Bitwarden Browser Extension Report

ISSUE SUMMARIES, IMPACT ANALYSIS, AND RESOLUTION

BITWARDEN, INC

Table of Contents

Summary

In August 2023, Bitwarden engaged with cybersecurity firm Cure53 to perform penetration testing and a dedicated audit of the Bitwarden browser extension. A team of two senior testers from Cure53 were tasked with preparing and executing the audit over two days to reach total coverage of the system under review.

Three issues were discovered during the audit. Two issues were resolved post-assessment. One informational-only issue is under planning and research.

This report was prepared by the Bitwarden team to cover the scope and impact of the issues found during the assessment and their resolution steps. For completeness and transparency, a copy of the report delivered by Cure53 has also been attached to this report.

Issues

BWN-08-011 WP2: Autofill functionality leaked via race condition in login (Medium)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/clients/pull/6700

A race condition was identified that could potentially facilitate filling of credentials on a different origin than where the original request was triggered. This race condition is very difficult to trigger naturally, but does exist as a potential method of exposing login credentials to an undesirable source. In order to trigger this race condition, an on-page script would need to redirect the user immediately after the user has triggered autofill.

The resolution to this issue came by checking if the `window.location.href` where the autofill page details were gathered is equivalent to the `window.location.href` where a fill is happening. If not, we return early ignoring the request to autofill the page.

BWN-08-020 WP2: Public suffix list incorrectly detected (High)

Status: Issue was fixed post-assessment.

Pull requests:

- https://github.com/bitwarden/clients/pull/6735

Public suffix values are not being properly parsed from a domain within the application. This sets up a situation where credentials for a website could be autofilled into another website that is not directly related. An example of where this could happen is with GitHub Pages that use the `github.io` TLD. Because those pages can be generated dynamically by any user, the actual domain should be treated as `<user-defined-domain>.github.io`. Instead, the application matches any URIs that contain `github.io` domains as the same domain, even if the item is only set for the `<user-defined-domain>.github.io` domain.

The changes made fix the issue, ensuring that we keep in consideration private domains, as per the Public Suffix List, when matching URIs for autofill.

BWN-08-019 WP2: Credentials-stuffing via Clickjacking on notification bar (Info)

Status: Issue is under planning and research and will involve a refactor of the notification bar.

Fine penetration tests for fine websites

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

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, each ticket has been given a unique identifier (e.g., *BWN-08-011*) to facilitate any future follow-up correspondence.

BWN-08-011 WP2: Autofill functionality leaked via race condition in login *(Medium)*

The Bitwarden browser extension offers the possibility to automatically fill in any stored credentials for the currently loaded domain. However, by changing the domain loaded in the currently active tab, it is possible to leak these credentials to an attacker-controlled domain.

The retrieval of stored credentials entails multiple messages between the content and background script. In the end, the background script retrieves the stored credentials and sends a *"fillForm"* command to the content script. As the background script uses the currently active tab to determine the page that should receive the message, it verifies that the tab's URL continues to match the domain it originally received the credentials request for.

By carefully timing an origin change, it is sometimes possible to load a new URL after the completed check but before sending credentials. As the content script does not have an additional origin check, it fills the received credentials into the displayed form, therefore leaking it to the attacker-controlled domain.

It must be noted that the time window to win this race is not only small but highly inconsistent, therefore making it highly improbable to leak credentials on every attempt. To verify this behavior, the background script was altered to add an artificial delay, so as to ensure that the race can be won.

Affected code:
clients-browser-v2023.7.1/apps/browser/src/autofill/services/autofill.service.ts

Affected file:
```
async doAutoFill(options: AutoFillOptions): Promise<string> {
    const tab = options.tab;
    [...]
    await Promise.all(
      options.pageDetails.map(async (pd) => {
        // make sure we're still on correct tab
        if (pd.tab.id !== tab.id || pd.tab.url !== tab.url) {
          return;
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
    }
[...]
BrowserApi.tabSendMessage(
      tab,
      {
        command: "fillForm",
        fillScript: fillScript,
        url: tab.url,
      },
      { frameId: pd.frameId }
    );
```

Affected file:
clients-browser-v2023.7.1/apps/browser/src/autofill/content/autofillv2.ts

Affected code:

```
chrome.runtime.onMessage.addListener(function (msg, sender, sendResponse) {
  [...]
  } else if (msg.command === "fillForm") {
    /* no further URL check */
    fill(document, msg.fillScript);
    sendResponse();
    return true;
  }
```

Despite this issue being highly difficult to exploit, it should be taken into consideration to offer an additional origin check in the *"fillForm"* command of the content script. As the received message structure already contains the URL associated with the sent credentials, it would be straightforward to compare this property against the currently loaded domain.

BWN-08-020 WP2: Public suffix list incorrectly detected *(High)*

The Bitwarden browser extension utilizes the *tldts*[1] library to parse and correctly detect the domain and subdomain parts of a loaded URL. However, the library fails to detect domains that are present in the public suffix list[2]. This leads to an incorrect domain part being used by the extension. Therefore, it will display or even autofill credentials that incorrectly match the domain in case the feature is activated.

The issue can be shown by using *github.io* as an example. As this domain is present on the public suffix list, it should be treated similarly to other TLDs. This is necessary as users are allowed to create *github.io* subdomains to host their own content.

[1] https://github.com/remusao/tldts
[2] https://publicsuffix.org/

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

As such, they should be treated as different domains to ensure they are properly separated. Yet, as shown below, this is not the case and any credentials stored for *test.github.io* will be utilized and potentially leaked to any other **.github.io* domain. Please note that other domains on the public suffix lists suffer from the same behavior.

PoC code:

```
const { parse } = require('tldts');

/*
Correctly detects that test is not a subdomain of co.at
co.at is on the public suffix list as it is a second level domain
*/
testCase1 = parse("test.co.at")
console.log(`Domain: ${testCase1.domain}`)
console.log(`Subdomain: ${testCase1.subdomain}`)

/*
Incorrectly detects that test is a subdomain of github.io
Github.io is on the public suffix list so the test case
should have a.github.io as the domain name
but it returns github.io instead
*/
testCase2 = parse("test.github.io")
console.log(`Domain: ${testCase2.domain}`)
console.log(`Subdomain: ${testCase2.subdomain}`)
```

Output

```
Domain: test.co.at
Subdomain:
Domain: github.io
Subdomain: test
```

Affected file:

clients-browser-v2023.7.1/libs/common/src/platform/misc/utils.ts

Affected code:

```
static getDomain(uriString: string): string {
    if (Utils.isNullOrWhitespace(uriString)) {
      return null;
    }

    uriString = uriString.trim();

    if (uriString.startsWith("data:")) {
      return null;
    }
```

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

```
  if (uriString.startsWith("about:")) {
    return null;
  }

  try {
    const parseResult = parse(uriString, { validHosts: this.validHosts });
    if (parseResult != null && parseResult.hostname != null) {
      if (parseResult.hostname === "localhost" || parseResult.isIp) {
        return parseResult.hostname;
      }

      if (parseResult.domain != null) {
        return parseResult.domain;
      }
      return null;
    }
  } catch {
    return null;
  }
  return null;
}
```

Cure53 recommends informing the developer of the library in order to have this issue addressed. The library should correctly detect all items present in the public suffix list. As a temporary hot-fix, the default URL-matching could be changed from domain to host, as the latter concerns the full hostname and is not affected by this behavior.

Dr.-Ing. Mario Heiderich, Cure53
Bielefelder Str. 14
D 10709 Berlin
cure53.de · mario@cure53.de

Miscellaneous Issues

This section covers any and all noteworthy findings that did not incur an exploit but 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, whilst a vulnerability is present, an exploit may not always be possible.

BWN-08-019 WP2: Credentials-stuffing via Clickjacking on notification bar *(Info)*

The Bitwarden browser extension displays a notification bar to allow the user to save credentials for an origin, with the prompt appearing after the credentials have been submitted via a form. To make this notification bar accessible in a web page, Bitwarden exposes it via the *web_accessible_resources* manifest key.

To display the resource, it is simply loaded by the content script via an iframe. A malicious website can abuse this design. Once data is automatically submitted, a form to trigger the *save* mechanism appears. The website's JavaScript can simply remove the iframe, which displays the notification bar, choosing to load the notification bar again. In this scenario, the utilized iframe becomes obscured by custom CSS, as it hides the presence of the iframe but it still receives events like clicks.

This approach is a so-called Clickjacking attack[3]. By carefully crafting the iframe and moving it to the current cursor position, the likelihood of the user unintentionally clicking on *save* can be increased. Therefore, storing arbitrary credentials for the attacker's domain cannot be at all excluded.

Affected file:
clients-browser-v2023.7.1/apps/browser/src/manifest.json

Affected code:

```
"web_accessible_resources": [
    "notification/bar.html",
    "images/icon38.png",
    "images/icon38_locked.png"
  ],
```

After a successful exploitation of this vulnerability, only an unintended set of credentials would be stored in the WebExtension, which means that the risk of this issue could be accepted. Bitwarden should consider creating the necessary "*Save*" and "*Edit*" buttons via the injected content scripts instead of relying on a web-accessible resource. This would prevent the threat of Clickjacking.

[3] https://portswigger.net/web-security/clickjacking