

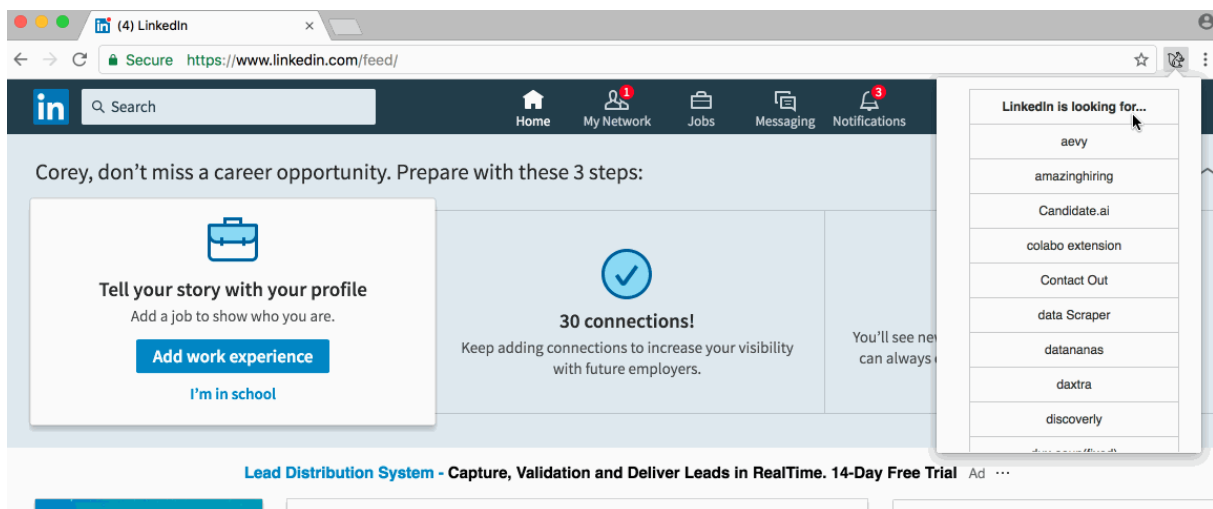
---

## Nefarious LinkedIn

LinkedIn violates their own users' privacy in an effort to detect the usage of browser extensions. At the time of writing this, LinkedIn is scanning visitors for 38 different browser extensions.

I will dive into how LinkedIn detects extensions and what extension developers can do to prevent detection.

I have built an open source browser extension that lists which extensions LinkedIn is currently scanning your browser for.



## Installation

You can click [here](#) to install the extension via the Google web store, or you can download any release from this repository and load it via Chrome's `chrome://extensions` page.

If you are unsure how to load an unpacked extension, you can see [how here](#).

## How does LinkedIn detect extensions?

There are two common ways to detect the usage of a browser extension. The first method is by scanning public resources available in the extension, the other is by analyzing the target web page for abnormal behavior.

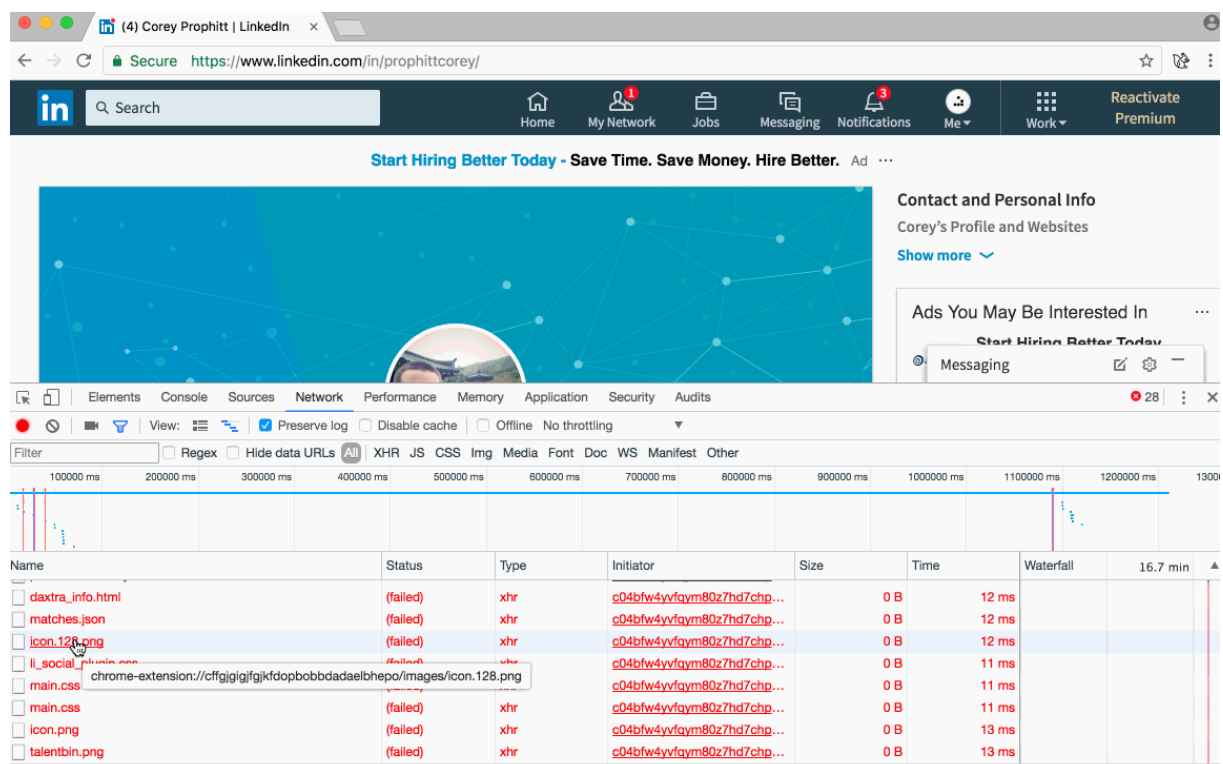
LinkedIn is currently using **both** of these methods.

Public Resources

Public resource detection is by far the simplest method to detect the usage of an extension. It is also the easiest to detect as an end user. In fact, it is what turned me on to LinkedIn's nefarious activities to begin with.

If you browser LinkedIn with your developer console open you may occasionally notice a series of network request errors. If you look at the requests you will notice they are not external requests, rather they are requests to local files. These local files begin with `chrome-extension://` which indicates the web page is making requests to files located in your browser itself.

I have attached a little animated gif showing what these requests look like on LinkedIn.



In this case, the presence of an error message indicates the resource was not available and therefore the extension they were looking for is not installed on your browser. If the request succeeded LinkedIn would then know you do have the extension installed.

## Behavioral Patterns

The second method to detect the usage of an extension is by examining the web page itself and identifying any side effects of an extension. LinkedIn is actively using CSS selectors to locate identifying elements on the page.

For instance, they may look for an ID or class used by the extension, etc.

LinkedIn uses your browser's local storage to store a JSON file that contains a list of extensions to

---

detect. Each extension contains one or more public resources to look for as well as one or more CSS selectors to use to try and detect the extension.

LinkedIn tries to obfuscate the JSON document but doesn't do a very good job.

The file itself is located in the local storage of your browser under the key `C_C_M`. If you look at the contents of the key, you'll see gibberish. The contents are base64 encoded.

Once decoded the contents are further obfuscated. It's clearly a JSON document but the characters are all converted to their unicode byte patterns which makes reading it as a human difficult. If you parse the JSON you can retrieve a human friendly object to examine.

Try it yourself, visit a LinkedIn page, open the inspector and run:

```
1 JSON.parse(window.atob(localStorage.getItem('C_C_M')))
```

You can see the output here:

```
> JSON.parse(window.atob(localStorage.getItem('C_C_M')))  
< ▼ Object {Config: Object, Metadata: Object, date: 1500160643045, version: "0.1.92"} ⓘ  
  ► Config: Object  
  ▼ Metadata: Object  
    ▼ ext: Array(35)  
      ▼ 0: Object  
        date: 1500159386169  
        dom: Object  
          ▼ selector: Array(1)  
            0: "#daxtra-info-div"  
            length: 1  
          ► __proto__: Array(0)  
          ► __proto__: Object  
          interval: 1800000  
          name: "w0mys0"  
          ▼ path: Array(1)  
            0: "ombdgbngokkngdbcahjbeimfcfimdoles/magnet/ChromePlugin/inject/daxtra_info.html"  
            length: 1  
          ► __proto__: Array(0)  
          ► topPath: Array(12)
```

Notice the `selector` attribute in the JSON? It's clear they are searching for an element with the id `#daxtra-info-div`.

If you examine the path list in the JSON you can see which public resources they are looking for. In this case: `ombdgbngokkngdbcahjbeimfcfimdoles/magnet/ChromePlugin/inject/daxtra_info.html`

Clearly, they are trying to identify users with the Daxtra extension installed.

Funnily enough they try to obfuscate the name of each extension. If you look at the name in the screenshot you can see, `w0mys0` which maps to `Daxtra`. It is fairly clear the encoded name has

---

gone through a substitution cipher.

There is enough information in the file to reverse engineer the substitution cipher (which is what I have done in the extension above).

The other information in the JSON is not so interesting. There's mainly book keeping information such as how often to scan your browser, the current version of the JSON document, which URLs to scan for each extension, etc.

I have kept an eye on the document over the last few weeks and I have noticed within the last two weeks the number of extensions LinkedIn is looking for has gone up from 28 to 38.

### **Tips for Extension Developers**

1. Don't use web accessible resources.

Out of the 38 extensions LinkedIn is currently looking for, 28 are using web accessible resources. This is by far the easiest thing to avoid which greatly helps your users' privacy.

Furthermore, there's no good reason to use web accessible resources in an extension! You can always find a solution to your problem that does not require them.

2. Limit content script activity to simple reads.

This is harder to do because it reduces the usefulness of many extensions, but if you want to isolate your extension from the web page you must do this.

3. Don't display your extension using a content script, try a browser action.

Consider using a browser action instead of modifying the page via content scripts. Browser actions run in an isolated environment and makes detecting them much harder if not impossible.

### **Credit**

The squirrel icon used by the extension was found on Flaticon and is credited to Freepik.