

Technical Report

OSINT

Nathan Ainsley 18028669  
Joe Davies 170   
Professional Development | Date

Table of Contents

[Implementation 2](#_Toc35222819)

[Keys 2](#_Toc35222820)

[Company House 2](#_Toc35222821)

[Who Is API 3](#_Toc35222822)

[Have I been Pwned 6](#_Toc35222823)

[Shodan 7](#_Toc35222824)

[After points 9](#_Toc35222825)

[User Guide 9](#_Toc35222826)

[Company House 9](#_Toc35222827)

[Searcher Class 9](#_Toc35222828)

[Who Is API 13](#_Toc35222829)

[Util Class 13](#_Toc35222830)

[Who Is Class 13](#_Toc35222831)

[DNS Class 14](#_Toc35222832)

[Have I Been Pwned 15](#_Toc35222833)

[Searcher Class 15](#_Toc35222834)

[Shodan 16](#_Toc35222835)

[ShodanSearcher Class 16](#_Toc35222836)

[Project Handover 17](#_Toc35222837)

[Table of Figures within report 20](#_Toc35222838)

# 

# Implementation

## Keys

Nathan quickly identified that multiple different APIs would be needed to be used for this project, he also identified that the keys that would be used by the group wouldn’t be viable for the end user as they would have corporate keys that would allow for more searches and faster results, so it was decided to create a config program that would allow the user to input their API keys and they would be saved into a json file that each module would pull from respectively, this allows the end user to set their keys once and then never have to worry about inputting them ever again without having them hard coded into the code.

## Company House

As a group the first objective was to create the companies house interface, we went through multiple different iterations of this interface but ended up with one that would connect with an external api to link up with the service. It worked by passing some code which we downloaded and loaded into ours some variables and having them return the data, Nathan then created some methods that would sort through the data and format it into the collections that would be useful for the end user. The first aspect of this module that Nathan created was the function to gather the companys number from their name. this can be seen in figure 1.



Figure 1

From there Nathan was able to create the rest of the methods, this is because the api can only retrieve data from the service if it has a company number not the name, so by having the name we where able to query the rest of the service for its data as can be seen in figure 2 where Nathan is retrieving the address of the company by using the company number.  


Figure 2

## Who Is API

Once the Company House module was completed Nathan moved onto the most difficult aspect of the program, the Who Is API. This was the most difficult aspect as the module went through the most different iterations out of all the modules. The initial development of the module consisted of using a script that would interface with the command line within windows however this was incredibly bulky in terms of the file size of the system but was also incredible slow to run. Nathan then did some research and discovered an alternative to this via the WhoIsXMLAPI. Using a http request to this service Nathan was able to retrieve all the data he would need to supply the end user with, however it was a lot of data, so instead of filtering the data down it was decided as a group that all the data would be returned as the ratio of useful data to useless was around 9/1 thus to save on time and complexity of the module that decision was made. As can be seen in Figure 3 the module itself is simple being simply running the request then returning the json that is returned, this pails to the following module



Figure 3

As seen in Figure 4 the Reverse Who is search is much more complex, this is because it is being fed a quadruped of terms to discover a domain compared to being passed a domain and retrieving details. Not only is this function larger but so is the information that is returned. Because of the amount of data that can possibly be returned Nathan decided the best choice for handling this data was to output it to a json file as like before in Figure 3 the data returned had a useful ratio of around 9/1 it wasn’t practical to scrub the data, and because this function could return anywhere between 1 and 1X the amount of results. The amount of data could quite possibly equate to more data than the entire program and was seen to happen multiple times whilst testing. This function can take multiple terms, to a max of 4, consisting of 2 include terms and 2 exclude terms.



Figure 4

Finally for this module whilst working with the API for Who Is Nathan discovered that the company also provided a DNS checkup service, so instead of having to work with multiple different keys and apis Nathan decided to simply use this service once again for this function as well which can be seen in Figure 5 

Figure 5

## Have I been Pwned

Whilst Nathan was working on the past 2 methods Joe had been researching and then working on the HIBP (Have I been pwned) method, this method was later reworked to be object orientated by Nathan. Despite that the module consists of 2 main functions, Hunter and HIBP.  
Hunter is used for the hunter function which worked by passing it a domain name and then it would return a list of email addresses linked to that domain. This can be seen in Figure 6.



Figure 6

The 2nd function HIBP worked by taking that list of emails that was generated within the Hunter function and then passing them to the HaveIBeenPwnedAPI, this api would then return a json consisting of the email that was passed and if it was breached or not, if it was breached then it would list all of the sites that where breached that the email was used on. This can be seen within figure 7. Because the HIBPAPI cant be queried very quickly Joe had to make the function run slower on purpose leaving a 2 second pause between each search, this means that the runtime of this function will be much slower than the rest of the program otherwise the API would not work and would return an error stating you where querying it too fast.



Figure 7

## Shodan

The Final API and class that needed to be created was done by Nathan, this module consisted of one function as can be seen within Figure 8. This function queried the Shodan Api and would return information that would be parsed, the useful information would be collected and then be loaded into an array which would be passed back to the user. This module was a simple one which Nathan remarked as “much simpler after doing the Who Is API as it used many principles that I learnt for the Who Is module”.



Figure 8

## After points

Once the Shodan module was completed the entire program was then completed and so Nathan and Joe had a call with their live client who expressed their pleasure in the product created however they had some issues with it which mainly consisted of hard coded mentions of their company within the code which Nathan was able to remove and fix on the spot.

# User Guide

Because of the nature of the program, from the specification having no interface or back end report generation Nathan decided to make a simple text-based interface for the showcase and for easy testing of the program. This interface can be seen within figure 9. This user guide however wont focus on this interface as this interface is simply for testing or for demonstrating the program at the showcase.

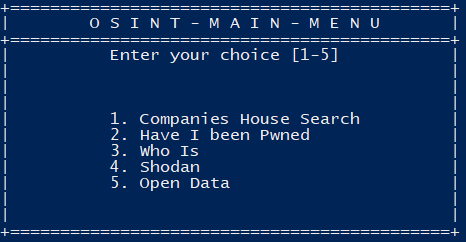


Figure 9

## Company House

### Searcher Class

#### getCompanyNumberFromName(Name)

To start off you will need to get the company number from the Module, to do this you will need to pass it the name of the company as seen in Figure 10.



Figure 10

#### getCompanyAddressFromNumber(Number)

Once you have the Company Number you can then pass it back to the Module to gain more information out of the system as seen in figure 11.

Figure 11

Running the method in figure 11 will return a string containing the address of the company, if you wish to separate the values into their components then they are separated by single a (,)

The Values returned are:

- Address Line 1

- Address Line 2

- Country

- Locality (aka county)

- Postal Code

#### getCompanyStakeHoldersByNumber(Number)

You can also then run the following command as seen in figure 12 to get details on the Company Stakeholders



Figure 12

This command will return with details on the Stakeholders which includes their name and date of birth. This info is returned in a list, with the name and Date of birth being again separated by a single (,)

Values Returned are:

- Name

- Date of birth in format (mm/yyyy)

#### getFillingHistory(Number)

You can also use the following command seen in figure 13 to gain all the filling that the company has done.



Figure 13

Passing this function, the Company number will return a rather large array on all the files that the company has filled.

Values Returned:

- Date of Filling

- Category of Filling

- Description of File

- Type of File

- The Number of Pages in File

- Barcode of File

These values are separated by a (,) and then stored as one large string within an array.

#### getFilingAmount(Number)

The Final Function included is to return the amount of filling that the company has done. This can bee seen in Figure 14.



Figure 14

This method returns a simple integer value that dictates the number of files that the company has on record.

## Who Is API

### Util Class

#### Domains(v):

This Method has the ability to have a string be passed for example ‘google’ this string will then have the most popular domain extensions added onto the end of the string and then saved into an array, the array could be used to iterate through to check all the most popular extensions with the WhoIsAPI. This function isn’t used within any of the main functions by is one that Nathan identified that would be useful for an end user. This function can be seen in figure 15. The Function can be run using the command in Figure 16



Figure 15



Figure 16

### Who Is Class

#### GetDomainInfoByName(domainName):

This method is run by using the following command as seen in figure 17.



Figure 17

The data that is returned would need to be parsed by the user because of the amount of data returned. It was deemed that most of the data within the original json was useful so it would be easier for the end user to pick out what they wanted and didn’t want.

#### Reverse(SearchTerm1,SearchTerm2,ExcludeTerm1, ExcludeTerm2)

This command which is seen in Figure 18 will return all of its data into a json file, this is the only command like it as the amount of data is way to much to handle in an efficient manor as it returns just as much information as the GetDomainInfoByName(domainName) function however it could return multiple different domains, this means that there would be X times the amount of Data where X is the amount of Domains that where found.



Figure 18

#### GetIP(domainName)

This is another function that is useful, without using a single api this function uses the socket function within python to be able to get an IP for a domain name, this function which is seen below in figure 19 is run by the command within Figure 20.



Figure 19



Figure 20

### DNS Class

#### DNS\_Lookup(domain):

This function can be run using the command listed in Figure 21



Figure 21

This function will return a json containing all the information on the DNS Server linked to the domain passed to it.

## Have I Been Pwned

### Searcher Class

#### Hunter(SearchTerm)

This function is run with the command listed within Figure 22.



Figure 22

This function will return a list of email addresses linked to the domain that was passed to it.

#### HaveIBeenPwned(emails)

This function is run with the command listed within Figure 23.



Figure 23

This function will iterate through the array that is passed to it and will search the haveibeenpwned database to see if there have been any breeches where the email was listed. The results are then returned as a json with the email address or if there was a breech included the individual breeches and if there wasn’t a breech.

## Shodan

### ShodanSearcher Class

#### Shodan(v)

This function is run with the command listed within figure 24.



Figure 24

This function when passed an IP will return a list containing the information on that IP. This list can either be parsed and used by another function or could be printed to terminal by the printer function also contained within this class.

#### Printer(v)

This function is run with the command listed within Figure 25.



Figure 25

This Function will print out the List that was generated by the Shodan function which was seen in Figure 24. This function isn’t useful if you don’t have any intentions of outputting this data to the terminal.

# Project Handover

If another team was to take this project and continue it which to our understanding that is the current intention with the live client that the project was run alongside. The live client indicated towards us that they intend to use the code that we used to be used alongside an interface another team had produced. There are some main key points that the team would need to know about the code which is listed below.

* The Companies House module could be reworked to gain more information and possibly a faster run speed.
  + This module should be reworked to use a http request instead of the method we used. When we started the project nor Nathan or Joe had ever used a HTTP request and so we didn’t think to use them. The solution we retrieved does exactly what a http request does just a bit worse. Nathan believed that using a http request would possibly get more data that we currently did however the current method did everything we needed it to do and so it was decided that no changes would need to be made for the time being.
* The Who Is was reworked as Nathan wasn’t happy with the amount of information being returned by the initial method however this came with a drawback.
  + A new method would need to be made to mount this function, this is because a dedicated method would be needed to parse the json that is returned and filter out the information that isn’t needed. However, because of the ratio being 9/1 as a group it was decided that it was a waste of time and recourses to filter out a couple datapoints in around 50 or so. A new method would certainly be needed to perform this filtration on the output as the amount of data returned to the user is simply too much to handle.
* The Reverse Who Is needs a similar mounting to the regular Who Is just much more extensive.
  + Because of the raw amount of data that it is possible for this method to return Nathan decided to output the data to a json file so it could then be read into the program again or a different idea implemented. It would have taken a significant amount of time to create a function that would have been able to handle the raw amount of data that is spat out by the function, so it was just decided to put it to a file. By having a dedicated method to filter down the data it could be made into a more useful format as currently it is just an overflow of information
* Key functions
  + Something that was mentioned within the implementation section but not within the user guide was the Key() functions, these functions load in the keys from the json file. Seen in Figure 26 is the Key() function from the WhoIs module. There is a function like this within each of the module, this function will most likely be located within the util class, there would be no reason to call this function however any function that requires the key for a search with an api will be calling this function to obtain the key. Under no circumstance should these methods be removed or edited as it will more than likely break the entire module.



Figure 26

Include within the code is a few txt and json files that haven’t been mentioned thus far.

* + These files are:
    - DNSOutput.json
    - ShodanOutput.json
    - OutputOSINT.txt
    - WhoIsOutput.json
  + These files are not used within the code of the main 4 modules except for the showcase program. These files can safely be discarded alongside the Showcase.py program if you have no intention of showing off the methods in a simple way.
    - The way the Showcase.py method works is a simple text interface that will output all its data into text or json files that it will then open to show off the data to the user who is looking at it. This can be seen in both Figure 27 and Figure 9 located on [page 9](#_User_Guide).



Figure 27

# Table of Figures within report

[Figure 1 2](#_Toc35223196)

[Figure 2 3](#_Toc35223197)

[Figure 3 4](#_Toc35223198)

[Figure 4 5](#_Toc35223199)

[Figure 5 6](#_Toc35223200)

[Figure 6 6](#_Toc35223201)

[Figure 7 7](#_Toc35223202)

[Figure 8 8](#_Toc35223203)

[Figure 9 9](#_Toc35223204)

[Figure 10 10](#_Toc35223205)

[Figure 11 10](#_Toc35223206)

[Figure 12 10](#_Toc35223207)

[Figure 13 11](#_Toc35223208)

[Figure 14 12](#_Toc35223209)

[Figure 15 13](#_Toc35223210)

[Figure 16 13](#_Toc35223211)

[Figure 17 14](#_Toc35223212)

[Figure 18 14](#_Toc35223213)

[Figure 19 14](#_Toc35223214)

[Figure 20 14](#_Toc35223215)

[Figure 21 15](#_Toc35223216)

[Figure 22 15](#_Toc35223217)

[Figure 23 15](#_Toc35223218)

[Figure 24 16](#_Toc35223219)

[Figure 25 16](#_Toc35223220)

[Figure 26 18](#_Toc35223221)

[Figure 27 19](#_Toc35223222)