

## Project overview for portfolio upload

**Project Title:** Shodan Scan: Uncovering Vulnerable Devices with Python

### Project Description:

Creating a Python script that implements the Shodan API which allows the user to search for vulnerable devices. Using queries the user can identify devices which might be vulnerable to a specified CVE. This report is centred around the use case of a company approaching a security researcher and requesting they identify technologies running vulnerable versions (as identified by known CVE's)

### Project Tooling:

- Shodan & Shodan API
- Python

### Project processes and techniques:

- Python Scripting
- API Usage
- Shodan

**Grade:** A+

*Appendices have been removed to avoid plagiarism from future students, however I am happy to provide additional information or discuss the work further if desired.*

**Disclaimer:** *I do not give permission for this work to be copied, those who do may be liable for plagiarism.*

---



# Abstract

---

This report aims to outline the development of a Python script which is created as a vulnerability scanner for Shodan. The development of the script adopts the following scenarios as potential use cases: security researcher looking for vulnerable devices for an organisation who have identified relevant CVEs, a network security audit, and a penetration test engagement. Utilising a multi perspective approach, the researcher aims to cover functionality that is beneficial to each use case, whilst outlining angles for future work to improve the comprehensiveness of the finished script. The researcher outlines the design and implementation choices involved in the development process, referring to why specific search filters were used, alongside determining how the output of the searches are handled.

The script implements the highlighted libraries then initialises the Shodan API key allowing the script to search Shodan. The script then takes in a user input and uses this as a query to then search Shodan for all the results that match the given query. From here the script retrieves the returned results and specifies which labels of the results it wishes to use. From here, the script will take the applicable labels and then parse them into a word document. The word document is then formatted with the appropriate information and saved with a title that has the query within its saved name. This process will be done using Shodan's API, alongside a Python library entitled '*docx*' for using word documents and finally a '*datetime*' library to provide additional information on the execution timing of the document.

The script outputs a word document with the specified labels, IP, Port, version, product, device type, org, location. These labels specify the fields retrieved by Shodan and allowed the researcher to find outdated technologies that are vulnerable to specified CVE's. Each document has a series of headings including the title of the script, the author of the script, the entered search query, and the time the search was produced. The document with the output only holds up to 100 results, therefore only accessing the first page of results. This is due to the academic membership having restricted functionality. The document then saves to the director the user is currently on, using the user query as part of its save name. The script was not as specific to CVE's as in the initial brief, however the use cases have changed to reflect this.

# Contents

---

1	Introduction.....	1
1.1	Background .....	1
1.2	Aim .....	2
2	Program and Development .....	3
2.1	Overview of Procedure (200 words) .....	3
2.2	PRE-DEVELOPMENT .....	4
2.3	Understanding Shodan's Functionality .....	5
2.4	Implementation .....	5
2.4.1	Library choices .....	5
2.4.1.1	Using the date and time module.....	5
2.4.2	Initialising API .....	6
2.4.3	User Input .....	7
2.4.3.1	Formatting user inputted query.....	7
2.4.4	Performing Search .....	7
2.4.5	Creating Document.....	8
2.4.5.1	Formatting Document.....	8
2.4.6	Iterating through results.....	8
2.4.6.1	Deciding on search appropriate search fields.....	9
2.4.7	Saving results to document .....	10
2.4.8	Error handling .....	10
2.5	Output.....	10
2.5.1	Headings .....	10
2.5.2	Format of output .....	11
3	Discussion.....	12
3.1	General Discussion .....	12
3.1.1	Chosen CVE's to demonstrate use case and outputs .....	13
3.1.2	Use Cases .....	13
3.1.2.1	Security Research for outdated products .....	13
3.1.2.2	Network Security Audit.....	13
3.1.2.3	Penetration Test Engagement.....	13
3.2	Difficulties .....	14

3.3	Countermeasures.....	14
3.4	Future Work.....	15
	References.....	16
	Appendices.....	<b>Error! Bookmark not defined.</b>
	Appendix A – Resources Used .....	<b>Error! Bookmark not defined.</b>
	Appendix B Output results .....	<b>Error! Bookmark not defined.</b>

# 1 INTRODUCTION

## 1.1 BACKGROUND

In today's digital landscape, the pervasiveness of Internet of Thing's (IoT) related devices is striking. The devices in question are often created by thousands of different organisations across the world, where each operating organisation may have different, varying levels of security they must adhere to when creating these devices. The rise in internet of things also provides a monetary benefit to these organisations who wish to capitalise on the trend of individuals becoming increasingly interconnected. Figure 1-1 shows how the IoT market In the USA is expected to grow substantially in the next 6 years, increasing the revenue of these companies by 42% (Al-Sarawi *et al.*, 2020).

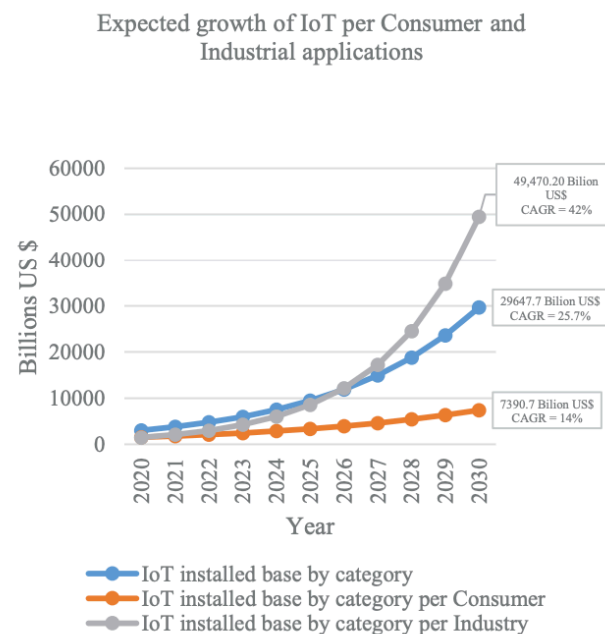


Figure 1-1 Statistics of IoT revenue in the USA  
Source : (Al-Sarawi *et al.*, 2020)

Because of this inability to regulate security rules and regulations, and the monetary gain to be had by the organisations making these devices, this can often lead to the created devices containing security faults, therefore leaving them open to exploitation by malicious actors. When popular devices become vulnerable, it can make mainstream news or be discussed on social media, and this forces organisations to become proactive in releasing patches. Additionally, it can allow for individuals to be aware that their devices are compromised and to update and use these patches. However, for other devices, such as those from lesser-known brands or boutique IoT devices, compromised devices may never be brought to light creating many vulnerable users (Albataineh and Alsmadi, 2019).

Platforms like Shodan can be used by individuals and organisations alike to better assess their own security posture or the security posture of their devices that are out on market. Shodan consistently scans for these IoT devices and as a result can display vulnerable devices when the right query is used on its search engine (Genge and Enăchescu, 2016). When combining this functionality of Shodan with the other tools security researchers have at their disposal such as databases of known vulnerabilities, known as CVE's, it can allow for the researchers to identify an organisations product and assess how vulnerable they are. However, this is often time consuming and as a result would benefit from tooling to provide quicker and more effective results so that individuals and organisations can regularly assess and improve their security.

## 1.2 AIM

---

The researcher aims to produce a script using the Python scripting language that will search the Shodan database through utilising Shodan's API. The researcher will attempt to produce a script that will be able to be beneficial in the following use cases (more information on the use cases can be outlined in section 3.1)

- Security Research for CVEs
- Network Security Audit
- Penetration Test engagement

To meet the requirements of the three specified use cases, it is important for certain criteria to be met. This criterion will provide the main aims of the script and accompanying report

- Utilise Shodan API to perform searches on Shodan
- Allow for the user to input their search queries and understand what they are searching for
- Process the search and retrieve the relevant information for each required field
- Allow for the user of the script to execute it from the terminal
- Output the retrieved search to another document to allow for further use
- Format the document to include important information, enabling the script output to be used In a larger context

Alongside the script, the research will aim to product a report which documents the procedure, acknowledges pitfalls, and proposes future work for the created script. This report will also provide the background knowledge to understand Shodan so the user can get the most out of the script.

# 2 PROGRAM AND DEVELOPMENT

## 2.1 OVERVIEW OF PROCEDURE

```
import shodan
from docx import Document
from datetime import datetime

#API key obtained from Shodan login dashboard
#Intilisaing shodan API object, allowing the use of the Shodan library
api_key = 'A4nUAZNda1GRGU0dXApja5o3DhFZLuel'
api = shodan.Shodan(api_key)

#initialising datetime object to provide date//time on outputted document containing results
execution_time = datetime.now()
date_and_time_str = execution_time.strftime("%d/%m/%Y %H:%M:%S") # dd/mm/YY H:M:S

#encloses the shodan search and handles the API errors if it fails execution
try:
    #function peforming the foundational search of shodan
    # Takes the shodan api object and searches using provided 'query' (user input)
    def search_shodan(query):
        search_results = api.search(query)
        return search_results

    #The following block formats a series of print statements to screen
    # - to tell user how to format input (particularly if more than one query)
    # - prompt the user to input this query
    print()
    print('Searchterms used are those used on shodan, for expmale if we wish to type port & technology it would be ==> port:443 product:nginx')
    print()
    query=input('Enter your searchterms here: ')

    #PERFORMS THE SEARCH! uses user query that was inputted
    results = search_shodan(query)

    #First line creates document called results_from_search
    #The rest of the lines set headings for the word document that results are being saved too
    results_from_search = Document()
    results_from_search.add_heading('ShodanScan: Uncovering Vulnerable Devices with Python', level=1)
    results_from_search.add_heading('Script by Jamie Rice, 2024', level=2)
    results_from_search.add_heading('Your Entered Search Query Was == ' + query, level=4)
    results_from_search.add_heading('Your search was produced at ' + date_and_time_str, level=4) #uses the variable define on line 17
    results_from_search.add_paragraph()
    results_from_search.add_paragraph("-----")

    #initialises for loop to cycle through matched results from shodan search
    #extracts the information(the specified filters) and displays in word document
    for matched_result in results['matches']:

        #Intilias placeholder with label, i.e IP: {} awaits recieved input, format marks what will go in {}
        #matched_result.get('ip_str') uses the key 'ip_str' to find associated value in matched_result dictionary and return shodan search value
        #some reuslts have N/A as they could potentially not have results - where as the other fields typically will when querying shodan
        results_from_search.add_paragraph('IP: {}'.format(matched_result.get('ip_str')))
        results_from_search.add_paragraph('Port: {}'.format(matched_result.get('port')))
        results_from_search.add_paragraph('Version: {}'.format(matched_result.get('version', 'N/A')))
        results_from_search.add_paragraph('Software: {}'.format(matched_result.get('product')))
        results_from_search.add_paragraph('Device: {}'.format(matched_result.get('devicetype')))
        results_from_search.add_paragraph('Organisation: {}'.format(matched_result.get('org', 'N/A')))
        results_from_search.add_paragraph('Location: {}'.format(matched_result.get('location', 'N/A')))
        results_from_search.add_paragraph()
        results_from_search.add_paragraph("-----")
        results_from_search.add_paragraph()

    #saves document and uses the user inputted query as the text file
    results_from_search.save((query) + ' Search results document.docx')

#part of the error handling process, will print errors regarding API execution
except shodan.APIError as e:
    print('Error: %s' % e)
```

Figure 2-1 – The whole script



Figure 2-1 provides an overview of the entire script for reference. The script is broken down into isolated sections that provide the reader with a better understanding of the functionality of the script.

The development of the created script required knowledge of Shodan, API's and some fundamental Python to get up and running, whilst development took place solely in Visual Studio. The output of the script is sent to a Word Document where the results are formatted and saved. When querying Shodan, the search fields that the script is looking for is the following;

- IP
- Port
- Version
- Software
- Device
- Organisation
- Location

The design decisions of why these fields are used is further outlined in section 2.3.6. When the output is parsed to a word document, the results are separated by a line of Octothorps with a series of headings at the top of the page providing more information about the search just performed. More on understanding the results is outlined in section 2.4.

Across process of research, design, development and implementation, the researcher did encounter some roadblocks, therefore they had to make mitigations, and in some instances change the functionality of the project first proposed. For example, when the researcher first decided to develop the project, there was more of a focus on the CVE and being able to search for specific CVE's. However, as the project progressed this proved to provide some roadblocks and as a result is listed in section 3.2 (pitfalls and mitigations) and the work that could have been carried out is documented further in section 3.3 (future work)

## 2.2 PRE-DEVELOPMENT

---

Before jumping into development, the researcher decided to break down the task at hand into modular problems to ensure the task was tackled correctly. This systematic approach is reflected in the implementation section 2.3, however the high-level breakdown is as follows;

- 1) Select libraries (*Shodan, docx, datetime*)
- 2) Utilise Shodan API to introduce Shodan functionality to the script
- 3) Take in user provided query
- 4) Search Shodan using query
- 5) Specify what parameters from the search the researcher wished to capture
- 6) Print search results to a word document, format and save the document
- 7) Handle potential errors should the search fail

## 2.3 UNDERSTANDING SHODAN'S FUNCTIONALITY

---

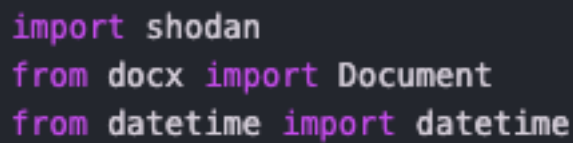
Before developing the script, it was important for the researcher to understand the way in which Shodan functions. Shodan performs continuous scans of the internet and indexes the information it obtains on its database. This is done through banner grabbing, banner grabbing obtains information such as software version, open ports, metadata, IP addresses, location, device type and more. Banner grabbing can sometimes even lead to vulnerable information being leaked (Fernández-Caramés and Fraga-Lamas, 2020) and this is where Shodan can be used by malicious actors. When using the search functionality on Shodan, the user can search for this banner information using filters, the filters are predefined by Shodan and can be seen on official Shodan documentation. These filters are used in the created script as part of the users input for the search query.

## 2.4 IMPLEMENTATION

---

### 2.4.1 Library choices

The libraries implemented in figure 2-2 cover the functionality for each part of the script. Firstly, the Shodan library allows access to the API and therefore access to the functionality of the website, it enabled the researcher to use the search queries just like they were on Shodan within a browser. There are no other libraries which provide this functionality therefore the choice remained quite simple for the researcher. However, when choosing the 'docx' library, the researcher could have selected numerous other ways to format the script output, such as something as 'PrettyTable', a python library that formats text into tables. However, the researcher believed importing the results to a word document, where they are stored in text and in a sequential order would allow for easier manipulation should the script be expanded to take file inputs, as outlined in section 3.4.



```
import shodan
from docx import Document
from datetime import datetime
```

Figure 2-2 Importing Libraries

#### 2.4.1.1 Using the date and time module

The rationale the researcher had behind implementing the datetime module is quite multi-layered and reflects the design choices regarding the word document. The researcher believed it was important to consider the use case of the script. Organisations request the security researcher to investigate for vulnerable devices on Shodan and produce a report. Therefore, its essential that the researcher considers the organisational usage when doing so. For example, when a client such as an organisation asks for work to be performed, it is important for both parties to determine when the search was executed. This can allow the researcher to manage the output files correctly and it also allows the clients to keep track of what vulnerable devices exist that they should be concerned about. Continuing this use case, the researcher understands that the organisations may wish to contact their customers who are running the outdated version to get them to update, keeping track of the progress of customers who update their technology is also useful. Therefore, this simple addition of including a timestamp can

have quite large benefits to the overall functionality of the script, in turn ensuring the security researcher had to make sure this was implemented correctly.

```
#initialising datetime object to provide date//time on outputted document containing results
execution_time = datetime.now()
date_and_time_str = execution_time.strftime("%d/%m/%Y %H:%M:%S") # dd/mm/YY H:M:S
```

Figure 2-3 – date & time library implementation

## 2.4.2 Initialising API

The API key was acquired by the researcher from the dashboard of the Shodan login as seen in figure 2-4. The researcher was only limited to an academic licence. The academic licence provides most of the functionality that a regular commercial licence does, apart from providing the ‘vuln’ filter that is available for commercial developers. This provided a slight roadblock in the researchers proposed design as it would have allowed a simple search to provide a comprehensive outlook into specified vulnerabilities. The work around and future work surrounding the use of the vuln filter is outlined in further sections.

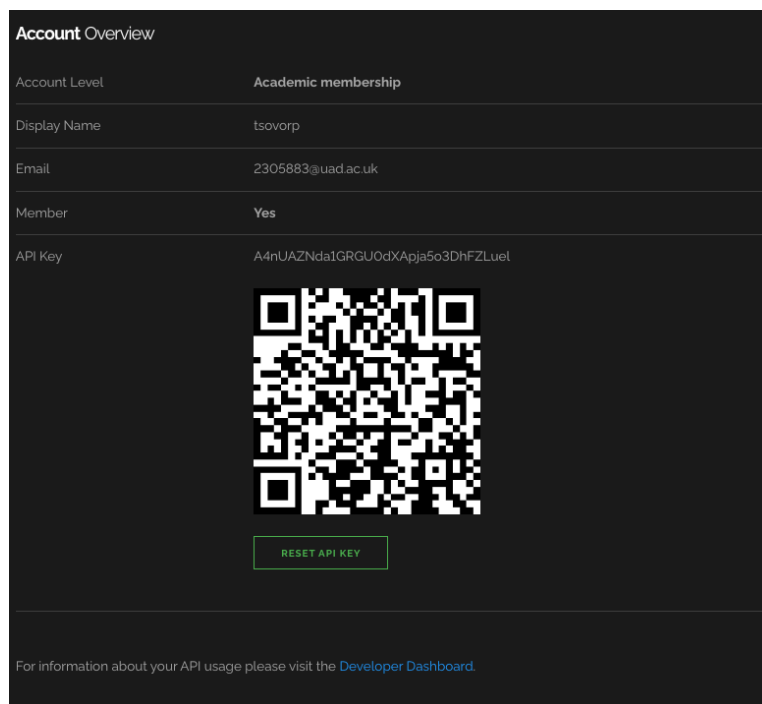


Figure 2-4 – dashboard for API key

The code outlined in figure 2-5 serves as the foundational connection between the Shodan API (therefore Shodan itself) and the script being created. The API key is initialised in the variable ‘*api\_key*’ and is passed the string that is seen in figure 2-5. Next the researcher created a Shodan API object which allows the functionality of the Shodan service to be used in the script.

```
#API key obtained from Shodan login dashboard
#Intilisaing shodan API object, allowing the use of the Shodan library
api_key = 'A4nUAZNda1GRGU0dXApja5o3DhFZLue1'
api = shodan.Shodan(api_key)
```

Figure 2-5 Initialising API Key

### 2.4.3 User Input

One of the most important parts of the functionality of the script was ensuring that the researcher was able to provide it with queries so that each search could be tailored to client requirement. Whilst the researcher is familiar with the way shoran formats its queries, the extra information outlined in the print statement in figure 2-6 would not be needed. However, as this script is built by the researcher with future work in mind, it is vital anyone who picks up the finished script can work with the script in the most basic of functionality.

#### 2.4.3.1 *Formatting user inputted query*

Shodan takes a specific way of querying for the engine to return the exact results. For example, when the researcher wishes to perform a filtered search, use of a colon (':') is required. This will allow the researcher, or any other user to specify the request of that filter. For example, figure 2-6 outlines the structure of one query – 'port:443 product:nginx'

```
#The following block formats a series of print statements to screen
# - to tell user how to format input (particularly if more than one query)
# - prompt the user to input this query
print()
print('Searchterms used are those used on shodan, for expmale if we wish to type port & technology it would be ==> port:443 product:nginx')
print()
query=input('Enter your searchterms here: ')
```

Figure 2-6 Formatting user query

### 2.4.4 Performing Search

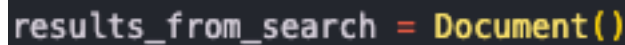
This section of the script performs its main function. The code outlined in figure 2-7 shows how the search query is executed with a 'search\_shodan' function. The search query takes in the user input and sends this to the Shodan API, thus initiating the search on the Shodan site. The researcher identified that Shodan returns its query in a dictionary format. Therefore, the researcher chose the variable name 'results' to allow for this. The data stored inside this variable will be a collection of metadata which matches the search query chosen by the user. It is important to take notice of the variable name 'results' as this is used by the researcher at further stages in the script to obtain the results.

```
#PERFORMS THE SEARCH! uses user query that was inputted
results = search_shodan(query)
```

Figure 2-7 Performing Search

### 2.4.5 Creating Document

The following section outlines the creation of the word document and the subsequent creation of headings which helped the researcher format the document to create an easy-to-understand display of results. Figure 2-8 shows the line of code which initialised a new word document object 'results\_from\_search'. This is the process required by the docx python library to start a new document.



```
results_from_search = Document()
```

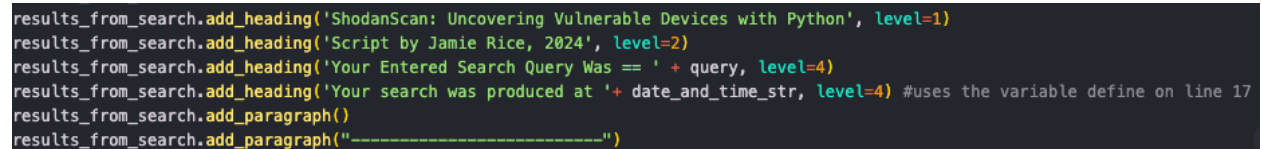
Figure 2-8 Initialising document for search results

#### 2.4.5.1 Formatting Document

After the creation of the document, it was important for the researcher to implement as much initial information in the document as possible without interfering with the results. Therefore, they implemented a hierarchy of headings.

- Heading 1 – Script Title & Description
- Heading 2 – Author name
- Heading 3 – Passing the 'query' variable so whatever search was used is printed to the doc
- Heading 4 – Using the 'date\_and\_time\_str' to indicate to the user when the script was used

After this, there are some lines of formatting inputted by the researcher to provide a more structured introduction to the document.



```
results_from_search.add_heading('ShodanScan: Uncovering Vulnerable Devices with Python', level=1)
results_from_search.add_heading('Script by Jamie Rice, 2024', level=2)
results_from_search.add_heading('Your Entered Search Query Was == ' + query, level=4)
results_from_search.add_heading('Your search was produced at ' + date_and_time_str, level=4) #uses the variable define on line 17
results_from_search.add_paragraph()
results_from_search.add_paragraph("-----")
```

Figure 2-9 formatting headings

### 2.4.6 Iterating through results

The code outlined in figure 2-10 shows the creation of a 'for' loop that will iterate over each matching result from the search performed in section 2.4.4. This functionality was crucial for the researcher to implement as it allows for numerous results to be obtained and not just the first one from the Shodan query. Each result 'matched\_result' that is obtained contained a specific device or technology which matched the user query that was input by the researcher. Therefore, when looking at the loop in relation to the loops body created by the researcher, the functionality of the loop is summarised in the following way:

- Extracts results matching the users query
- Formats the result
- Adds the extracted information to the word document created in the previous section

```
#initialises for loop to cycle through matched results from shodan search
#extracts the information(the specified filters) and displays in word document
for matched_result in results['matches']:
```

Figure 2-10 Initialising For loop to iterate over results

#### 2.4.6.1 Deciding on search appropriate search fields

The researcher achieved a lot with the first seven lines of the code outline lined in figure 2-11 as they attempted to structure and format the output of the search retrieved by Shodan.

```
#Intiliases placeholder with label, i.e IP: {} awaits recieved input, format marks what will go in {}
#matched_result.get('ip_str') uses the key 'ip_str' to find associated value in matched_result dictionary and return shodan search value
#some results have N/A as they could potentially not have results - where as the other fields typically will when querying shodan
results_from_search.add_paragraph('IP: {}'.format(matched_result.get('ip_str')))
results_from_search.add_paragraph('Port: {}'.format(matched_result.get('port')))
results_from_search.add_paragraph('Version: {}'.format(matched_result.get('version', 'N/A')))
results_from_search.add_paragraph('Software: {}'.format(matched_result.get('product')))
results_from_search.add_paragraph('Device: {}'.format(matched_result.get('devicetype')))
results_from_search.add_paragraph('Organisation: {}'.format(matched_result.get('org', 'N/A')))
results_from_search.add_paragraph('Location: {}'.format(matched_result.get('location', 'N/A')))
results_from_search.add_paragraph()
results_from_search.add_paragraph("-----")
results_from_search.add_paragraph()
```

Figure 2-11 implimenting the search fields to the document

Firstly, each line was added as a 'paragraph' to the word document object, which at that point was still named '*results\_from\_search*'. The '*add\_paragraph*' functionality is built into the '*docx*' library, it creates a new line of text in the word document, therefore for each line of code that this is included in, it will automatically start a new line in the document.

Next, a placeholder was used, which is indicated by a string and {}. Each line had its own associated label and those were the fields which were obtained in the Shodan search. The labels include **IP**, **Port**, **Version**, **Software**, **Device**, **Organisation**. It was important for the researcher to use the place holder ('{}') as it allowed to factor for the changing results often obtained for each search retrieved from Shodan. Following this the '*format()*' method will be used to insert the values of the search into the specified place holders, an example of this can be outlined in the first line where '*matched\_result.get('ip\_str')*' retrieved the IP information from the Shodan search, where '*ip\_str*' is the key that is used on the Shodan results dictionary stored in the previous variable '*results*'. The output of this is then inserted in the placeholder '{}'

Some lines include a 'N/A' string at the end of the format method. This was entered by the researcher to account for results that will not have specific sections of the search, this is reflected as it only appeared on certain lines, for example org, location & version. For example, a returned search result might not belong to an organisation or not be able to have its identified location, but each result will always have an IP or port as this the main functionality for how Shodan performs its searches.

Finally, the researcher included some simple formatting to separate each set of returned results.

### 2.4.7 Saving results to document

After all the information is obtained, passed to a word document, and formatted to the researchers' standards, it is important to ensure the word document is stored correctly as to not lose the retrieved search. The '`results_from_search.save`' method allowed the researcher to do just this. By passing in the query document to the methods arguments, the researcher was able to include the inputted search query into the documents name allowing for clean and effective storage. Finally, a string is appended and given the .docx extension to ensure the document was stored.

```
#saves document and uses the user inputted query as the text file
results_from_search.save((query) + ' Search results document.docx')
```

Figure 2-12 Saving results to document

### 2.4.8 Error handling

It was crucial for the researcher to include some level of error handling in the script as the functionality of the script is not always guaranteed to work correctly. This is for numerous examples such as API connectivity issues, search results being incompatible or the search timing out, amongst numerous other errors that could occur. The following error exception was derived from Shodan documentation as correct practice, especially when used in tandem of the try block. The '`except shodan.APIError as e`' created an alias of the standard API error handling which was then easily printed in a following line. The error handling is the bare minimum required; therefore the error handling should be improved on and is an area the researcher highlights in section 3.4

```
#part of the error handling process, will print errors regarding API execution
except shodan.APIError as e:
    print ('Error: %s' % e)
```

Figure 2-13 Error Handling

## 2.5 OUTPUT

---

### 2.5.1 Headings

As the researcher previously considered, having headings to indicate to the other users of the script or allowing the researcher to use the script to report to an organisation in the form of organised reports would be very beneficial. Therefore, figure 2-14 shows the created headings implemented by the researcher. The headings correlate to those outlined in 2.4.5.1, providing summary information.

```
ShodanScan: Uncovering Vulnerable Devices with Python
Script by Jamie Rice, 2024
Your Entered Search Query Was == product:MongoDB version:3.4.10
Your search was produced at 06/05/2024 21:41:16
```

Figure 2-14 Headings of document

## 2.5.2 Format of output

Sample outputs can be seen in figure 2-15 and 2-16. What is interesting about the reports is that the benefits of some design choices made by the researcher can be seen directly in the output. For example, figure 2-15 searches for devices running MongoDB and a specific version which is vulnerable to a CVE outlined by an organisation as discussed in section 3.4. The output shows this in the *'Software:'* and *'Version'* labels.

```
-----  
IP: 82.214.164.202  
Port: 27017  
Version: 3.4.10  
Software: MongoDB  
Device: None  
Organisation: Intelligent Technologies S.A.  
Location: {'city': 'Warsaw', 'region_code': '14', 'area_code': None, 'longitude': 21.01178,  
'latitude': 52.22977, 'country_code': 'PL', 'country_name': 'Poland'}  
-----
```

Figure 2-15 MongoDB output

However, when looking at when the researcher inputted just the search term *'Device:webcam'* there is sometimes no given version number. The researcher accounted for this, therefore when this is applicable it is denoted by a *'Version N/A'*. If the researcher did not account for this, often results can be missed or errors are thrown on execution of the code. See appendix B for an example of output.

```
-----  
IP: 24.227.198.83  
Port: 80  
Version: N/A  
Software: GeoVision GeoHttpServer for webcams  
Device: webcam  
Organisation: Charter Communications Inc  
Location: {'city': 'Carrollton', 'region_code': 'TX', 'area_code': None, 'longitude':  
-96.89028, 'latitude': 32.95373, 'country_code': 'US', 'country_name': 'United States'}  
-----
```

Figure 2-16 Vulnerable device output



## 3 DISCUSSION

### 3.1 GENERAL DISCUSSION

As previously stated, the use case the report was centred around was a security researcher attempting to find devices that are vulnerable to three selected CVEs. The reason these CVEs were selected over others is highlighted by their widespread use across devices. Similarly, this is why when deciding on a device to search, the webcam was chosen. When showing the amount of potentially vulnerable webcams, it can be a very striking demonstration to the reader about how powerful the use of Shodan can be (Al-Alami, Hadi and Al-Bahadili, 2017). The question can be posed, if Shodan it can be used by security researchers to ensure organisations users are updating their software then it can be used by hackers looking to do malicious activities. This thought is often the rationale behind using Shodan as a motivation for individuals and organisations to update their technology or software.

To further demonstrate this popularity of the chosen technologies and devices, and how they intertwine, a facet analysis of webcam had been carried out on Shodan. Facet analysis is used as wide scope filter to show how different searches can show summary information of the search queries (John Matherly, 2017). Figure 3-1 highlights the use of nginx and Apache on products that are named or have in their name 'webcam'.

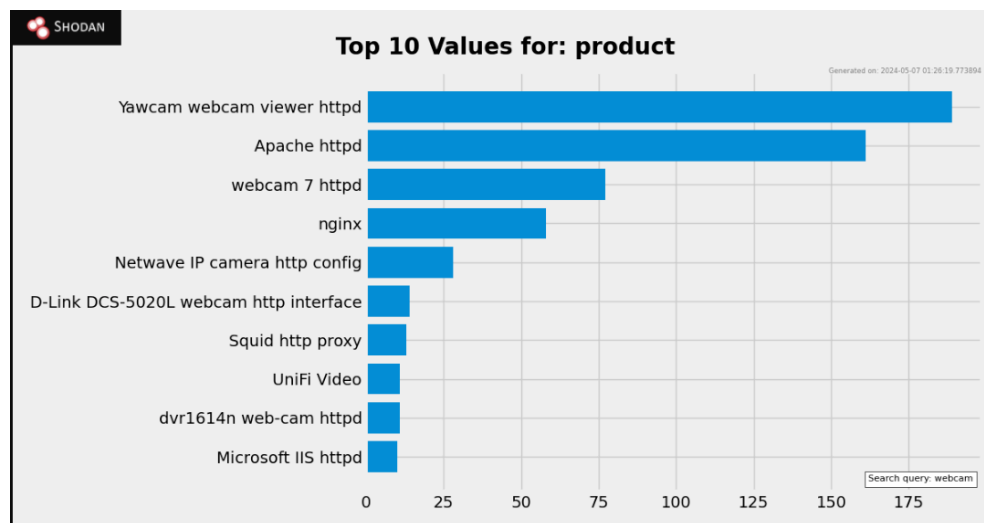


Figure 3-1 Demonstrating the top technologies for accessible webcams via Shodan

Source - <https://www.shodan.io/search/facet>

### 3.1.1 Chosen CVE's to demonstrate use case and outputs

The CVE's chosen to search for in this report were as follows:

- [CVE-2023-25690](#) - Apache HTTP Server versions 2.4.0 through 2.4.55 allow a HTTP Request Smuggling attack. (Apache Software Foundation and CVEdetails, 2024)
- [CVE-2017-15535](#) - MongoDB 3.4.x before 3.4.10, and 3.5.x-development, has a disabled-by-default configuration setting, networkMessageCompressors (aka wire protocol compression), which exposes a vulnerability when enabled that could be exploited by a malicious attacker to deny service or modify memory. (MITRE and CVEdetails, 2017)
- CVE-2024-24989 - When NGINX Plus or NGINX OSS are configured to use the HTTP/3 QUIC module, undisclosed requests can cause NGINX worker processes to terminate. (F5 Networks and CVEdetails, 2024)

### 3.1.2 Use Cases

#### 3.1.2.1 *Security Research for outdated products*

This is the use case the report is centred around an organisation coming to security researcher as a client to task the researcher to find all devices associated with a certain CVE (CVEs used are listed in the previous section). By utilising the created script, the researcher can efficiently identify devices or technologies associated with the CVE's and provide a formatted list to the organisation so they can begin contacting their customers to help update their product.

#### 3.1.2.2 *Network Security Audit*

Similarly, this could be when an organisation approaches the security researcher to help perform a network security audit of the organisation, this process would involve identifying potential risks and vulnerabilities within an organisation's infrastructure. The developed script would allow the researcher to identify open ports for exposed services and report these back to the client. This shows what devices of the organisations are not only vulnerable but open to the public, which for private companies can be quite damaging.

#### 3.1.2.3 *Penetration Test Engagement*

Finally, the script can be leveraged by those conducting a penetration test of an organisation. Like the use case outlined above however this can be highlighted as a more of an attacker approach when utilising the script. The created script can be used to perform OSINT (Open-Source Intelligence) on the target. Future work for the script will aim to introduce automatic penetration testing tasks such as automatic exploits such as SQL injections to be used against the devices acquired by the script.

## 3.2 DIFFICULTIES

---

- When this project was initially scoped out, the researcher believed there was an ability for the development of the 'vuln:' filter that can be used in Shodan to be used in the script. This filter allows for the user to search for specific vulnerabilities. Whilst this is available on the website version of Shodan, it is not available to implement as part of the script as it is not included in the API for this level of membership.
  - To get around this, the researcher had to alter the use case of the script, by centring the functionality of the script around a different use case it allowed for the script to be developed with the wider context in mind, focusing more on format of the word document and extra information such as date and time.
- Lack of feedback on input
  - When the user is inputting their query into the script, there is only an introductory sentence printed to screen to indicate to the user how their query should be formatted. The script currently does not look at different alterations of the query and there is no feedback regarding the correctness of the input provided.
- There is not enough sufficient feedback to the user when searching for something and it comes back with no results.
- Lack of error handling – now there is only the one try/except for the overall functionality of the script, however implementing more error handling would allow for the script to provide more feedback and better use from the user.

## 3.3 COUNTERMEASURES

---

It is important to highlight the countermeasures that should be put into place to prevent such a tool being used against an individual or an organisation's devices. As for the reasons previously outlined in the discussion section, it is vital to maintain a correct security posture when programs like Shodan exist.

- Device visibility management through firewalls or extra security on routers. Close ports for technologies open to web scanning
- Network segmentation to stop those who get access to the devices getting access to further information on the network, create access controls within the technologies
- Regularly update devices and ensure these patches are well communicated to customers to ensure they know how and when to update. As an individual it is important to regularly check the outlets of companies whose devices you use to handle sensitive information
- Security awareness is crucial within organisations to foster correct security practice in developing devices but also to help the individual user be more knowledgeable

### 3.4 FUTURE WORK

---

- Integration of Vulnerability Filter
  - Current level of API membership usage does not allow access to the vulnerability filter; therefore, it would be beneficial to get an organisation to pay for this membership to allow for a more tailored script to be made for their use case
- Integrate with CVE database and other vulnerability databases to allow for cross referencing.
  - Combining this with the vulnerability filter would allow for a very comprehensive vulnerability scanner to be made from the script
- Batch Search Handling
  - Whilst the user input allows flexibility around what the script searches for, it would be beneficial to allow for the user to input a batch of searches at once to automatically scan and create documents, this would allow for quicker penetration testing engagements for example. Additionally, would allow the researcher to allocate their time to something else.
- Multipage results
  - Once again restricted by the current membership to Shodan, upgrading the membership would allow for multiple pages of results to be used as they currently do not let this membership to do

# REFERENCES

- achillean (2014) *Basic Shodan Search — shodan-python 1.0 documentation*. Available at: <https://shodan.readthedocs.io/en/latest/examples/basic-search.html> (Accessed: May 7, 2024)
- Al-Alami, H., Hadi, A. and Al-Bahadili, H. (2017) 'Vulnerability scanning of IoT devices in Jordan using Shodan', *2017 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes & Systems (IT-DREPS)*, , pp. 1-6. doi: 10.1109/IT-DREPS.2017.8277814 <https://ieeexplore.ieee.org/document/8277814>
- Al-Sarawi, S., Anbar, M., Abdullah, R. and Al Hawari, A.B. (2020) 'Internet of Things Market Analysis Forecasts, 2020-2030', *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, , pp. 449-453. doi: 10.1109/WorldS450073.2020.9210375 <https://ieeexplore.ieee.org/document/9210375>
- Albataineh, A. and Alsmadi, I. (2019) 'IoT and the Risk of Internet Exposure: Risk Assessment Using Shodan Queries', *2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, , pp. 1-5. doi: 10.1109/WoWMoM.2019.8792986 <https://ieeexplore.ieee.org/document/8792986>
- Apache Software Foundation and CVEdetails (2024) *CVE-2023-25690 : Some mod\_proxy configurations on Apache HTTP Server versions 2.4.0 through 2.4.55 allow a HTTP Request Smuggling attack*. Available at: <https://www.cvedetails.com/cve/CVE-2023-25690/> (Accessed: May 7, 2024)
- Fernández-Caramés, T.M. and Fraga-Lamas, P. (2020) 'Teaching and Learning IoT Cybersecurity and Vulnerability Assessment with Shodan through Practical Use Cases', *Sensors (Basel, Switzerland)*, 20(11), pp. 3048. doi: 10.3390/s20113048 <https://pubmed.ncbi.nlm.nih.gov/PMC7309102>
- F5 Networks and CVEdetails (2024) *CVE-2024-24989 : When NGINX Plus or NGINX OSS are configured to use the HTTP/3 QUIC module, undisclosed requests can cause NGINX worker p*. Available at: <https://www.cvedetails.com/cve/CVE-2024-24989/> (Accessed: May 7, 2024)

Genge, B. and Enăchescu, C. (2016) 'ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services', *Security and Communication Networks*, 9(15), pp. 2696-2714. doi: 10.1002/sec.1262 <https://onlinelibrary.wiley.com/doi/abs/10.1002/sec.1262>

John Matherly (2017) *Complete Guide to Shodan*. 3rd edn. Lean Publishing [https://ucilnica.fri.uni-lj.si/pluginfile.php/160496/mod\\_resource/content/1/Matherly%2C%20J.%20\(2016\).%20The%20Complete%20Guide%20to%20Shodan.pdf](https://ucilnica.fri.uni-lj.si/pluginfile.php/160496/mod_resource/content/1/Matherly%2C%20J.%20(2016).%20The%20Complete%20Guide%20to%20Shodan.pdf) May 7, 2024

Matt, B. (2017) *Scripting Saturday: Shodan via Python | by Matt B | Medium*. Available at: <https://bromiley.medium.com/scripting-saturday-shodan-via-python-21b5c55f7a6b> (Accessed: May 7, 2024)

MITRE and CVEdetails (2017) *CVE-2017-15535 : MongoDB 3.4.x before 3.4.10, and 3.5.x-development, has a disabled-by-default configuration setting, networkMessageCompr*. Available at: <https://www.cvedetails.com/cve/CVE-2017-15535/> (Accessed: May 7, 2024)

Ofri Ouzan (2023) *Advanced Shodan Use for Tracking Down Vulnerable Components*. Available at: <https://medium.com/@ofriouzan/advanced-shodan-use-for-tracking-down-vulnerable-components-7b6927a87c45> (Accessed: May 7, 2024)

TAKHION (2019) *The Hacks of Mr. Robot: How to Use the Shodan API with Python to Automate Scans for Vulnerable Devices*. Available at: <https://null-byte.wonderhowto.com/how-to/hacks-mr-robot-use-shodan-api-with-python-automate-scans-for-vulnerable-devices-0180975/> (Accessed: May 7, 2024)