Link Scanner Documentation

**Requirements**

**Functional:**

Must be an extension for Google Chrome

Can be an extension for Mozilla Firefox

Must be able to scan web pages for links

Must be able to interface with Virus Total

Must be able to send links to Virus Total and return the results

Must be able to interpret results received from Virus Total

Must notify the user of all malicious or suspicious links

Must be able to be turned on and off by user

Must contain a manifest JSON file and attached html and JS scripts

Must contain a UI with options and a way to deactivate the extension

Must have options that allow the user to enable and disable notifications

Must have options to change how the links are altered after being scanned

Must include the ability for the extension to remove malicious links from the page

Must include the ability to color code safe and malicious links

Can include the ability to place safe and unsafe logo indicators next to links

Must be able to send popups to the user to warn of detected malicious links

Can disable any links it determines are suspicious

**Non-Functional:**

Should be reliable, having no less than 95% accuracy on links scanned

Should be fast, able to scan a link in no longer than 3 seconds

Should Not use more than 5% of the user’s computer’s resources

Should be clear to the user which links are safe and which are unsafe

Should be simple, not requiring more than three clicks to navigate to options or the power button

**Timeline**

December 6th have a basic extension able to communicate with Virus total.

February 6th the extension should be able to interpret the data generated by Virus Total and react accordingly

March 6th Add a script able to read links from any webpage visited and send them to virus total

March 20th add a UI element that has both the options page and a shutdown feature

April 3rd finish implementing the options page and give it functionality

April 10th Polish and Refine

**Design**

The basic design of a chrome extension is a base JSON file that contains information about the program files, permissions, and functionality. The script files are written in JavaScript and html, to start, I created a content script, which is injected into every webpage visited, as well as a popup which is an HTML page than can be viewed when the extension icon is clicked.

**Progress**

Week of 10/11/2020:

The base Manifest.json file was created, it contains basic information such as the extension name, the version of the program, a description, browser action (the extension logo and what pops up when it is clicked), and finally the scripts that execute on webpages visited.

The popup.html file was added and contains information for what the UI will look like. Currently it only contains a placeholder header field.

The content.js file is the JavaScript that is injected into webpages. I added a basic script that uses “.getElementByTagName” to get all link tags and store them as an HTML collection. I then console.log all the links to display them in the console.

Week of 10/18/2020:

The Background File was created, it contains code that should be completed in the background independent of the user or any webpages in the browser. It was initially created as a way to send links back and forth from the extension to the VirusTotal servers. This is accomplished using JavaScript along with an HTML form and the post method.

Week of 10/25/20

The Post method in the Background files throws exceptions whenever there is an attempt to throw a URL, this week was spend researching and attempting to repair the error so that the project can continue. The process used to test sending URLs is looking at the console of the background script using chromes extension manager, while sending a simple URL such as to google.com. I will add the functionality of sending URLs from the visited webpage once commination is set up.

Week of 11/1/2020

The Post method still did not work after a week so I have decided to pivot and attempt a different method. During my research I have discovered the Put method which acts differently than the Post method. In my Understanding, the Post Method attempts to change variables in PHP that have been set up by the developer prior, while PUT simple places desired strings into the address bar. In this case it is impossible to use POST without first knowing the required Variables to change, so I began changing the code to use PUT.

Week of 11/8/2020

The PUT method has successfully placed the strings I give it into the address bar; however, Virus Total does not recognize the URLs I give it. This week I will look deeper into what URLs VirusTotal needs as well as the basic ability for my scripts to communicate with eachother.

Week of 11/15/2020

I implemented a function to convert URLs to their ascii form for special character. I made sure to include the: and / symbols for URLs in addition to other special characters to include other unforeseen cases. The next step is to begin collecting data from virus total to be analyzed and changed.

Week of 11/22/2020

The previous function was removed and replaced with the encodeURIComponent function. I also began working on analyzing the Virus Total results which lead me to find that Virus Total has implemented an API that allows for quick and easy communication and analysis. I intent to change the current code in order to utilize this API instead of the raw PUT method.

Week of 11/29/2020

The structure of the project has shifted from an extension that uses a background script to communicate with Virus Total, to an extension that sends URLs to a server which sends and receives data from Virus Total and sends back the response. I have added a Node.js server which for now runs on the local host and waits for a connection to port 8080, upon which it sends an arbitrary link to Virus Total. The next step is to create communication between the Node.js server and the extension, as well as the ability for the extension to send specific links rather than only needing to connect.