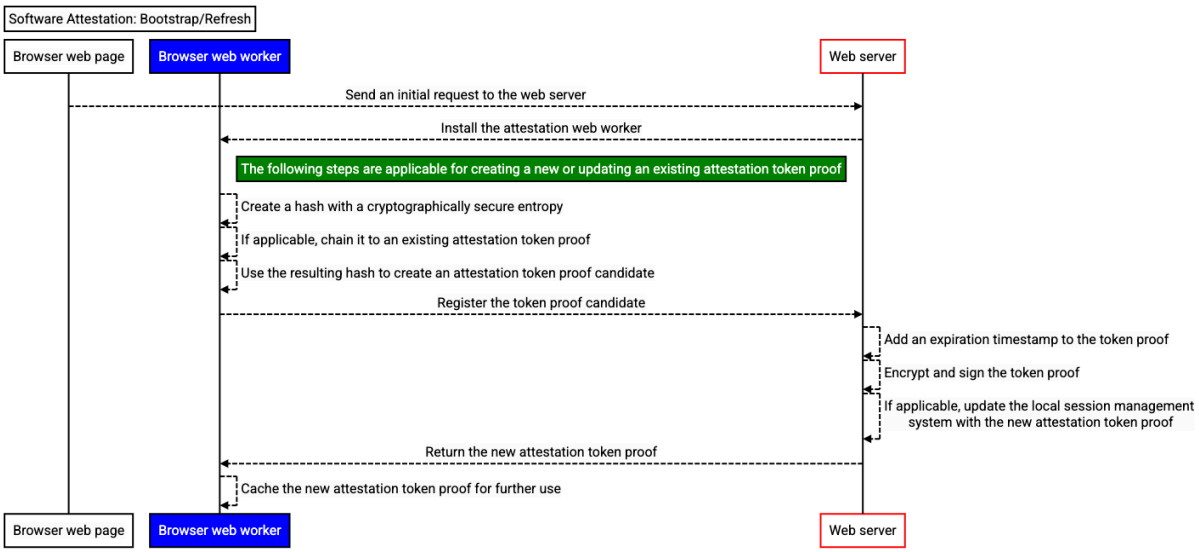

Fig. 1: Software attestation using web workers: bootstrap or refresh

Fig. 1 illustrates an example sequence of interactions between a browser web page, a web worker, and a web server to perform software attestation during bootstrap or refresh, e.g., creating a new attestation token proof or updating an existing one. The browser sends an initial request to the server. The server installs the attestation web worker. The web worker is seeded

with a hashed token (which is a secret shared between the worker and the server) that comes through an encrypted session cookie.

The hashed token can also be derived from or signed by a hardware security key based, e.g., on the WebAuthn standard or another similar standard. The worker periodically regenerates the token using hash chaining such that the new token is a hash of a random event timestamp, a cryptographically random number from the machine, and a previous hash (Merkle tree). The new hash (token proof) is signed with the old token, e.g., the one inside the cookie. The new hash is registered on the server and the session cookie thereby updated, e.g., the token proof is augmented with an expiration timestamp, encrypted, and signed. The web server returns the new attestation token proof to the web worker. The procedure is repeated as necessary.

Effectively, a state is built within the web worker through the use of hash chaining such that parts of the input are machine specific, e.g., secure crypto-random. Frequent updating of the hash makes it difficult for a token to be used with a different device and prevents the building of a session.

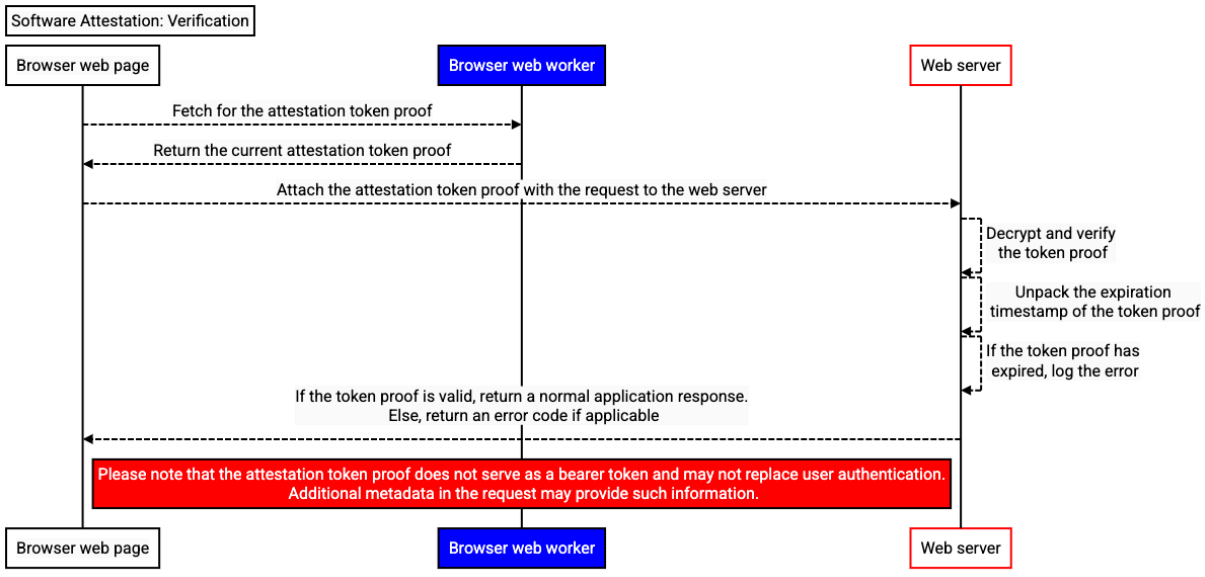

Fig. 2: Software attestation using web workers: Verification

Fig. 2 illustrates an example sequence of interactions between a browser web page, a web worker, and a web server to verify software attestation. The browser web page fetches the attestation token proof from the web worker and transmits it to the web server for purposes of verification. The web server decrypts and verifies the token proof, unpacks its expiration timestamp, and, if the token proof is determined as valid, returns a normal application response to the browser. If the token proof is determined to be invalid or expired, an error is logged and returned to the browser.

In this manner, the techniques described herein secure browser-only cookies, local or session data by saving them in the protected memory space of the browser rather than writing them to disk. With such a configuration, an attacker who attempts to take over the user's computer has to examine its memory, including protected browser memory, to find the cookies, a very time-consuming procedure. If at all an attacker is able to discover and access the cookies, the attestation token of the browser would've changed and rendered obsolete, thereby foiling the attack.

The techniques can be used by any website that can create web workers or similarly isolated, long-lived jobs. Similarly, a browser or user agent can natively include the described, memory-only, state transition based attestation. The techniques effectively provide a generic mechanism for binding user sessions to a given browser without relying on explicit device identity in cryptographically deprived environments, e.g., without requiring hardware-backed secure zones. The techniques avoid disk persistence of cookies, thereby thwarting web attacks that attempt to extract internal states and/or memory attacks to extract original state, while maintaining a low client-interaction latency.

Further to the descriptions above, a user is provided with controls allowing the user to

make an election as to both if and when systems, programs or features described herein may enable collection of user information (e.g., information about a user's browser, a user's preferences, or a user's current location), and if the user is sent content or communications from a server. In addition, certain data may be treated in one or more ways before it is stored or used, so that personally identifiable information is removed. For example, a user's identity may be treated so that no personally identifiable information can be determined for the user. Thus, the user has control over what information is collected about the user, how that information is used, and what information is provided to the user.

CONCLUSION

This disclosure describes techniques to protect cookies by creating a web platform based isolated service for an authentication domain. The isolated service creates an ongoing measurement of the on-origin actions taken to provide an attestation that the browser session in use is the one that the server has been interacting with. The isolated service runs as part of the browser using web workers, which is tamperproof from normal web platform APIs and does not depend on on-disk data. The techniques provide a generic mechanism for binding user sessions to a given browser without relying on explicit device identity in cryptographically deprived environments. By avoiding disk persistence of cookies, the techniques thwart both web and memory attacks.

REFERENCES

[1] Dacosta, Italo, Saurabh Chakradeo, Mustaque Ahamad, and Patrick Traynor. "One-time cookies: Preventing session hijacking attacks with stateless authentication tokens." *ACM Transactions on Internet Technology (TOIT)* 12, no. 1 (2012): 1-24.

[2] Bugliesi, Michele, Stefano Calzavara, Riccardo Focardi, and Wilayat Khan. "CookiExt: Patching the browser against session hijacking attacks." *Journal of Computer Security* 23, no. 4 (2015): 509-537.

[3] "Session management with service workers | Firebase Documentation" available online at https://firebase.google.com/docs/auth/web/service-worker-sessions

[4] "Goodbye Short Sessions: A Proposal for Using Service Workers to Improve Cookie Management on the Web" available online at https://developers.google.com/web/updates/2016/06/2-cookie-handoff