



---

# AUTO OSINT

---

Final Report



LIAM MOORE

C00196503

## Contents

1. Introduction.....	2
2. Project Description .....	3
3. Issues.....	4
3.1 Python .....	4
3.2 API's.....	4
3.3 Cross Origin Resource Sharing (CORS) .....	5
3.4 Export/Copy Functionality.....	5
3.5 Resubmission .....	6
4. Achievements.....	7
4.1 Auto OSINT Tool.....	7
4.2 Communication with OSINT Sites.....	7
4.3 Data Retrieval.....	7
4.4 Malicious Determination.....	8
4.5 Results Layout.....	9
4.6 Export/Copy Function .....	10
5. Future Development Options .....	10
5.1 Additional API Usage.....	10
5.2 Additional Browser Support .....	10
6. Learning Outcomes.....	11
6.1 Personal.....	11
6.2 Technical.....	11
7. What I would do Differently.....	12
8. Testing.....	12
9. Relevance to Cyber Security Industry.....	13
User Manual .....	13
Bibliography.....	14
Acknowledgements .....	14
Declaration.....	14

# 1. Introduction

As cyber-attacks are a growing threat to any business or corporation in today's ever evolving world, it is important that the people who help protect critical infrastructures and businesses can be provided with all the necessary tools to carry out their every-day work tasks. This can be seen by the vast amount of OSINT (Open-source intelligence) websites that have been created to track malicious actors who provide databases of malicious IP addresses, file hashes and URL's. We trust cyber security professionals to be able to gather this information in a timely manner whilst also ensuring that information is up to date and trustworthy. From first hand experience, the process of investigating potential malicious activity can seem monotonous and a time sink. This is mainly due to the several instances of copy/pasting from a security tool or platform to an OSINT site and then repeating this action to gather the relevant information for note taking.

To address this, I have created a Google Chrome Extension, that provides users with the option to submit an IP address to verify if and how it is malicious. It does this by retrieving the most important information that gives users a better understanding of the IP address and the activity it is associated with. The extension provides users with an instant response on a submitted IP address which is populated within the extension itself while also offering users the option to export the information gathered to their favoured text editor or note taking application.

## 2. Project Description

The Auto OSINT Google Chrome extension is a tool designed with the cyber security professional in mind. It enables users to obtain a centralised location within their Google Chrome browser where they can reliably obtain information regarding a certain IP address that may have been included in an alert that was triggered by potentially malicious activity. It allows users to submit the IP address and instantly receive information back from an OSINT site detailing various aspects such as location, owner, domain name, confidence of abuse, number of reports and if it is deemed malicious or not along with a link to the original web page. It provides all of this information in an “IP Reputation Report” that is displayed underneath the search area in a clean easy to read manner. The extension is easy to use and install in your own instance of Google Chrome. All you have to do is load the folder containing all the necessary files to your extensions page, pin the extension to your browser and open it like any other extension. There is no additional steps or setup needed but do note that this is designed specifically with Google Chrome in mind, so you might run into issues trying other browsers.

The copy function that is implemented within the extension provides the user with the ability to export the “IP Reputation Report” by copying the results to the clipboard without restricting the user to a .txt file download or a .csv file download which is a better fit for the goal of the extension.

## 3. Issues

While creating this extension, many issues arose that affected the development stage. Some issues could be fixed easily, some required more effort through investigation and research. These issues ranged from general issues such as learning new coding languages to technical issues like usage of API's.

### 3.1 Python

One of the major issues that affected the development stage of this product was my lack of knowledge of the Python coding language. For me to successfully build and develop this product in Python it involved me learning Python by myself. After attempting to learn Python, I realised that it was going to severely hamper my chances of trying to complete this product as it was too time consuming. To combat this issue, I decided to switch to JavaScript, which although I am not too familiar with, I have had exposure and some experience with it from my studies. This eventually gave me a starting point as I could just start developing right away without any delay.

### 3.2 API's

Since I never had experience using or implementing an API (Application Programming Interface), attempting to utilise the available AbuseIPDB API was a more of a challenge than I expected. At the time I started working on trying to use the API, I was unaware how they work or what is needed in a request to allow the call to be successful. After researching for quite a while, I was able to find some documentation on how the AbuseIPDB API worked and realised that each function of the API had a different endpoint that I was supposed to use. This allowed me to fix the original link I had tried as it was for an older version of their API. As I was only using the 'CHECK' endpoint of the API, I was then able to alter my code to construct the correct URL and append the correct headers, allowing me to make the API call. The following code snippet shows where I constructed the URL using the correct API endpoint, combining it with the IP address supplied by the user and attaching the specified headers which included the API key in an attempt to get a response.

```
//Gather the user supplied information necessary to construct the correct URL
var ip = document.getElementById('ipInput').value;
var apiKey = '6e472e745788187d0fbd5b01502432bc493813d92417fe4f33a1a3026a55708e344ee74b553c98ca';
//define the api endpoint I wish to use
var url = 'https://api.abuseipdb.com/api/v2/check?ipAddress=' + ip;

//Attach the headers to the request
fetch(url, {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
    'Accept': 'application/json',
    'Key': apiKey
  }
})
```

Figure 1 - API Call Code Snippet

### 3.3 Cross Origin Resource Sharing (CORS)

By far my biggest issue, Cross Origin Resource Sharing is a HTTP header-based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources (Mozilla, 2023). This was designed to prevent malicious actors from creating scripts that could steal data or perform various actions on behalf of a user without them knowing. Although this is a widely accepted security feature, it does pose its own problems when it comes to developing web applications that communicate with APIs on a different domain.

Since I was able to enter an IP and click the submit button, I assumed that it would work with no problems but as I observed no response and no information being populated from the API it was clear I had another issue. After some research, I discovered that a lot of developers were unable to generate responses from an API. One of the solutions that was highly recommended was to introduce a proxy server to handle my API calls. This server would sit between my extension and the API server, attaching the appropriate headers and then relaying the information back to the extension to be displayed. This involved me researching how to deploy a proxy server and how to send my requests to the proxy server to be handled and back again. This added to the time pressure I was already under and because of my poor coding capabilities seemed like a real though challenge. I had decided on using Microsoft Azure to create a Virtual Machine that would act as my proxy server to handle my API requests.

It wasn't until I had spent time learning how to set up the proxy server and my trial version of Microsoft Azure had run out that I realised there was an error within my code that meant my request was not reaching the API because it was not even executing. I had implemented some error handling, and it was terminating before being processed. Adding another check to make sure that data was not undefined appeared to have fixed it and the requests were eventually being sent as I could see my API usage limit increasing with each submit.

### 3.4 Export/Copy Functionality

Implementing this function took a while for me to figure out. I wanted to provide an export function so that the user could easily get the results wherever it was needed and not tied to a specific type of file or text editor. The issue I was facing was that when exporting the results it was including the export/copy button itself so that when I used the function and pasted it into another location it was not accurate and went against my extensions main goal. To resolve this, I simply moved the button to be outside of the result area which prevented it from being copied to the clipboard. I also did this with the link that displays at the bottom of the results page and then add it back in when copying.

### 3.5 Resubmission

While testing the functionality I came across another issue. I discovered that when I would resubmit another IP address while looking at the results of the previous one I, the export/copy button and the link to the original site would generate a second time and be added to the results page while keeping the previous link and button. This in turn made the extension appear unprofessional and unorganised. I managed to fix this by implementing code to clear the previous results before as another submit occurs. This image shows the issue that I was having. You can see that the remains of the last IP submission were not resetting like the remainder of the report as they were separated from the results in a container.

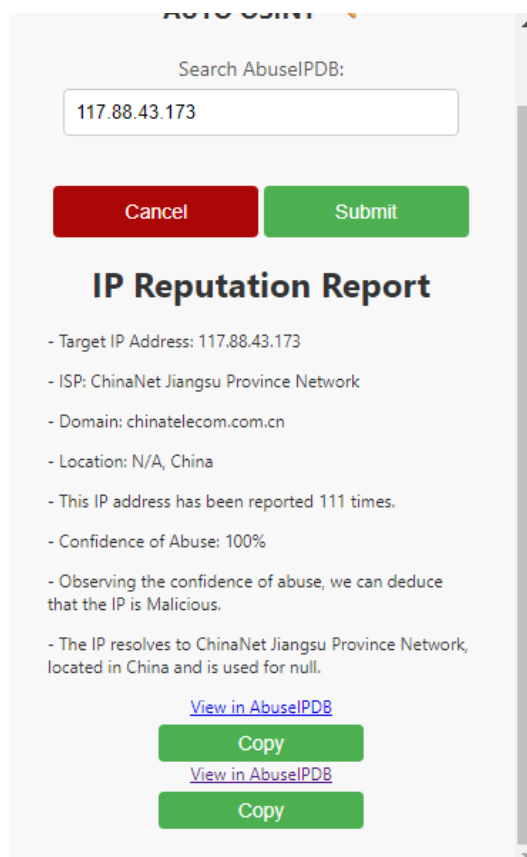


Figure 2 - Resubmission Issue

This code snippet shows the code I introduced to resolve the issue, effectively clearing the container containing the previous link.

```
//clear the previous results
document.getElementById('result').innerHTML = ' ';
//clear the previous link container
var previousLinkContainer = document.getElementById('linkContainer');
{
  if (previousLinkContainer) {
    previousLinkContainer.parentNode.removeChild(previousLinkContainer);
  }
}
```

Figure 3 - Code to Fix Resubmission Issue

## 4. Achievements

In my research document I had listed multiple core deliverables as goals that would help me develop this extension to the way I had imagined it to be and to ensure that it works as intended. I completed most of the core deliverables I had set for myself.

### 4.1 Auto OSINT Tool

I had declared this the most integral part of the development process. I have successfully managed to create and deploy my extension to a browser instance. I have done this while keeping very close to my products original design goal of being easy to read, simple to navigate and although having multiple pieces of information it still appears to be uncluttered. The aesthetics are pleasing to view and don't offer any distractions to users.

### 4.2 Communication with OSINT Sites

I was successfully able to communicate with the API by properly constructing the URL to the correct endpoint to carry out checks on IP addresses. This involved figuring out the correct headers needed to be sent along with the URL.

### 4.3 Data Retrieval

With the URL and the headers successfully being sent to the API, I was successfully able to choose pick and choose which information I deemed most relevant to a cyber security professional. In the end I settled on the most important information being:

- **The target IP Address** - This contains the current IP being searched.
- **ISP** – Contains the Internet Service Provider associated with the IP.
- **Domain** – Contains the domain of the IP.
- **Location** – Shows where the IP address is located.
- **How many times an IP was reported** – Number of times the IP address was reported.
- **Confidence of Abuse Score** – Score given by AbuseIPDB to determine how confident they are that an IP address is malicious.
- **Whether the IP is malicious** – This is my custom logic to return how malicious an IP address is.
- **What the IP is used for** – This contains the information on what activity is typically seen by the IP, (returning null if no results exist).



## 4.4 Malicious Determination

I achieved my goal of coming up with logic based on the information retrieved that will help determine if the target IP address is malicious or not. For this I utilised AbuseIPDB's documentation to discover that I can use the Confidence of Abuse field associated with an IP address as a means to decide whether an IP is malicious or not. Here is the logic I used to create this.

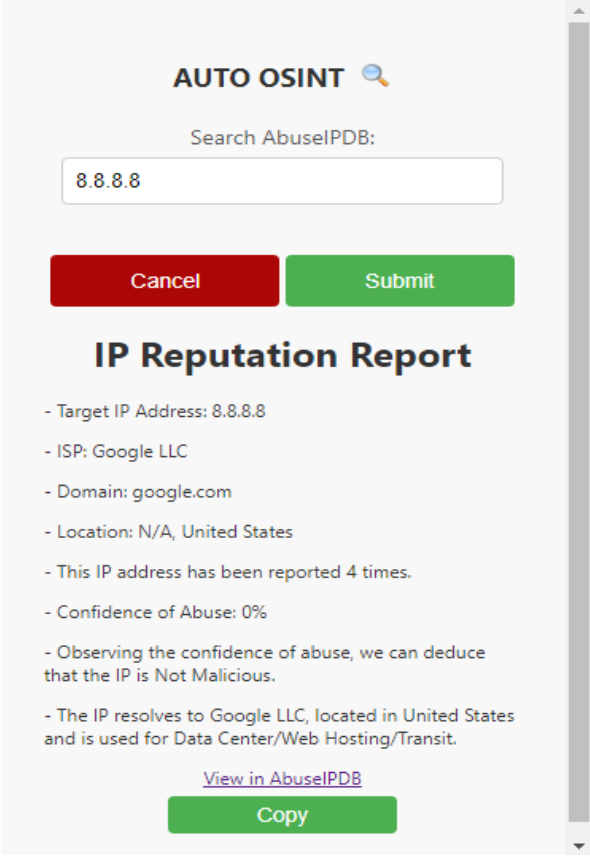
```
//logic to decide what malicious category the IP falls under
var isMalicious = '';
  if (reputationScore === 0) {
    isMalicious = 'Not Malicious'
  }
  else if (reputationScore > 0 && reputationScore <= 50) {
    isMalicious = 'Partially Malicious'
  }
  else {
    isMalicious = 'Malicious'
  }
}
```

Figure 4 - Determine Malicious Rank

According to their website, this is how AbuseIPDB class their Confidence of Abuse. “Our confidence of abuse is a rating (scaled 0-100) of how confident we are, based on user reports, that an IP address is entirely malicious. So, a rating of 100 means we are sure an IP address is malicious, while a rating of 0 means we have no reason to suspect it is malicious” (AbuseIPDB, 2024). Using this information, I was able to create a simple if statement that can assign a rank to an IP based on its Confidence of Abuse score. This can be modified for users who are looking for more accurate reports.

## 4.5 Results Layout

Since the goal of my extension was to provide a clean, easy to read design, I thought it was necessary to adhere to those standards when creating the results display. From this screenshot you can observe that I adjusted the results to suit the size of the display by keeping the, left aligned. I also had to modify the position of the hyperlink and the export/copy button to the centre as it looked out of place being set to the left and directly under each other. I used a container to place them directly in line with each other.



The screenshot shows a web interface for 'AUTO OSINT'. At the top, there is a search bar labeled 'Search AbuseIPDB:' containing the IP address '8.8.8.8'. Below the search bar are two buttons: a red 'Cancel' button and a green 'Submit' button. The main content area is titled 'IP Reputation Report' and contains a list of details for the IP address 8.8.8.8:

- Target IP Address: 8.8.8.8
- ISP: Google LLC
- Domain: google.com
- Location: N/A, United States
- This IP address has been reported 4 times.
- Confidence of Abuse: 0%
- Observing the confidence of abuse, we can deduce that the IP is Not Malicious.
- The IP resolves to Google LLC, located in United States and is used for Data Center/Web Hosting/Transit.

At the bottom of the report, there is a purple hyperlink 'View in AbuseIPDB' and a green 'Copy' button.

Figure 5 - Results Screen

## 4.6 Export/Copy Function

Another of my main priorities was successfully achieved when I implemented a working export/copy function. The whole idea of my product was to make investigations easier and faster for cyber security professionals and I believe that this feature really stands to that. With this functionality, users can copy the results altogether and paste them into their desired text editor or application. Originally, this idea was centered around exporting to a specific file type, either a .txt file or a .csv file. This didn't sit right with me as it restricted the user into a choice of two file types when the main goal is to make the process easier, so I opted for a copy to clipboard functionality. Once users click the 'Copy' button, an alert box will appear letting the user know the copy was successful.

## 5. Future Development Options

### 5.1 Additional API Usage

Even though I completed most of the core deliverables I set for myself, I was not able to completely add a second API to submit file hashes for checks. Due to issues with the first API and time constraints, I was only able to get one of the APIs to work fully. The initial plan was to have two APIs, one for IP addresses and one for file hashes. The file hash checks would have been done using the VirusTotal API which is a very reputable OSINT site within the industry. That being said, I was able to get the IP address checker working pretty well and can't imagine implementing the other API would be much different and something I would have like to have accomplished if I had more time.

### 5.2 Additional Browser Support

Although this wasn't listed as one of my goals, I feel like it would benefit some users to have multi-browser support. That would involve some alteration to the code, but it could be interesting to gain an insight into how other browsers work and could be a fun project to work on and implement.

## 6. Learning Outcomes

Throughout the development of this product, I have learned a lot about what kind of work goes into developing such a product. Some learning outcomes are more personal, while others were technical.

### 6.1 Personal

- While working on this product I have exposed myself to new areas and technologies that I would have been afraid to delve into in the past. I have gained some knowledge about project management and what it takes to have an idea and all the steps involved that turn that idea into a fully functional product.
- I learned that it is important to manage time wisely and that separating the project down into multiple stages can help with time management.
- Changes to projects are almost impossible to avoid. I learned that sometimes when your ideas seem good on paper they don't necessarily equate to the real world.
- When issues arise, coming up with a good alternative or a workaround can help overcome those issues.
- I have learned how important planning and research is in the overall development of the product. It helps identify certain goals and methods.
- I have improved in my ability to problem solve. There is always going to be issues arising, figuring out how to overcome them is important.

### 6.2 Technical

- Throughout the development of my project, I have gain valuable insights how a Google Chrome extension works and what it takes to create one.
- I was able to get more familiar with JavaScript which was a big milestone for me as I am not great at programming or programming languages. I learned a whole host of different functions that I never would have thought existed.
- I had the opportunity to work with APIs for the first time and because of that I now have a better understanding of how they work.
- Using the clipboard API will be of great value to me as I can see it being implemented in other projects for myself going forward.
- I have learned about using containers in my design which I had never had to use before.

## 7. What I would do Differently

If I was to start again, I think I would choose to develop a web application instead of a Google Chrome extension. In my opinion, after creating and using the extension, I feel that I am limited with the amount of screen space I can use. Being cautious to remember that it is an extension and not to take over the user's screen. With more screen space I could have implemented more detailed reports that generate from different sources and allow a user to compare across them. Developing a web application would also allow me to use any browser which would increase the user base potentially. With the extra time gained from not having to learn how to create an extension I could have potentially added more APIs or more functionality such as a user history or user accounts which would involve more security aspects to be considered.

## 8. Testing

Testing this extension was a relatively painful process. Once I had completed the integral parts of my extension it was time to test how it would fair and if I was getting the information I was hoping for. I decided to test for multiple use cases. The first use case I tested was if a user submitted a clean IP address. I observed the results in the extension and compared them to the results within the website. I then tested this on a malicious IP address and noticed that it was working as intended.

After this I tested my logic to determine if an IP was malicious or not. For this I used a wide range of clean and malicious IP addresses. This returned expected results.

Next for testing I tested the link that is provided to the use if they want to visit AbuseIPDB for a more in depth look at their web page. This also functioned as expected with no issues being noted.

I attempted to enter a random string of characters into the input box to observe how it would behave. My error handling successfully handled this and led to an error stating the address cannot be found.

Lastly, I tested the export/copy functionality. Upon the results being displayed to the user, I initiated the function and pasted into a text editor. I noticed that this copied the text in the button itself so that needed to be fixed which I did successfully.

All in all, I am pretty happy with how the testing phase went and I am confident that the extension works as intended.

## 9. Relevance to Cyber Security Industry

As my product is aimed mostly at cyber security professionals already within the industry, I believe it will positively impact their investigations and help with decreasing overall time spent on investigations. From my experience, carrying out checks on IP addresses is an important part of every investigation and some analysts could carry out a high magnitude of checks a day and which takes up a lot of time that could be spent on other important areas with the help of my extension.

Although it is aimed primarily at professionals, it can be used as a learning tool for cyber security students to get a hands-on experience as to what an investigation process can consist of. It can also teach them about what to be on the look out for when it comes to malicious IP addresses and what information would be valuable.

### User Manual

- 1) Please navigate to [Github](#).
- 2) Download all relating files and place them in a newly created folder together.
- 3) Open Google Chrome.
- 4) Click the three dots on the top left of the screen and find 'Extensions'.
- 5) Click 'Manage Extensions'.
- 6) In the top right of the screen, enable Developer Mode.
- 7) Navigate to the top left and click load unpacked.
- 8) Select the newly created folder.
- 9) The AutoOSINT extension should now be in your extensions and ready to use.

## Bibliography

Mozilla , 2023. *Cross-Origin Resource Sharing (CORS)*. [Online] Available at: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS> [Accessed 10 April 2023].

[www.abuseipdb.com](http://www.abuseipdb.com). (n.d.). *Frequently Asked Questions - AbuseIPDB*. [online] Available at: <https://www.abuseipdb.com/faq.html> [Accessed 19 Apr. 2024].

## Acknowledgements

I would like to thank my supervisor, Richard Butler for providing guidance and feedback for the duration of this project. I would also like to thank Security Risk Advisors as they inspired me to create this product.

## Declaration

1. I declare that all material in this submission e.g., thesis/essay/project/assignment is entirely my own work except where fully acknowledged.
2. I have cited the sources of all quotations, paraphrases, summaries of information, Tables, diagrams, or other material; including software and other electronic media in which intellectual property rights may reside.
3. I have provided a complete bibliography of all works and sources used in the preparation of this submission.
4. I understand that failure to comply with SETU's regulations governing Plagiarism constitutes a serious offence.

Student Name (Printed): Liam Moore

Student Number(s): C00196503

Date: 19<sup>th</sup> April 2024