فرهاد امان ۹۹۳۱۰۰۶

۱-
پراکسی فوروارد، که به عنوان پراکسی وب نیز شناخته می‌شود، از طرف کاربران برای دسترسی به منابع موجود در اینترنت عمل می‌کند. هنگامی که کاربری درخواستی برای دسترسی به یک صفحه وب یا منابع دیگر ارسال می‌کند، این درخواست ابتدا به سرور پراکسی فوروارد ارسال می‌شود. سپس، سرور پراکسی فوروارد این درخواست را به سرور مقصد منتقل می‌کند. در این حالت، سرور مقصد درخواست را مثل اینکه که از سوی سرور پراکسی ارسال شده تلقی می‌کند و نه از کاربر اصلی. به این ترتیب، هویت کاربر از سرور واقعی پنهان می‌ماند.

علاوه بر مخفی کردن هویت کاربران، پراکسی‌های فوروارد کاربردهای دیگری نیز دارند:

- کش کردن: این پراکسی‌ها می‌توانند برای کاهش استفاده از پهنای باند و بهبود زمان پاسخ‌دهی استفاده شوند.
- کنترل دسترسی و فیلتر محتوا: پراکسی‌های فوروارد می‌توانند برای محدود کردن یا فیلتر کردن دسترسی کاربران به اینترنت به کار روند.
- توازن بار: پراکسی‌های فوروارد می‌توانند درخواست‌های کاربران را بین چندین سرور پشتیبان توزیع کنند تا بار را متعادل سازند و عملکرد را بهبود بخشند. سرور پراکسی کش Apache Traffic Server یک سرور پراکسی کش متن‌باز است که می‌توان به عنوان یک پراکسی فوروارد برای کاهش استفاده از پهنای باند به کار گرفت.

پراکسی معکوس یک سرور است که بین دستگاه‌های کاربر و یک وب سرور قرار می‌گیرد. این سرور برای مدیریت درخواست‌های ورودی کاربران و توزیع آن‌ها به سرورهای پشتیبان مناسب استفاده می‌شود، که اغلب برای تعادل بار، امنیت و بهینه‌سازی عملکرد به کار می‌رود. برخلاف پراکسی فوروارد که برای حفاظت از کاربران به کار می‌رود، پراکسی معکوس برای حفاظت از سرورها استفاده می‌شود. پراکسی معکوس، سروری است که درخواستی از کاربر دریافت می‌کند، آن را به یکی از چندین سرور دیگر ارجاع می‌دهد و نتیجه را از سروری که واقعاً درخواست را پردازش کرده است، بازمی‌گرداند. بنابراین، سرور

پشتیبان واقعی از کاربر پنهان می‌ماند، زیرا کاربر فقط مستقیماً با پراکسی معکوس ارتباط برقرار می‌کند. کاربردهای دیگر پراکسی معکوس عبارتند از:

- کش کردن: پراکسی‌های معکوس می‌توانند محتوای استاتیک را کش کرده و ارائه دهند، که این امر بار روی سرورهای پشتیبان را کاهش می‌دهد.
- تعادل بار: یک روش مهم در مدیریت ترافیک شبکه و سرورها است که به توزیع یکنواخت بار کاری بین چندین سرور یا منابع شبکه کمک می‌کند. هدف اصلی از تعادل بار این است که از هیچ یک از سرورها به صورت بیش از حد استفاده نشود و از طرفی هیچ سروری بیکار نماند. این کار باعث بهبود عملکرد و افزایش قابلیت اطمینان سیستم‌ها می‌شود.
- امنیت: پراکسی‌های معکوس می‌توانند با بررسی و فیلتر کردن ترافیک ورودی، یک لایه امنیتی اضافی ایجاد کنند و در برابر حملات مختلف مانند حملات DDoS محافظت کنند.
- Termination SSL/TLS: پراکسی‌های معکوس می‌توانند رمزگذاری و رمزگشایی SSL/TLS را به نمایندگی از سرورهای پشتیبان انجام دهند، و این وظیفه‌ی سنگین را از آن‌ها بردارند. Nginx یک وب سرور و سرور پراکسی معکوس متن‌باز با پرفورمنس بالا است. این وب سرور به دلیل قابلیت‌های مقیاس‌پذیری، تعادل بار و قابلیت‌های Termination SSL/TLS، به طور گسترده‌ای استفاده می‌شود.

۲-

الف) portal.aut.ac.ir/aportal/index.html

ب) HTTP/1.1

ج) Keep-Alive نشان دهنده اتصال دائمی به سرور است. Keep-Alive: 300 و مدت آن ۳۰۰ ثانیه است.

د) User-Agent نشان دهنده ابزار مورد استفاده برای اتصال است. که در اینجا سیستم‌عامل مک و مرورگر سافاری است.

۳-

الف)

2xx نشان دهنده موفقیت آمیز بودن درخواست است.

200: OK

201: Created

4xx نشان دهنده مشکل سمت کاربر است.

400: bad request

401: unauthorized

403: forbidden

404: not found

405: method not allowed

ب) مکانیزم کش کردن پراکسی به سیاست‌های مدیریت کش آن بستگی دارد که شامل انقضای کش، اعتبارسنجی کش و غیره می‌شود.

- Cache hit اتفاق بیفتد و محتوا بروز باشد: اگر کش پراکسی نسخه‌ای کش شده از فایل "index.html" داشته باشد که هنوز تازه است و اطلاعات موجود در درخواست قدیمی‌تر از تاریخ آخرین تغییرات باشد، پراکسی می‌تواند مستقیماً با استفاده از محتوای کش شده به کاربر پاسخ دهد.
- Cache hit اتفاق بیفتد و محتوا منقضی شده باشد: اگر کش پراکسی نسخه‌ای کش شده از "index.html" داشته باشد ولی منقضی شده باشد، و تاریخ "If-Modified-Since" قدیمی‌تر از تاریخ آخرین تغییرات محتوای کش شده باشد، پراکسی ممکن است محتوای منقضی شده را به کاربر ارائه دهد در حالی که

همزمان یک درخواست مشروط به سرور اصلی برای اعتبارسنجی کپی کش شده ارسال می‌کند. اگر سرور اصلی با وضعیت "Not Modified 304" پاسخ دهد، پراکسی می‌تواند کپی کش شده را به‌روزرسانی کند. اما اگر سرور اصلی با وضعیت "200" پاسخ دهد، پراکسی باید محتوا را با نسخه جدیدتر که در بدنه پاسخ قرار دارد، به‌روز کند.

- Cache Miss اتفاق: اگر کش پراکسی نسخه‌ای کش شده از "index.html" نداشته باشد یا تاریخ "If-Modified-Since" جدیدتر از تاریخ آخرین تغییرات محتوای کش شده باشد، پراکسی درخواست را به سرور اصلی ارجاع می‌دهد. سرور اصلی سپس با محتوای کامل "index.html" پاسخ می‌دهد، و پراکسی پاسخ جدید را برای درخواست‌های آتی در کش ذخیره می‌کند."

۴-

الف)

- پردازش درخواست‌ها و پاسخ‌ها به صورت ترتیبی: اگر یک درخواست به دلیل ازدحام مسدود شود، می‌تواند تحویل درخواست‌های بعدی را به تأخیر اندازد. در مقابل، HTTP/2 قادر است با استفاده از یک اتصال TCP تکی، چندین جریان داده را بدون مسدود کردن منابع دیگر همزمان ارسال کند.
- تأخیر بالا: مشکلی دیگر که در HTTP/1 وجود دارد، تأخیر بالا در ارسال و دریافت داده‌ها است.
- عدم پشتیبانی از Server Push

HTTP/1 از Server Push پشتیبانی نمی‌کند، که در آن سرور می‌تواند منابع را قبل از درخواست کاربر به او ارسال کند.

- فشرده‌سازی هدر: در حالی که HTTP/1 از فشرده‌سازی پشتیبانی می‌کند، اما همیشه به صورت مؤثر استفاده نمی‌شود.

ب) هنوز تعداد زیادی وب‌سایت وجود دارند که با استفاده از HTTP/1 توسعه یافته‌اند. گذار به نسخه‌های جدیدتر HTTP نیازمند به‌روزرسانی‌های قابل توجه است و ممکن است برای سیستم‌ها یا وب‌سایت‌های قدیمی‌تر امکان‌پذیر نباشد.

HTTP/1 همچنین در پیاده‌سازی بسیار آسان‌تر است و درک آن بسیار ساده است. این پروتکل برای سال‌ها استاندارد بوده و توسط تمام مرورگرهای وب و وب‌سرورها به طور گسترده پشتیبانی می‌شود. گذار به پروتکل‌های جدیدتر شامل پیچیدگی‌ها و تغییرات بزرگ در پیاده‌سازی‌ها می‌باشد.

علاوه بر این، برخی وب‌سایت‌ها به گونه‌ای طراحی شده‌اند که سبک باشند و عملکرد آن‌ها بین HTTP/1 و HTTP/2 تفاوت چندانی ندارد، بنابراین تغییر به پروتکل‌های جدیدتر ممکن است بسیار حیاتی نباشد.

۵-

الف)

- Stateful Connection: در این نوع ارتباط، سرور اطلاعات مربوط به وضعیت Session کاربر را در طی تعاملات متعدد حفظ می‌کند. این بدان معناست که سرور درخواست‌های قبلی کاربر را به یاد دارد و می‌تواند از این اطلاعات برای درک درخواست فعلی استفاده کند. ارتباطات Stateful اغلب در برنامه‌هایی که نیازمند حفظ داده‌های خاص Session مانند بازی‌های آنلاین استفاده می‌شوند.
  نمونه‌هایی از پروتکل‌های Stateful شامل FTP و SSH هستند.
- Stateless Connection: در یک ارتباط Stateless هر درخواست کاربر به عنوان یک تراکنش مستقل و جدا در نظر گرفته می‌شود. سرور هیچ اطلاعاتی در مورد تعاملات یا Session قبلی کاربر ذخیره نمی‌کند. ارتباطات Stateless به طور معمول در پروتکل‌های ساده درخواست-پاسخ مانند HTTP استفاده می‌شوند.
  زمانی که مقیاس‌پذیری، تعادل بار و تحمل خطا مهم هستند، ارتباطات Stateless انتخاب می‌شوند.

ب) برای ایجاد ارتباط Stateful در HTTP، از کوکی‌ها استفاده می‌شود. کوکی‌ها قطعات کوچکی از داده هستند که سرور به مرورگر کاربر ارسال می‌کند و مرورگر آن‌ها را به صورت محلی ذخیره می‌کند. وقتی کاربر وب‌سایتی را بازدید می‌کند، سرور هدر Set-Cookie را در پاسخ HTTP ارسال می‌کند. این هدر شامل یک کوکی است که شامل شناسه Session منحصر به فرد می‌باشد. اگر کاربر کوکی‌ها را بپذیرد، یک هدر کوکی در پیام درخواست تنظیم می‌شود سپس، سرور می‌تواند کاربر را با کوکی‌هایش شناسایی کند و وضعیت کاربر را حفظ نماید.

کوکی‌ها نقش مهمی در فراهم کردن تجربه کاربری بهتر و شخصی‌سازی محتوا دارند. برای مثال، آن‌ها اطلاعاتی مانند تنظیمات زبان، سبد خرید، و ورود به حساب کاربری را حفظ می‌کنند. همچنین، کوکی‌ها در تحلیل وب‌سایت‌ها و ردیابی رفتار کاربران برای بهبود عملکرد و ارائه تبلیغات هدفمند کاربرد دارند. با این حال، استفاده از کوکی‌ها مسائل حریم خصوصی را نیز به همراه دارد. وب‌سایت‌ها موظفند از کاربران رضایت بگیرند قبل از ذخیره‌سازی کوکی‌ها در دستگاه‌های آن‌ها و باید اطلاعات دقیقی درباره نحوه استفاده و مدیریت کوکی‌ها ارائه دهند.

۶-

الف) Forward Proxy یا Caching Proxy

این نوع پروکسی به عنوان یک نقطه واسط بین کلاینت‌های درخواست‌کننده (مانند سیستم‌های دانشگاه) و سرورهای مرجع (پکیج منیجر سرور) عمل می‌کند. از Forward Proxy برای دریافت پکیج‌ها از سرورهای مرکزی استفاده می‌شود، و اگر پکیج‌هایی قبلاً دریافت شده باشند، از قابلیت Caching Proxy برای ارائه سریعتر پکیج‌ها به سیستم‌های درخواست‌کننده استفاده می‌شود.

ب) Reverse Proxy

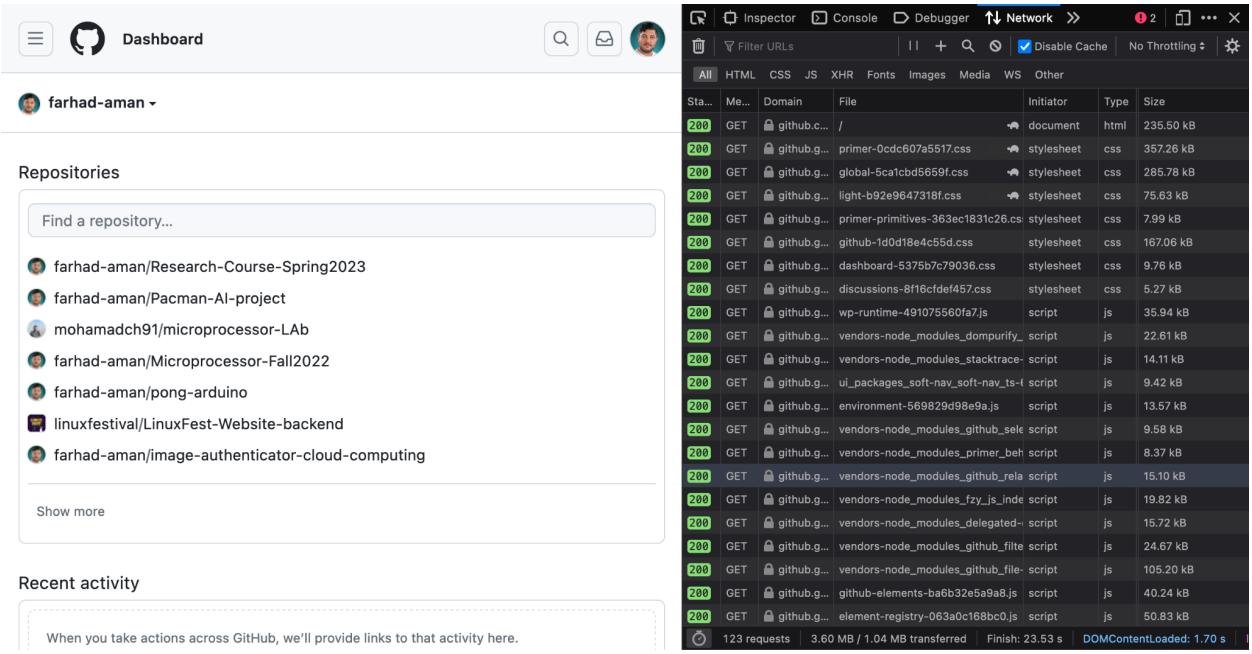
این نوع پروکسی به عنوان یک نقطه واسط بین کلاینت‌ها و سرورهای مرکزی عمل می‌کند. از Reverse Proxy با توزیع بار جغرافیایی برای هدایت درخواست‌های کلاینت‌ها

به سرورهای محلی یا فراهم‌کنندگان محتوا در ناحیه جغرافیایی مشخص استفاده می‌شود.

ج) Reverse Proxy

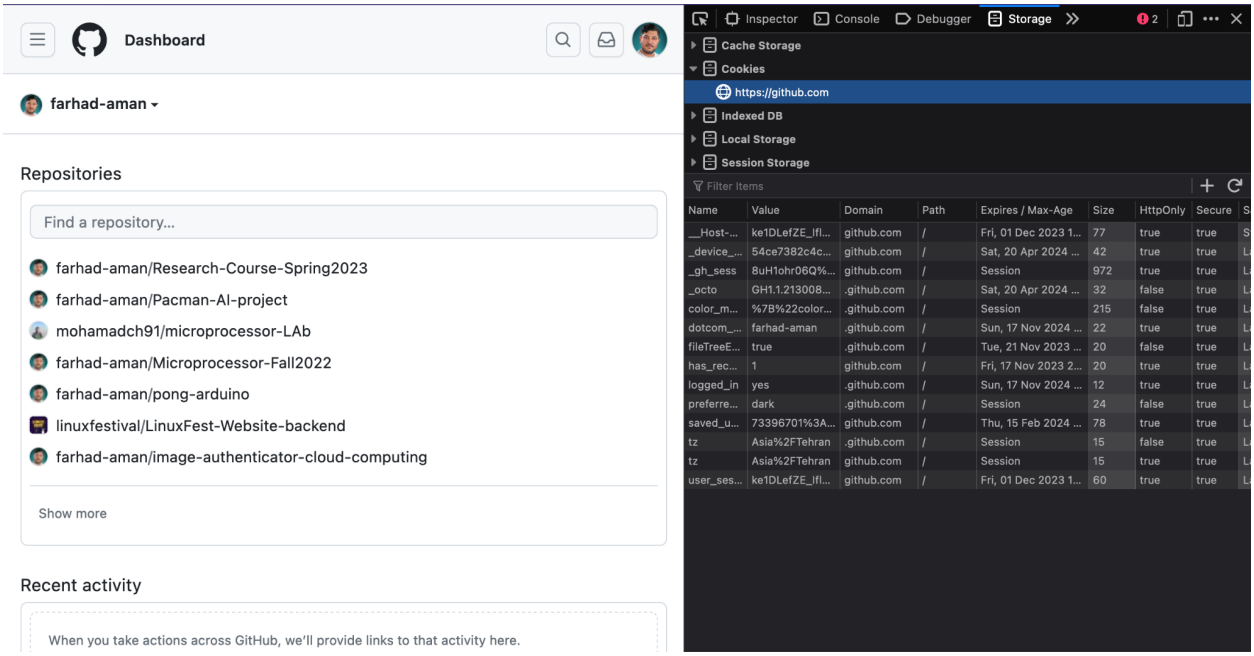
در این جا یک پروکسی ریورس نزدیک به سرور قدیمی قرار دارد درخواست ابتدا به ریورس پروکسی رسیده و کار احراز هویت انجام می‌شود سپس نتایج به سرور اصلی می‌رسند.

بخش عملی:



با ورود به بخش شبکه (Network Tab) مرورگر، ما می‌توانیم تمامی درخواست‌هایی را که برای بارگذاری صفحه اصلی گیت‌هاب انجام شده‌اند، مشاهده کنیم. ابتدا یک درخواست GET برای دریافت HTML پایه‌ی صفحه ارسال می‌شود. سپس، درخواست‌های GET دیگری برای دریافت استایل‌شیت‌ها و تصاویر وب‌سایت انجام می‌گیرد. در نهایت، تکه‌های مربوط به Webpack از سرور درخواست داده می‌شوند.

این روند نشان‌دهنده‌ی مکانیزم عمومی بارگذاری وب‌سایت‌ها در مرورگر است. درخواست‌های GET برای بارگذاری منابع مختلف مانند جاوااسکریپت، فونت‌ها و داده‌های API نیز ممکن است انجام شود. مدیریت این درخواست‌ها و بهینه‌سازی آن‌ها برای کاهش زمان بارگذاری و بهبود تجربه‌ی کاربری اهمیت زیادی دارد. همچنین، استفاده از فناوری‌هایی مانند کش‌سازی و تاخیر در بارگذاری (Lazy Loading) می‌تواند در کارایی وب‌سایت تاثیر مثبتی داشته باشد.



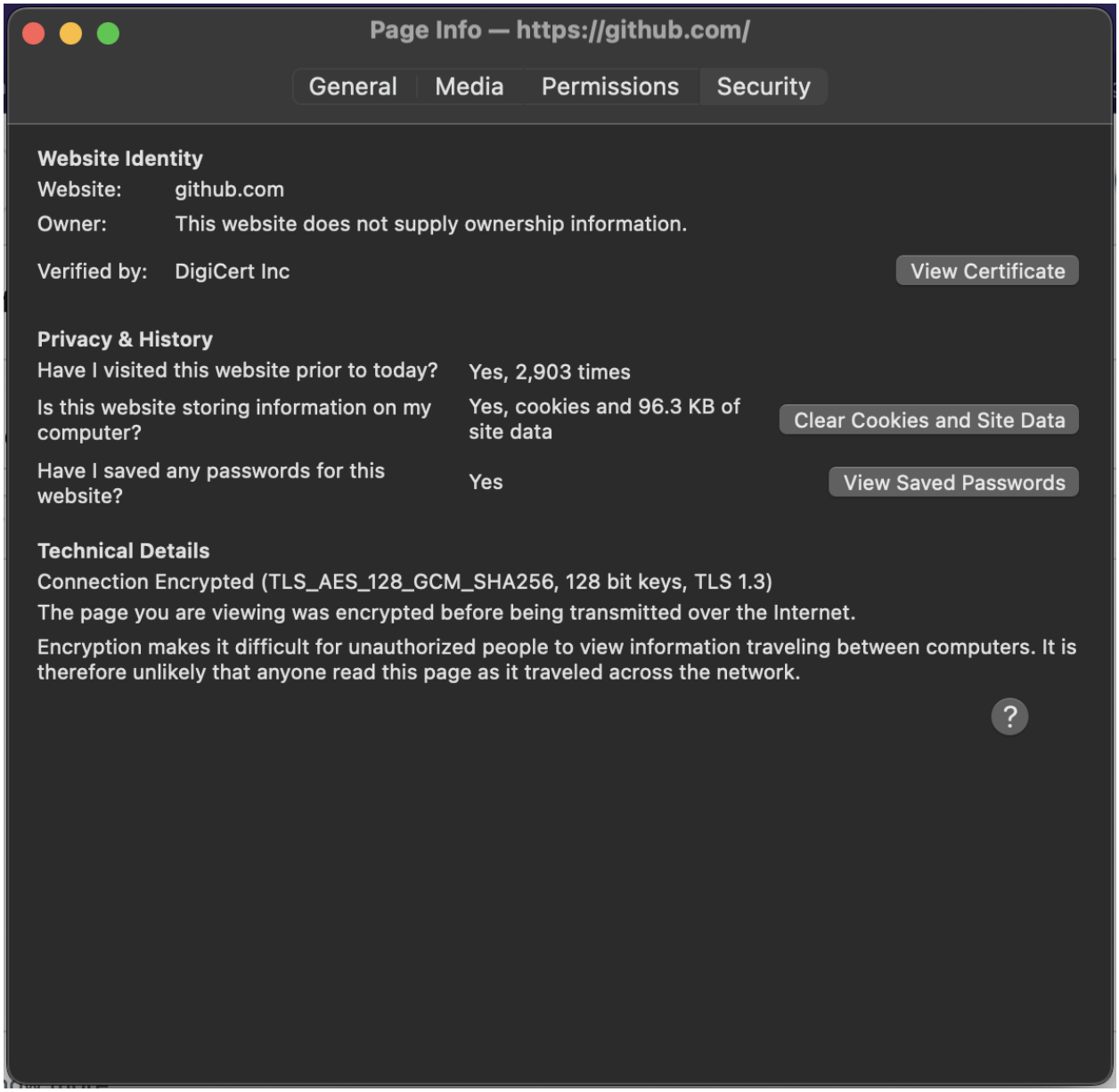
کوکی‌هایی که برای وب‌سایت تنظیم شده‌اند، درون زبانه‌ی Storage قابل دسترسی هستند. ما می‌توانیم ببینیم که داده‌های مرتبط با کاربر مانند preferred_color_mode برای تم وب‌سایت تنظیم شده است. سایر کوکی‌ها شامل color_mode، logged_in و user_session و غیره می‌باشند.

کوکی‌ها فایل‌های کوچکی هستند که توسط وب‌سایت‌ها بر روی دستگاه کاربران ذخیره می‌شوند تا اطلاعاتی مانند تنظیمات کاربری، وضعیت ورود به سیستم و سایر تنظیمات را به خاطر بسپارند. این امر به وب‌سایت‌ها این امکان را می‌دهد تا تجربه‌ی شخصی‌تر و مؤثرتری را برای کاربران فراهم کنند. برخی از کوکی‌ها می‌توانند برای پیگیری رفتار کاربران

در سایت‌های مختلف به منظور نمایش تبلیغات هدفمند استفاده شوند. همچنین، امنیت در استفاده از کوکی‌ها بسیار مهم است تا از سوء استفاده و دسترسی غیرمجاز به داده‌های حساس جلوگیری شود.

گیت‌هاب کاربران خود را با استفاده از نام کاربری و رمز عبور هنگام ورود به سیستم تأیید هویت می‌کند و با تنظیم یک کوکی "user_session" که می‌تواند در هدر هر فراخوانی API ارسال شود (برای مثال، اگر بخواهیم یکی از مخازن خود را مشاهده کنیم، این توکن درون فیلد هدر کوکی فرستاده می‌شود).

تأیید هویت و مجوز دسترسی در گیت‌هاب به منظور حفاظت از حساب‌های کاربری و داده‌ها بسیار مهم است. تأیید هویت از طریق نام کاربری و رمز عبور اولین گام در این روند است. کوکی "user_session" که پس از ورود به سیستم تنظیم می‌شود، به گیت‌هاب این امکان را می‌دهد که هر درخواست API را به یک کاربر خاص مرتبط کند، که این امر امنیت را در هنگام دسترسی به منابع حساس تقویت می‌کند.



نسخه TLS در بخش More Information مرورگر فایرفاکس قابل مشاهده است که TLS 1.3 است.

TLS یا امنیت لایه حمل (Transport Layer Security) یک پروتکل استاندارد برای ایجاد ارتباطات اینترنتی امن است. نسخه 1.3 از این پروتکل، جدیدترین و پیشرفته‌ترین نسخه‌ی موجود است که امنیت و کارایی بهتری را نسبت به نسخه‌های قبلی فراهم می‌کند. این نسخه بهبودهایی را در رمزنگاری، کاهش زمان برقراری ارتباط و کاهش پیچیدگی پروتکل به همراه دارد.

با استفاده از TLS 1.3، داده‌های مبادله شده بین کاربر و سرور به طور موثری رمزگذاری می‌شوند، که این امر به حفاظت از حریم خصوصی و امنیت اطلاعات کمک می‌کند.
همچنین، این نسخه به دلیل کاهش دست‌داده‌های (Handshakes) مورد نیاز برای برقراری ارتباط، زمان بارگذاری صفحات وب را بهبود می‌بخشد. این ویژگی‌ها TLS 1.3 را به یک انتخاب محبوب برای وب‌سایت‌هایی تبدیل کرده که به دنبال ارائه تجربه‌ای سریع‌تر و امن‌تر برای کاربران خود هستند.