Extension Checker

Targeted extension : sconnect-ff-v2.15.1.0 .xpi

Downloaded and unpacked the extension:

The file tree is given below :

|  |
| --- |
| inflating: actionpage.html  inflating: manifest.json  inflating: optionspage.html  inflating: \_locales/ar/messages.json  inflating: \_locales/de/messages.json  inflating: \_locales/en/messages.json  inflating: \_locales/es/messages.json  inflating: \_locales/fr/messages.json  inflating: \_locales/nb/messages.json  inflating: \_locales/nl/messages.json  inflating: \_locales/no/messages.json  inflating: \_locales/pt/messages.json  inflating: \_locales/pt\_PT/messages.json  inflating: \_locales/sv/messages.json  inflating: css/html\_dialog.css  inflating: img/close.png  inflating: img/help.png  inflating: img/icon-status-active.png  inflating: img/icon-status-blocked.png  inflating: img/icon-status-inactive.png  inflating: img/icon.png  inflating: img/icon128.png  inflating: img/icon19-bw.png  inflating: img/icon19.png  inflating: img/icon38-bw.png  inflating: img/icon38.png  inflating: img/icon48.png  inflating: img/info.png  inflating: img/options-light.png  inflating: img/options.png  inflating: img/remove.png  inflating: img/trash.png  inflating: js/actionpage.js  inflating: js/contentscript.js  inflating: js/eventpage-ff.js  inflating: js/locale.js  inflating: js/optionspage.js  inflating: META-INF/cose.manifest  inflating: META-INF/cose.sig  inflating: META-INF/manifest.mf  inflating: META-INF/mozilla.sf  inflating: META-INF/mozilla.rsa |

Unzip done

Starting to analyze using - jq [file listing]

Jq . manifest.json

|  |
| --- |
| {  "applications": {  "gecko": {  "id": "jid1-HfFCNbAsKx6Aow@jetpack",  "strict\_min\_version": "42.0"  }  },  "background": {  "scripts": [  "js/locale.js",  "js/eventpage-ff.js"  ]  },  "browser\_action": {  "default\_icon": {  "19": "img/icon19-bw.png",  "38": "img/icon38-bw.png"  },  "default\_title": "\_\_MSG\_inactive\_\_",  "default\_popup": "actionpage.html"  },  "content\_scripts": [  {  "matches": [  "http://\*/\*",  "https://\*/\*",  "file://\*/\*"  ],  "js": [  "js/contentscript.js"  ],  "run\_at": "document\_end",  "all\_frames": true  }  ],  "default\_locale": "en",  "description": "SEcure Addons Manager for Firefox",  "icons": {  "48": "img/icon48.png",  "128": "img/icon128.png"  },  "manifest\_version": 2,  "name": "SConnect",  "short\_name": "SConnect",  "options\_ui": {  "page": "optionspage.html"  },  "permissions": [  "tabs",  "nativeMessaging",  "<all\_urls>"  ],  "version": "2.15.1.0"  } |

rg "permissions|web\_accessible\_resources|externally\_connectable" -n manifest.json

|  |
| --- |
| 48: "permissions": [ |

---Inspect manifest for dangerous or overbroad permissions: <all\_urls>, cookies, nativeMessaging, webRequest, downloads, activeTab.

└─$ jq .permissions manifest.json

[

"tabs",

"nativeMessaging",

"<all\_urls>"

]

┌──(zman㉿kali)-[~/RedFolder/tools/ExtensionTest]

└─$ jq .host\_permissions manifest.json

null

┌──(zman㉿kali)-[~/RedFolder/tools/ExtensionTest]

└─$ jq .web\_accessible\_resources manifest.json

null

### search for risky APIs

rg "eval\(|new Function|document\.write|innerHTML|setTimeout\(|setInterval\(" -n src || true

search for secrets









Search for js files using the following command

find . -type f -name '\*.js' -print

Got t he following Result

The following file sare present under the extension folder

./eventpage-ff.js

./optionspage.js

./locale.js

./contentscript.js

./actionpage.js

Within this look for any the following

· Calls that collect DOM or cookies, then fetch to external hosts → suspicious.

· Calls to chrome.runtime.sendNativeMessage / connectNative → native host communication.

Command to use :

# Search for network / DOM / native messaging APIs in content scripts and background scripts

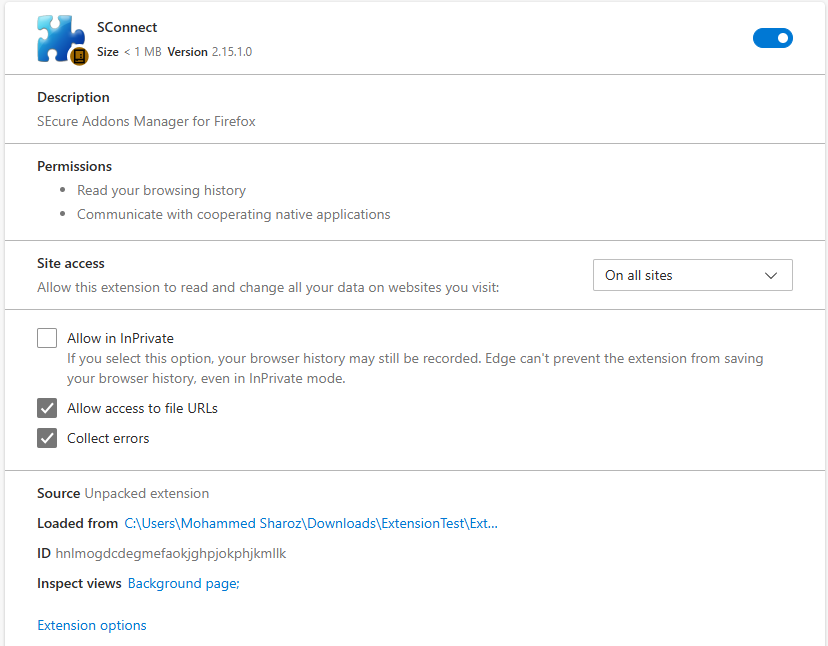
grep -nE "fetch\(|XMLHttpRequest|WebSocket|sendBeacon|document.cookie|localStorage|FileReader|chrome.runtime.sendMessage|chrome.runtime.connectNative|chrome.runtime.sendNativeMessage|browser.runtime.sendNativeMessage|browser.runtime.connectNative" -R .

Result is we found a nativeConnect Call . Screenshot has been attached below



#let confirm whether file access is actually allowed in the browser

Load the file in edge as extension and check whether the allow access to file URLs is enabled . Screenhshot shows that the feature is enabled



Lets Create benign test pages and confirm content script injection

The allow access to files URLs confirms the following

file://\*/\* **in matches + “Allow access to file URLs” enabled**

This permits the extension to run on **local files** (e.g., file:///home/user/notes.html).

If the user opens an attacker-supplied local file (or local file content is otherwise exposed), the content script can read local-file data.

Many people store credentials, tokens, or exported data in local files — making file access dangerous.

In our case we were able to read the following data from the local files in the extension screenshot for the same is attached below

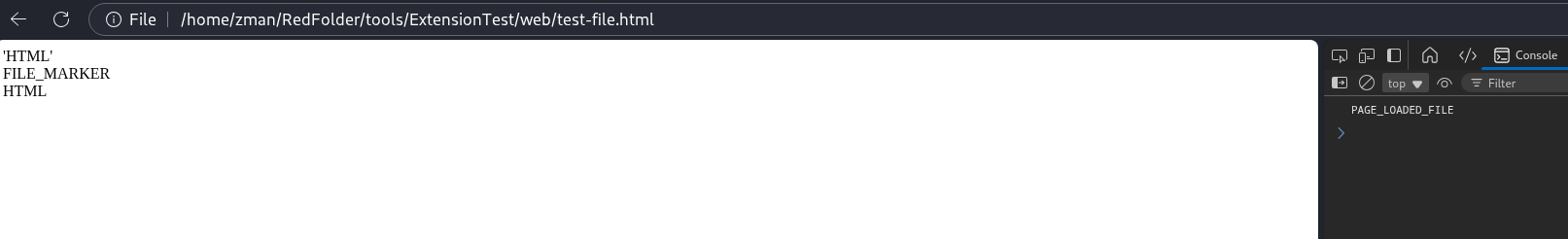
Steps

Access the unpacked location of the extension

Cretate a web folder

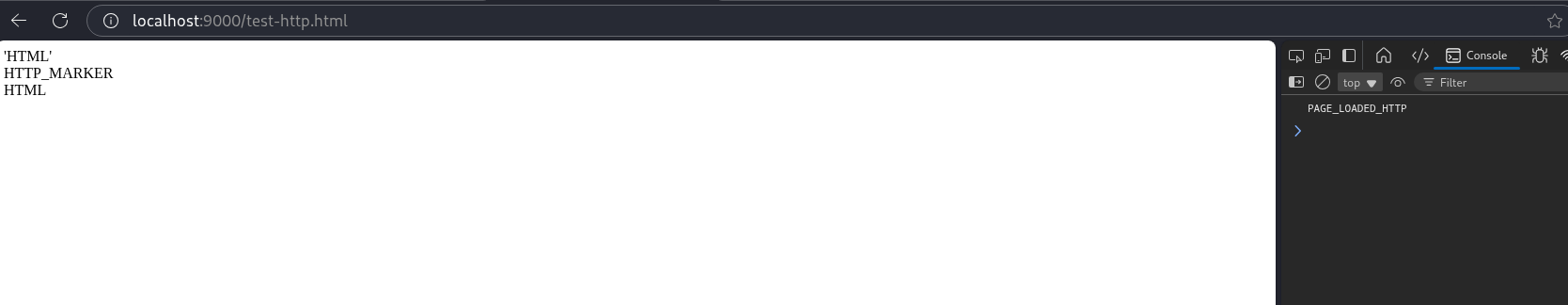
Create a html file to confirm loading of scripts

|  |
| --- |
| 'HTML'  <!doctype html><html><head><title>HTTP TEST</title></head>  <body><div id="marker">HTTP\_MARKER</div><script>console.log('PAGE\_LOADED\_HTTP');</script></body></html>  HTML |



Create html file to confirm whether the local file can be read from the internal extension

|  |
| --- |
| 'HTML'  <!doctype html><html><head><title>FILE TEST</title></head>  <body><div id="marker">FILE\_MARKER</div><script>console.log('PAGE\_LOADED\_FILE');</script></body></html>  HTML |



The <AllURLS>

## What <all\_urls> Enables

The extension can:

Make fetch() or XMLHttpRequest() calls to any URL,

Use chrome.webRequest or chrome.tabs APIs on all sites,

Inject content scripts on any webpage (if matches are also broad).

What we will demonstarte

An extension with <all\_urls> permission can **read** and **send requests** to any domain (since <all\_urls> matches all schemes and hosts).

Even if a page is not related to the extension’s normal function, it can still access that page’s content.

This represents a **wide attack surface**, because malicious code or compromised extension could harvest or send data anywhere.

Steps :

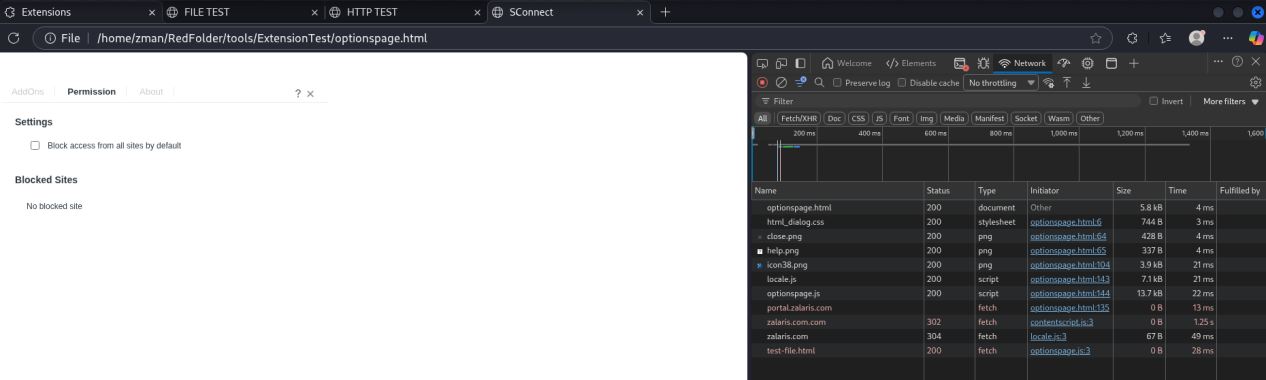
Unpack the extension - access the folder where ‘Java Scripts’ are loaded

Edit the file to fetch any URL or domain , the html code for the same is given below

|  |
| --- |
| // [SAFE TEST CODE] Demonstration of <all\_urls> permission  try {  fetch("https://zalaris.com")  .then(r => r.text())  .then(t => console.log("[DEMO] Successfully fetched example.com length:", t.length))  .catch(e => console.error("[DEMO] Fetch failed:", e));  } catch (e) {  console.error("[DEMO] Exception:", e);  } |

Reload the extension

It can be seen in the console that the js file from extension tried to fetch the following domain - zalaris.com and has received 304 response. Screenshot for the same is given below



Now we have the conten scripts with ://\*/\* enabled and also has the allURLs set plus together with the following

Remediation

· Turn **off** “Allow access to file URLs” unless strictly required.

· Vendor should scope content\_scripts to exact SWIFT/bank domains — not \*://\*/\*.

· Ensure native host allowed\_origins lists only the single extension origin.

· Make native binary & manifest owned by root and not writable by others (chown root:root; chmod 755 for binary; chmod 644 for manifest).

· Use a dedicated VM for SWIFT 3SKey operations.