Dr.-Ing. Mario Heiderich, Cure53
Wilmersdorfer Str. 106
D 10629 Berlin
cure53.de · mario@cure53.de

Pentest-Report ExpressVPN VPN Browser Extension 05.2024

Cure53, Dr.-Ing. M. Heiderich, M. Wege, MSc. J. Moritz

Index

Introduction

> *"Go online safely and securely with strong VPN encryption. Spoof your location and control the ExpressVPN app from your Google Chrome browser."*

From https://www.expressvpn.com/vpn-download/chrome-vpn

This report describes the results of a penetration test and source code audit against the ExpressVPN VPN browser extension for Firefox and Chrome.

To give some context regarding the assignment's origination and composition, ExpressVPN contacted Cure53 in April 2024. The test execution was scheduled for June 2024, namely in CW24. A total of six days were invested to reach the coverage expected for this project, and a team of three senior testers were assigned to its preparation, execution, and finalization.

The methodology conformed to a white-box strategy, whereby assistive materials such as sources, test-user credentials, as well as all further means of access required to complete the tests were provided to facilitate the undertakings.

The work was structured using a single work package (WP), defined as:

- **WP1**: Source-code assisted penetration tests against ExpressVPN VPN browser extension

Note that this was not the first time the ExpressVPN VPN browser extension was tested by Cure53. It was already the focus of an audit held in September and October 2022 (see EXP-12).

All preparations were completed in early June 2024, specifically during CW21, to ensure a smooth start for Cure53. Communication throughout the test was conducted through a dedicated and shared Slack channel, established to combine the teams of ExpressVPN and Cure53. All personnel involved from both parties were invited to participate in this channel. Communications were smooth, with few questions requiring clarification. The scope was well-defined and clear, and no significant roadblocks were encountered during the test. Cure53 provided frequent status updates through the aforementioned Slack channel. Live reporting was not specifically requested for this audit.

The Cure53 team achieved good coverage over the scope item, and identified a total of two findings. Of the two security-related discoveries, one was classified as a security vulnerability, and one was categorized as a general weakness, both with low exploitation potential.

The overall number of findings made during this engagement was very small, and this can certainly be interpreted as a positive sign in regards to the security of the inspected VPN browser extension. Furthermore, it can be positively acknowledged that the severity of the findings did not exceed *Medium*, which showcases the extension's well-implemented security measures, which protect against the majority of severe threats. All in all, Cure53 would like to congratulate the ExpressVPN team on their excellent work.

The report will now shed more light on the scope and testing setup, and will provide a comprehensive breakdown of the available materials. Next, the report will detail the *Test Methodology* used in this exercise. This chapter will show which areas of the software in scope have been covered, and what tests have been executed, despite the limited number of findings made during the course of the exercise. Following this, the report will list all findings identified in chronological order, starting with the *Identified Vulnerabilities*, and followed by the *Miscellaneous Issues* unearthed. Each finding will be accompanied by a technical description, Proof-of-Concepts (PoCs) where applicable, plus any fix or preventative advice to action.

In summation, the report will finalize with a *Conclusions* chapter in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the ExpressVPN VPN browser extension.

Scope

- **Source code audits & security assessments against ExpressVPN VPN browser extension**
    - **WP1:** Source-code assisted penetration tests against ExpressVPN VPN browser extension
        - **Primary scope item:**
            - *xv_chrome*
        - **VPN Browser Extension:**
            - A .ZIP archive of the source code was provided
            - Commit: 0b01c67e27cfb569ebd1ee2c546d9cdefb2482e2
            - Version: 6.0.7.6160
        - **Out of scope:**
            - Any build dependencies or build scripts found in the code
            - Any code and dependency used for tests (i.e. mocks, end to end (e2e) tests)
            - Any third-party dependencies included within the browser extension
            - Source code not relevant to the browser extension, targeting other platforms (e.g. iOS, Android, Windows, Aircove Router, Linux, MacOS)
            - Testing of the API servers
            - VPN servers and the individual protocol implementations (both Lightway and OpenVPN)
            - The ExpressVPN client application and its components
            - Chromium-related geolocation spoofing weaknesses
    - **Test User Credentials**
        - U: mike@cure53.de
        - U: johannes@cure53.de
    - **Test-supporting material was shared with Cure53**
    - **All relevant sources were shared with Cure53**

Test Methodology

This section documents the testing methodology applied during this engagement. Steps taken by the testers during the assessment are enumerated in detail, to ensure that the ExpressVPN team can tailor their future efforts accordingly. Notes on coverage and methods are organized by topic.

VPN Browser Extension

The security assessment of the ExpressVPN browser extension commenced with an examination of the content script. This script injects JavaScript code into all webpages, to hook functions of the Geolocation API.

The analysis effectively uncovered a bypass that could be exploited to reveal a user's true location (see EXP-14-001). The content script was also evaluated for other client-side security weaknesses. However, due to the absence of external user input processing within the script, no further attack vectors were discovered.

Following the examination of the content script, the security audit proceeded with an in-depth analysis of the web-accessible resources. Particular attention was directed towards the *networkLock.html* file, as it can be embedded by other web pages. This analysis prioritized the validation procedures employed for user input within the context of this file. It could be confirmed that the user input handling was implemented appropriately for its intended use. Notably, the validation still maintained proper mitigation of the previously identified Clickjacking vulnerability (EXP-12-001), as the team had maintained the fix since that assessment.

The Chrome extension utilizes the *externally_connectable* property with a wildcard configuration, granting unrestricted access to all web pages. This configuration permits any web page to establish a connection with the background script and subsequently retrieve information regarding the currently selected VPN location within the extension. This introduces an unnecessary information leakage, as described in EXP-14-002.

A meticulous review of the Vue.js components was conducted, focusing on the potential exploitation of well-known JavaScript sinks such as *innerHTML*, *location.assign*, or *eval*. Additionally, the code was reviewed for common Vue.js-specific sinks, such as the *v-html* directive. Either these security-sensitive functions were entirely absent from the codebase, or were not employed in conjunction with user input. This approach significantly mitigates the risk of malicious injection attacks.

The security audit proceeded with an in-depth analysis of the background script, with particular focus directed towards the native communication mechanism employed to transmit messages to the native ExpressVPN application.

This analysis focused on the potential for a malicious extension to exploit this communication channel and take control of the VPN. Fortunately, the review yielded positive results, as no such vulnerabilities were identified.

Finally, the extension configuration and build files were examined for misconfigurations or outdated dependencies. However, no weaknesses were found in this area.

Severity Glossary

The following section details the varying severity levels assigned to issues discovered in this report.

Critical: The highest possible severity level. Categorizes issues that allow attackers to achieve extensive access to sensitive areas, such as critical systems, applications, data or other pertinent components in scope.

High: Categorizes issues that allow attackers to achieve limited access to sensitive areas in scope. This also includes issues with limited exploitability that can facilitate a significant impact upon the target in scope.

Medium: Categorizes issues that do not incur major impact on the areas in scope. Additionally, issues requiring a more limited exploitation are graded as *Medium*.

Low: Categorizes issues that have a highly limited impact on the areas in scope. Mostly does not depend on the level of exploitation but rather on the minor severity of obtainable information or lower grade of damage targeting the areas in scope.

Info: Categorizes issues considered merely informational in nature. They are mostly considered as hardening recommendations, or improvements that can generally enhance the security posture of the areas in scope.

Identified Vulnerabilities

The following section lists all vulnerabilities and implementation issues identified during the testing period. Notably, findings are cited in chronological order, rather than by degree of impact, with the severity rank offered in brackets following the title heading for each vulnerability. Furthermore, all tickets are given a unique identifier (e.g., EXP-14-001) to facilitate any future follow-up correspondence.

EXP-14-001 WP1: Location spoofing bypass via callback function *(Medium)*

***Fix Note**: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

CVSS Score: 5.1
CVSS String: 4.0/AV:N/AC:L/AT:N/PR:N/UI:A/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N
CWE: https://cwe.mitre.org/data/definitions/200.html

The ExpressVPN browser extension includes a feature called *Spoof your location,* which aims to conceal the user's actual location when queried through the browser's Geolocation API. This is achieved by injecting a script into all webpages, which overrides the *getCurrentPosition* and *watchPosition* functions of the Geolocation API. If this feature is enabled, the modified functions return the fake coordinates corresponding to the VPN location the user is connected to.

However, during the review, a method was discovered that could bypass this feature and retrieve the user's real location.

Both Firefox and Chrome use Google Location Services to determine a user's location by sending the computer's IP address and information about nearby wireless access points, among other data[1]. Testing confirmed that devices with an enabled WiFi adapter might leak the real location, despite an active VPN connection and enabled location spoofing feature. However, the user must give explicit consent to each webpage to access the real location. Therefore, this issue was only assigned a medium severity rating.

In the following, a Proof-of-Concept is shown that illustrates how the *Spoof your location* feature can be bypassed.

Both hooked functions receive as first argument a callback function. This callback is executed in the context of *hookedObj* which holds a reference to the original *getCurrentPosition* and *watchPosition* functions. Therefore the original functions can be called within the callback to access the real location. Further, the *fakeGeo* property can be set to *false* to disable the hook for subsequent calls.

[1] https://support.mozilla.org/en-US/kb/does-firefox-share-my-location-websites?redirectslug=[...]=en-US

PoC script:

```
<script>
   setTimeout(function() {
        navigator.geolocation.getCurrentPosition(
            function(pos) {
                // fake position
                console.log(pos);
                // real position
                this.getCurrentPosition(console.log);
                //disable the hooking
                this.fakeGeo=false;
            })
    }, 1000);
</script>
```

The following code excerpt shows that the user-supplied callback function is executed in the context of the *hookedObj*, which holds a reference to the original Geolocation API functions.

Affected file:
xv_chrome/xv_chrome/source/scripts/content/gps.js

Affected code:

```
function waitGetCurrentPosition() {
    if ((typeof hookedObj.fakeGeo !== 'undefined')) {
        if (hookedObj.fakeGeo === true) {
              hookedObj.tmp_successCallback({
                     coords: {
                            latitude: hookedObj.genLat,
                            longitude: hookedObj.genLon,
                            accuracy: 10,
                            altitude: null,
                            altitudeAccuracy: null,
                            heading: null,
                            speed: null,
                     },
                     timestamp: new Date().getTime(),
              });
[...]
```

It is recommended to make the original Geolocation API functions inaccessible to potentially malicious web pages. Users should also be made aware that consenting to the Geolocation API can leak their real location, even with the *Spoof your location* feature enabled.

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit, but which may assist an attacker in successfully achieving malicious objectives in the future. Most of these results are vulnerable code snippets that did not provide an easy method by which to be called. Conclusively, while a vulnerability is present, an exploit may not always be possible.

EXP-14-002 WP1: Leakage of selected location setting in Chrome *(Info)*

***Fix Note**: The issue was addressed by the ExpressVPN team and the fix was verified by Cure53 who were able to review the related diff & PR. The issue no longer exists.*

CVSS Score: 0.0
CVSS String: CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:N/VI:N/VA:N/SC:N/SI:N/SA:N
CWE: https://cwe.mitre.org/data/definitions/200.html

The Chrome extension utilizes the *externally_connectable* property with a wildcard configuration, granting unrestricted access to all web pages. This configuration permits any web page to establish a connection with the background script and subsequently retrieve information regarding the currently selected VPN location within the extension. While connected to the VPN, the user's location can be readily ascertained by leveraging the assigned IP address.

However, an information leak arises when the user is disconnected from the VPN. In such a scenario, a malicious website could exploit this issue to leak the location setting. This leaked information could encompass the user's most recently connected location, or even predict their probable future connection location.

To access the currently selected location, it is sufficient to embed the following script in a web page. The coordinates of the selected location will be printed to the browser console.

PoC script:

```
<script>
  chrome.runtime.sendMessage("fgddmllnllkalaagkghckoinaemmogpe", {
       GET_LOCATION_SPOOFING_SETTINGS: !0
  }, (response => {
       console.log(response)
  }));
</script>
```

The following code excerpt shows the handler that processes messages from potentially malicious web pages. This handler returns the currently selected location.

Affected file:
xv_chrome/source/scripts/background.js

Affected code:

```
chrome.runtime.onMessageExternal.addListener((request, sender,
sendResponse) => {
  sendResponse({
       coords: this.currentInfo.selectedLocation?.coords,
       fakeIt:
  ['connected', 'connecting', 'disconnecting', 'reconnecting',
'connection_error'].includes(this.currentInfo.state) &&
  this.prefs.hideLocation,
});
```

To enhance user privacy and prevent unintended location disclosure, it is recommended to modify the extension's behavior so that the selected VPN location is not returned to web pages when the user is disconnected from the VPN.

Conclusions

As noted in the *Introduction*, this June 2024 penetration test and source code audit carried out by Cure53 comprehensively assessed the security posture of the ExpressVPN browser extension for Chrome and Firefox. This rigorous review identified a single *Medium* severity vulnerability, along with an additional weakness categorized as *Info*.

Both of the identified security concerns are directly related to the functionality of the Spoof your location feature. The first issue (see EXP-14-001) pertains to the potential for a user's real location to be leaked even when the Spoof your location feature is actively enabled. While malicious websites would still require explicit user permission to access the current location, the activated Spoof your location feature could create a false sense of security for users, potentially leading them to inadvertently grant the access required.

The second identified weakness involved the potential for malicious web pages to disclose the user's currently selected VPN location within the browser extension (see EXP-14-002). While resolving this issue will further improve user privacy, it is worth noting that the security impact was assessed as *Info* only.

The limited number of vulnerabilities identified during this engagement can be attributed to two key factors. Firstly, the extension's design prioritizes a minimal attack surface, which reduces potential avenues for exploitation. Secondly, the way the development team has adhered to best practices for browser extension development is commendable. This focus on secure coding principles, coupled with the implementation of robust input validation measures, significantly reduces the likelihood of successful attacks. The selection of the well-established Vue.js framework, recognized for its secure default configurations, also demonstrably enhanced the overall security posture of the extension.

Resolving both of the issues identified herein will further solidify the already robust security posture of the VPN extension, ensuring continued user privacy and protection.

Overall, Cure53 can conclude that the security posture of the ExpressVPN VPN browser extension for Firefox and Chrome is good. It is clear that the developers are capable in this area, and have made a lot of excellent decisions.

Cure53 would like to thank Brian Schirmacher and Harsh Shah from the ExpressVPN team for their excellent project coordination, support and assistance, both before and during this assignment.