# AUTONOMOUS VULNERABILITY SCANNER FOR NETWORK SECURITY

#### 

#### A PROJECT REPORT

***Submitted by***

**MUKUNTH VENKATESH N G [REGISTER NO: 211418104161] SANTHOSHKUMAR A [REGISTER NO: 211418104228]**

**MATHESWARAN K K [REGISTER NO: 211418104149]**

***in partial fulfillment for the award of the degree***

***of***

## BACHELOR OF ENGINEERING

***in***

**COMPUTER SCIENCE AND ENGINEERING**



**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**MAY 2022**

**PANIMALAR ENGINEERING COLLEGE**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

## BONAFIDE CERTIFICATE

# Certified that this project report “ AUTONOMOUS VULNERABILITY SCANNER FOR NETWORK SECURITY ” is the bonafide work of “MUKUNTH VENKATESH N G(211418104161) , SANTHOSHKUMAR A(211418104228), MATHESWARAN K K(211418104149)” who carried out the

project work under my supervision.

#### SIGNATURE SIGNATURE

**Dr.S.MURUGAVALLI, M.E., Ph.D., Dr.S.BALAJI, BTech, M.E, PhD;**

#### HEAD OF THE DEPARTMENT, ASSOCIATE PROFESSOR,

DEPARTMENT OF CSE, DEPARTMENT OF CSE,

PANIMALAR ENGINEERING COLLEGE, PANIMALAR ENGINEERING COLLEGE,

POONAMALLEE, POONAMALLEE,

CHENNAI-600 123. CHENNAI-600 123

Certified that the above mentioned students were examined in End Semester project viva-voice

held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**INTERNAL EXAMINER EXTERNAL EXAMINER**

## DECLARATION BY THE STUDENT

#### We MUKUNTH VENKATESH N G(211418104161) , SANTHOSHKUMAR A (211418104228), MATHESWARAN K K(211418104149), hereby declare that this project report titled“ AUTONOMOUS VULNERABILITY SCANNER FOR NETWORK SECURITY”, under the guidance of Dr.S.BALAJI is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

**ACKNOWLEDGEMENT**

We would like to express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We express our sincere thanks to our Directors **Tmt.C.VIJAYARAJESWARI**, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Dr.**

**SARANYASREE SAKTHIKUMAR, B.E., M.B.A.,Ph.D.,** for providing us with the necessary facilities to undertake project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** who facilitated us in completing the project.

We thank the Head of the CSE Department, **Dr. S.MURUGAVALLI, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Dr.S.BALAJI,B.Tech,M.E,Ph.D,** and all the faculty members of the Department of CSE for their advice and encouragement for the successful completion of the project.

MUKUNTH VENKATESH N G

SANTHOSHKUMAR A MATHESWARAN K K

#### ABSTRACT

Vulnerability Scanning is a methodology to identify and report on security issue that affect a system. The aim of this project is to analyze all the ports which are open globally and show vulnerability if it is vulnerable. Initially ports are scanned for a specific IP and determine whether the port is open or not. Since project makes use of 5 Ports FTP, TELNET, RDP, SMB, SMTP, the open ports are tested for a vulnerability. The data collected will be stored in python dictionary and then it is converted directly into JSON and finally uploaded to Mongo DB. By vulnerability, we mean, the potential flaws in the system that make it prone to the attack. Assessment of these system vulnerabilities provide a means to identify and develop new strategies so as to protect the system from the risk of being damaged. The objective of this project is to scan the IP and then to identify the vulnerability present in the IP. This helps the security researcher to analyze and assess a solution for the vulnerability present in the system. This methodology to detect the various vulnerabilities available in the web applications or the remote host across the network and tries to identify new mechanisms that can be deployed to secure the network.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.**  **1.** | **TITLE**  **ABSTARCT**  **INTRODUCTION** | **PAGE NO.**  v 1 |
|  | 1.1 Overview | 2 |
| **2.** | **LITERATURE SURVEY** | 3 |
| **3.** | **SYSTEM ANALYSIS** | 8 |
|  | 3.1 Existing System | 9 |
|  | 3.2 Proposed system | 9 |
| 3.3 Feasibility Study | 9 |
|  | 3.3.1 Technical Feasibility | 9 |
|  | 3.3.2 Economic Feasibility | 10 |
|  | 3.4 Hardware Requirements | 10 |
| 3.5 Software Requirements | 10 |
| 3.5 Software Specification | 11 |

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
| **4.** | **SYSTEM DESIGN** |  |
|  | 4.1 Data Dictionary | 17 |
|  | 4.2 Data Flow Diagram | 18 |
|  | 4.3 UML Diagrams | 19 |
| **5.** | **SYSTEM ARCHITECTURE** | 25 |
|  | 5.1 Architecture Overview | 26 |
|  | 5.2 Modules Explanation | 27 |
|  | 5.3 Algorithm | 30 |
| **6.** | **SYSTEM IMPLEMENTATION** | 31 |
|  | 6.1 Client-side Coding | 32 |
|  | 6.2 Server-side Coding | 45 |
| **7.** | **CONCLUSION** | 59 |
|  | 7.1 Conclusion | 60 |
|  | 7.2 Future Enhancement | 60 |
|  | **APPENDICES**  A.1 Sample Screens  **REFERENCES** | 64  61  66 |

# CHAPTER 1 INTRODUCTION

1

## 1.1 OVERVIEW

The Emergence of information technology, the security aspect of the users has become a more concerned factor. Since most of the software developers are not aware of various security measures to be introduced into the system as their motive is just to make the software application run in a desired state without taking into consideration the flaws that the programming language might have introduced into the system; to protect the users from the risk of being attacked by any unauthorised access, it becomes significantly more important to devise new strategies and methodologies that will consider the security breaches to which the user is prone to. Not only the software developed with flaws makes the user vulnerable to attacks, most often network also becomes a key factor by compromising the security aspect of the users.

Assessing and eliminating the vulnerabilities requires the knowledge and deep understanding of these vulnerabilities. It becomes necessary enough to know the basic idea that works behind these vulnerabilities such as what makes them to appear in the system, what flaws need to be corrected to make the system free from these vulnerabilities, what alternatives can be further devised for these vulnerabilities so that in future, their risk can be reduced and many more.

Various methods have been deployed to identify these vulnerabilities and appropriate steps are taken.

Strategies such as static analysis, attack graph generation and its analysis, usage of vulnerability scanners are some of them. However, the use of vulnerability scanners to detect the vulnerabilities is quite prominent today. They play a significant role in the generation of attack graphs.

Automated vulnerability scanner offers a much faster and experience when comes to vulnerabilities ,however the speed and functions of Automated vulnerability can affect by following factors.

* Type of connection:Remote and local
* Authentication:Unauthoried scan
* Number of host being scanned

# CHAPTER 2 LITERATURE SURVEY

## Performance Comparison on SQL Injection and XSS Detection using Open Source Vulnerability Scanners

Automatic software vulnerability scanners are common tools for detecting such vulnerabilities among software developers. It helps to illustrate the program for the attacker by creating a great deal of engagement within the program. SQL Injection and Cross-Site Scripting (XSS) are two of the most commonly spread and dangerous vulnerabilities in web apps that cause to the user. It is very important to trust the findings of the site vulnerability scanning software. Without a clear idea of the accuracy and the coverage of the open- source tools, it is difficult to analyze the result from the automatic vulnerability scanner that provides. The important to do a comparison on the key figure on the automated vulnerability scanners because there are many kinds of a scanner on the market and this comparison can be useful to decide which scanner has better performance in term of SQL Injection and Cross-Site Scripting (XSS) vulnerabilities. In this paper, a method by Jose Fonseca et al, is used to compare open-source automated vulnerability scanners based on detection coverage and a method by Yuki Makino and Vitaly Klyuev for precision rate. The criteria vulnerabilities will be injected into the web applications which then be scanned by the scanners. The results then are compared by analyzing the precision rate and detection coverage of vulnerability detection. Two leading open source automated vulnerability scanners will be evaluated. In this paper, the scanner that being utilizes is OW ASP ZAP and Skipfish for comparison.

## Improving Vulnerability Scanner Performance in Detecting AJAX

The development of internet and web application gave birth to a new technology called AJAX. Web application utilizes this technology and create a new trend in web application - AJAX application. As a web application, there are some security issues related to AJAX application, one of which is the existence of web application vulnerabilities. These vulnerabilities can be detected using a tool called vulnerability

scanner. A vulnerability scanner has three processes - crawling, attack and analysis. However, AJAX's characteristics makes it hard for vulnerability scanner to do its job properly. One of the reason is the ability to change the content of DOM without reloading the page, called dynamic DOM, in AJAX application. A dynamic DOM makes it impossible for conventional crawling method to crawl an AJAX application properly and thus, reducing the overall performance of vulnerability scanner. A method to crawl an AJAX application is called AJAX crawling. In this paper, we propose a method to improve the performance of vulnerability scanner by implementing AJAX crawling as its crawling process. This method has been tested by scanning a news aggregator web application called Gregarius version 0.5.2. From the result, this method can be used to detect vulnerabilities in AJAX application.

## Evaluation of web vulnerability scanner

In recent yeas a lot of web applications have been released in the world. At the same time, cyber attacks against web application vulnerabilities have also increased. In such a situation, it is necessary to make web applications more secure. However checking all web vulnerabilities by hand is very difficult and time-consuming. Therefore, we need a web application vulnerability scanner. In this work, we evaluate two open source vulnerability scanners OWASP Zed Attack Proxy (OWASP ZAP) and Skipfish using vulnerable web applications Damn Vulnerable Web Application (DVWA) and The Web Application Vulnerability Scanner Evaluation Project (WAVSEP).

## A cost-effective approach to evaluating security vulnerability scanner

Web applications are exposed to various threats and attacks, and therefore numerous tools are developed for detecting web application vulnerabilities. Many studies have focused on evaluating vulnerability scanners. An efficient evaluation approach for

5

detection tools is essential and can be extremely helpful to the users. In this paper, we propose a cost-effective approach to evaluating vulnerability scanners by considering redundant vulnerability alert problem. We define the redundant alert problem in scanner evaluation with our motivational example and propose the advanced confusion matrix by extending two defined attributes, true duplication (TD) and false duplication (FD). Then we apply our proposed cost-effective evaluation approach and build up the web Vulnerability Scanner Testbed.

## Common Requirements for Web Application Vulnerability Scanners for the Internet of Things

This paper presents 3 common requirements for web application vulnerability scanners for the Internet of things devices, including browser's rendering engine support, false positive minimization, and device setting change minimization. These requirements have been drawn from the experience of the previous project, security vulnerabilities in residential gateways.

### An Automatic Vulnerability Scanner for Web Applications

With the progressive development of web applications and the urgent requirement of web security, vulnerability scanner has been particularly emphasized, which is regarded as a fundamental component for web security assurance. Various scanners are developed with the intention of that discovering the possible vulnerabilities in advance to avoid malicious attacks. However, most of them only focus on the vulnerability detection with single target, which fail in satisfying the efficiency demand of users. In this paper, an effective web vulnerability scanner that integrates the information collection with the vulnerability detection is proposed to verify whether the target web application is vulnerable or not. The experimental results

6

show that, by guiding the detection process with the useful collected information, our tool achieves great web vulnerability detection capability with a large scanning scope.

### A Systematic Literature Review on the Characteristics and Effectiveness of Web Application Vulnerability Scanners

Web applications have been a significant target for successful security breaches in the last few years. They are currently secured, as a primary method, by searching for their vulnerabilities with specialized tools referred to as Web Application Vulnerability Scanners (WVS’s). Although, these dynamic approaches of testing have some advantages, there is still a scarcity of studies that explore their features and detection capabilities in a systematic way. This article reports findings from a Systematic Literature Review (SLR) to look into the characteristics and effectiveness of the most frequently used WVS’s. A total of 90 research papers were carefully evaluated. Thirty (30) WVS’s were collected and reported, with only 12 having at least one quantitative assessment of effectiveness. These

12 WVS’s were evaluated by 15 original evaluation studies. We found that these evaluations tested mostly only two of the Open Web Application Security Project (OWASP) Top Ten vulnerability types: SQL injection (SQLi) (13/15) and Cross-Site Scripting (XSS) (8/15). Additionally, only one work evaluated six of the OWASP Top Ten vulnerability types and for only one scanner. We also found that the reported detection rates were highly dissimilar between these 15 evaluations. Based on these surprising results we suggest avenues for future directions.

7

# CHAPTER 3 SYSTEM ANALYSIS

8

## EXISTING SYSTEM

The existing system uses the synchronous method and they use the simple scanning method to ping an IP address.When a synchronous method is invoked, it completes executing before returning to the caller. A synchronous program is executed one step at a time. Even with conditional branching, loops and function calls, you can still think about the code in terms of taking one execution step at a time. When each step is complete, the program moves on to the next one.

## PROPOSED SYSTEM

The proposed method uses the asynchronous method. An asynchronous method starts a job in the background and returns to the caller immediately. An **asynchronous program** behaves differently. It still takes one execution step at a time. The difference is that the system may not wait for an execution step to be completed before moving on to the next one.This means that the program will move on to future execution steps even though a previous step hasn’t yet finished and is still running elsewhere. This also means that the program knows what to do when a previous step does finish running.In this project we use zmap which gives the better performance than the other map tools.

### FEASIBILITY STUDY

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

### TECHNICAL FEASIBILITY

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

9

### ECONOMIC FEASIBILITY

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of increase in the expenses. Programming Language for Web-App development is open source and readily available for all users.

### HARDWARE REQUIREMENTS

|  |  |
| --- | --- |
| STORAGE REQUIRED | 500 MB |
| PROCESSOR | INTEL i5 |
| SYSTEM TYPE | 64 bit |
| RAM | >4GB |

* 1. **SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| OPERATING SYSTEM | LINUX |
| PROGRAMMING LANGUAGE (BACKEND) | PYTHON |
| PROGRAMMING LANGUAGE (FRONTEND) | HTML, CSS |
| DATABASE | MONGO DB |
| APPLICATION PROGRAMMING INTERFACE (API) | FLASK |

## SOFTWARE SPECIFICATION

* + 1. **LINUX**

Linux is a family of [open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [Unix-like](https://en.wikipedia.org/wiki/Free_and_open-source_software) [operating systems](https://en.wikipedia.org/wiki/Operating_system) based on the [Linux](https://en.wikipedia.org/wiki/Linux_kernel) [kernel](https://en.wikipedia.org/wiki/Linux_kernel), an [operating system kernel](https://en.wikipedia.org/wiki/Kernel_(operating_system)) first released on September 17, 1991, by [Linus](https://en.wikipedia.org/wiki/Linus_Torvalds) [Torvalds](https://en.wikipedia.org/wiki/Linus_Torvalds). Linux is typically [packaged](https://en.wikipedia.org/wiki/Package_manager) in a [Linux distribution](https://en.wikipedia.org/wiki/Linux_distribution) interface . Linux was originally developed for [personal computers](https://en.wikipedia.org/wiki/Personal_computer) based on the [Intel x86](https://en.wikipedia.org/wiki/Intel_x86) architecture, but has since been [ported](https://en.wikipedia.org/wiki/Porting) to more [platforms](https://en.wikipedia.org/wiki/Computer_hardware_platforms) than any other operating system. Because of the dominance of the Linux-based [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) on [smartphones](https://en.wikipedia.org/wiki/Smartphone), Linux also has the [largest installed](https://en.wikipedia.org/wiki/Usage_share_of_operating_systems) [base](https://en.wikipedia.org/wiki/Installed_base) of all [general-purpose operating systems](https://en.wikipedia.org/wiki/General-purpose_operating_system). Linux also runs on [embedded systems](https://en.wikipedia.org/wiki/Embedded_system), i.e. devices whose operating system is typically built into the [firmware](https://en.wikipedia.org/wiki/Firmware) and is highly tailored to the system. Linux is one of the most prominent examples of free and open- source [software](https://en.wikipedia.org/wiki/Software) collaboration. The [source code](https://en.wikipedia.org/wiki/Source_code) may be used, modified and distributed commercially or non-commercially by anyone under the terms of its respective licenses, such as the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License).

## PYTHON

Python is an [interpreter](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) and [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Its’s design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant](https://en.wikipedia.org/wiki/Off-side_rule) [indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before

executing it. This is similar to PERL and PHP. Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python is Object-Oriented − Python supports Object- Oriented style or technique of programming that encapsulates code within objects. Pythonis a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

## HTML

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for textsuch as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity

references. HTML tags most commonly come in pairs like <h1> and

</h1>, although some represent empty elements and so are unpaired, for example

<img>.The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags). HTML documents imply a structure of nested HTML elements. These are indicated in the document by HTML tags, enclosed in angle brackets thus: In the simple, general case, the extent of an element is indicated by a pair of tags: a "start tag" <p> and "end tag"</p>. The text content of the element, if any, is placed between these tags.

* + 1. **CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technologyof the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specificationof presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes fromthe specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

### FLASK

Flask is a micro [webframework](https://en.wikipedia.org/wiki/Web_framework) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework) because it does not require particular tools or libraries. It has

no [database](https://en.wikipedia.org/wiki/Database) abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

### AJAX

Ajax is a set of [web development](https://en.wikipedia.org/wiki/Web_development) techniques that uses various web technologies on the [client-side](https://en.wikipedia.org/wiki/Client-side) to create asynchronous [web applications](https://en.wikipedia.org/wiki/Web_application). With Ajax, web applications can send and retrieve data from a [server](https://en.wikipedia.org/wiki/Web_server) asynchronously (in the background) without interfering with the display and behaviour of the existing page. By decoupling the data interchange layer from the presentation layer, Ajax allows web pages and, by extension, web applications, to change content dynamically without the need to reload the entire page. In practice, modern implementations commonly utilize [JSON](https://en.wikipedia.org/wiki/JSON) instead of XML. Ajax is not a technology, but rather a programming concept. [HTML](https://en.wikipedia.org/wiki/Hypertext_Markup_Language) and [CSS](https://en.wikipedia.org/wiki/Cascading_Style_Sheets) can be used in combination to mark up and style information. The webpage can be modified by JavaScript to dynamically display—and allow the user to interact with the new information. The built- in [XMLHttpRequest](https://en.wikipedia.org/wiki/XMLHttpRequest) object is used to execute Ajax on webpages, allowing websites to load content onto the screen without refreshing the page.

### Z-MAP

ZMap is a [free and open-source security scanner](https://en.wikipedia.org/wiki/Free_and_open-source_software) that was developed as a faster alternative to [Nmap](https://en.wikipedia.org/wiki/Nmap). ZMap was designed for [information security](https://en.wikipedia.org/wiki/Information_security) research and can be used for both [white hat](https://en.wikipedia.org/wiki/White_hat_(computer_security)) and [black hat](https://en.wikipedia.org/wiki/Black_hat_(computer_security)) purposes. The tool is able to discover [vulnerabilities](https://en.wikipedia.org/wiki/Vulnerability_(computing)) and their impact, and detect affected [IoT](https://en.wikipedia.org/wiki/Internet_of_things) devices.Using one [gigabit](https://en.wikipedia.org/wiki/Gigabit) per second of [network](https://en.wikipedia.org/wiki/Bandwidth_(computing)) [bandwidth](https://en.wikipedia.org/wiki/Bandwidth_(computing)), ZMap can scan the entire [IPv4](https://en.wikipedia.org/wiki/IPv4) [address space](https://en.wikipedia.org/wiki/IPv4) in 44 minutes on a single [port](https://en.wikipedia.org/wiki/Port_(computer_networking)). With a ten gigabit connection, ZMap scan can complete a scan in under five minutes. ZMap iterates on techniques utilized by its predecessor, [Nmap](https://en.wikipedia.org/wiki/Nmap), by altering the scanning method in a few key areas. Nmap sends out individual signals to each IP address and waits

for a reply. As replies return, Nmap compiles them into a database to keep track of responses, a process that slows down the scanning process. In contrast, ZMap uses [cyclic](https://en.wikipedia.org/wiki/Cyclic_group) [multiplicative groups](https://en.wikipedia.org/wiki/Cyclic_group), which allows ZMap to scan the same space roughly 1,300 times faster than Nmap. The ZMap software takes every number from 1 to 232-1 and creates an iterative formula that ensures that each of the possible [32-bit](https://en.wikipedia.org/wiki/32-bit) numbers is visited once in a [pseudorandom](https://en.wikipedia.org/wiki/Pseudorandomness) order. ZMap also speeds up the scanning process by sending a probe to every IP address only once by default, whereas Nmap resends a probe when it detects a connection delay or fails to get a reply. This results in about 2% of IP addresses being missed during a typical scan, but when processing billions of IP address, or potential [IoT](https://en.wikipedia.org/wiki/Internet_of_things) devices being targeted by [cyberattackers](https://en.wikipedia.org/wiki/Cyberattack), 2% is an acceptable [tolerance](https://en.wikipedia.org/wiki/Engineering_tolerance).

#### MONGO DB

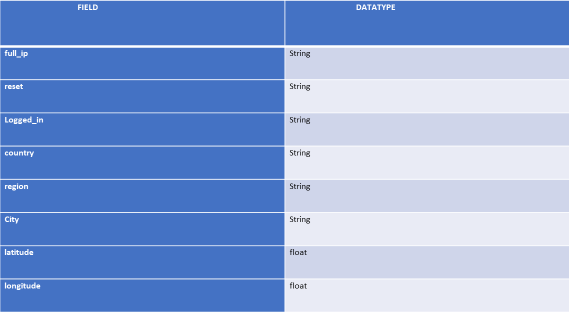
MongoDB is an open source [NoSQL](https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL) database management program. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document- oriented information, store or retrieve information.MongoDB supports various forms of data. It is one of the many nonrelational [database](https://searchsqlserver.techtarget.com/definition/database) technologies that arose in the mid- 2000s [under the NoSQL banner](https://www.techtarget.com/searchdatamanagement/infographic/NoSQL-database-comparison-to-help-you-choose-the-right-store) -- normally, for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in [relational databases](https://www.techtarget.com/searchdatamanagement/definition/relational-database), the MongoDB architecture is made up of collections and documents.Organizations can use Mongo DB for its ad-hoc queries, indexing, load balancing, aggregation, server-side JavaScript execution and other features.

# CHAPTER 4 SYSTEM DESIGN

### DATA DICTIONARY

A data dictionary, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". Oracle defines it as a collection of tables with metadata. The term can have several closely related meanings pertaining to databases and database management systems (DBMS):

* + - A document describing a database or collection of databases user
    - An integral component of a DBMS that is required to determine its structure
    - A piece of middleware that extends or supplants the native data dictionary of a DBMS.



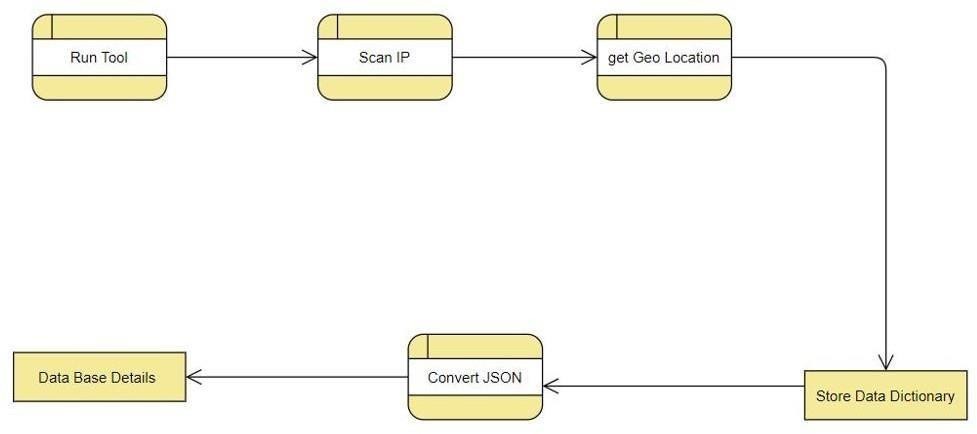
#### DATA DICTIONARY

17

* 1. **DATA FLOW DIAGRAM**

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system. It is usually beginning with a context diagram as the level 0 of the DFD diagram, a simple representation of the whole system. To elaboratefurther from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing function really depends on the complexity that function.

18



**DATA FLOW DIAGRAM**

## UML DIAGRAMS

UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes.

There are 14 UML diagram types. Unified Modeling Language (UML) is a standard visual modeling language intended to be used for

* modeling business and similar processes,
* analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

UML is intentionally process independent and could be applied in the context of different

19

processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).UML is not complete, and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases. Name of an abstract classifier is shown in italics while the final classifier has no specific graphical notation, so there is no way to determine whether the classifier is final or not from the diagram.

## List of UML Diagram Types

There are two main categories; structure diagrams and behavioral diagrams.

## Structure Diagrams

The structural diagrams represent the static aspect of the system. These static aspects

represent those parts of a diagram, which forms the main structure and are therefore

stable.These static parts are represented by classes, interfaces, objects, components, and

nodes.

## Class Diagram

Class diagrams are the most common diagrams used in UML. Class diagram consists of classes, interfaces, associations, and collaboration. Class diagrams basically represent the object-oriented view of a system, which is static in nature.Active class is used in a class diagram to represent the concurrency of the system.Class diagram represents the object orientation of a system. Hence, it is generally used for development purpose. This is the most widely used diagram at the time of system construction.

## Component Diagram

Component diagrams represent a set of components and their relationships. These components consist of classes, interfaces, or collaborations. Component diagrams represent the implementation view of a system.During the design phase, software artifacts (classes, interfaces, etc.) of a system are arranged in different groups depending upon their relationship. Now, these groups are known as components.Finally, it can be said component diagrams are used to visualize the implementation.

## Deployment Diagram

Deployment diagrams are a set of nodes and their relationships. These nodes are physical entities where the components are deployed.Deployment diagrams are used for visualizing the deployment view of a system. This is generally used by the deployment team.

## BEHAVIORAL DIAGRAMS

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered.Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

## Use Case Diagram

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system.A use case represents a particular functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as **actors**.

## Activity Diagram

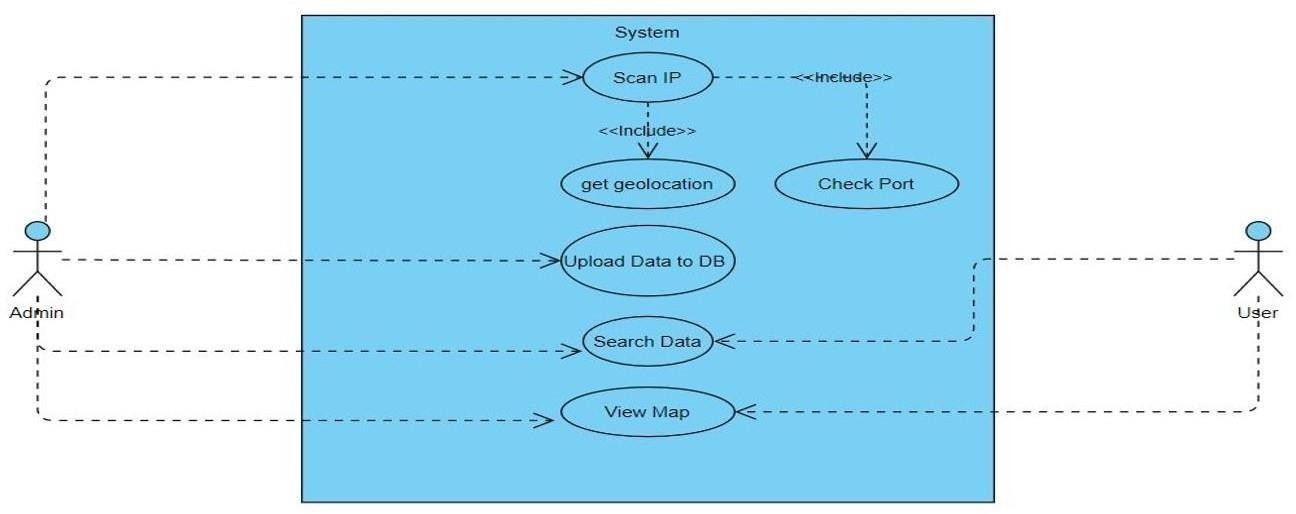
Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams.

## Sequence Diagram

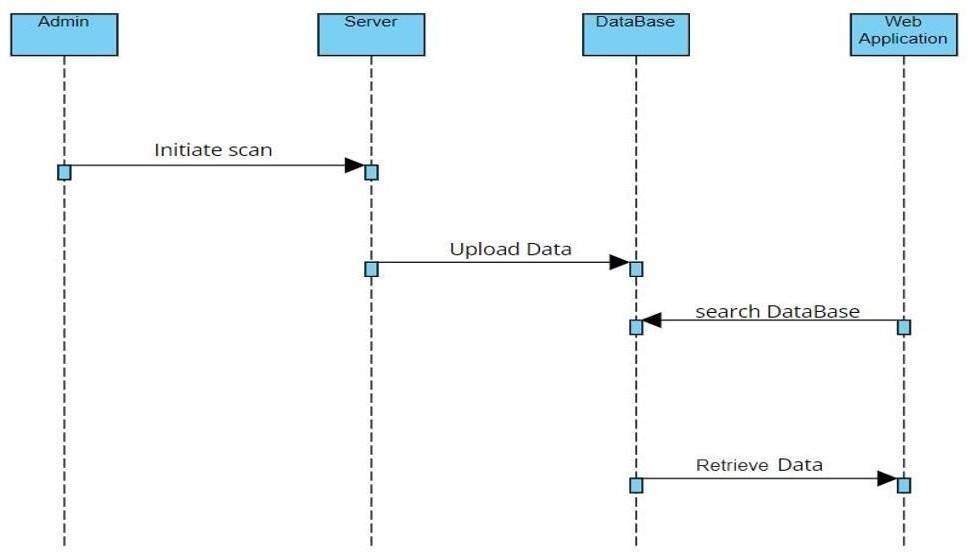
A sequence diagram is an interaction diagram. From the name, it is clear that the diagram deals with some sequences, which are the sequence of messages flowing from one object to another.Interaction among the components of a system is very important from implementation and execution perspective. Sequence diagram is used to visualize the sequence of calls in a system to perform a specific functionality.

## Collaboration Diagram

Collaboration diagram is another form of interaction diagram. It represents the structural organization of a system and the messages sent/received. Structural organization consists of objects and links.The purpose of collaboration diagram is similar to sequence diagram. However, the specific purpose of collaboration diagram is to visualize the organization of objects and their interaction.



#### USE CASE DIAGRAM



**SEQUENCE DIAGRAM**

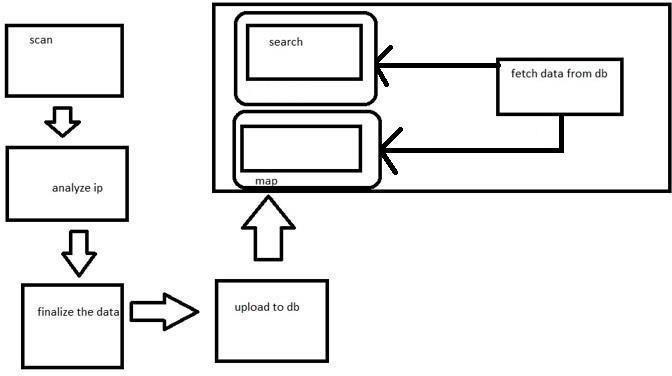
24

# CHAPTER 5

**SYSTEM ARCHITECTURE**

## ARCHITECTURE OVERVIEW

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.



## MODULE EXPLANATION

* + 1. **JSON**

JSON (JavaScript Object Notation), is an [open standard](https://en.wikipedia.org/wiki/Open_standard) [file format](https://en.wikipedia.org/wiki/Open_standard) and [data](https://en.wikipedia.org/wiki/Electronic_data_interchange) [interchange](https://en.wikipedia.org/wiki/Electronic_data_interchange) format that uses [human-readable](https://en.wikipedia.org/wiki/Human-readable_medium) text to store and transmit data objects consisting of [attribute–value pairs](https://en.wikipedia.org/wiki/Attribute%E2%80%93value_pair) and [arrays](https://en.wikipedia.org/wiki/Array_data_type) (or other [serializable](https://en.wikipedia.org/wiki/Serialization) values). It is a common data format with diverse uses in [electronic data interchange](https://en.wikipedia.org/wiki/Electronic_data_interchange), including that of [web](https://en.wikipedia.org/wiki/Web_application) [applications](https://en.wikipedia.org/wiki/Web_application) with [servers](https://en.wikipedia.org/wiki/Server_(computing)).JSON is a [language-independent](https://en.wikipedia.org/wiki/Language-independent_specification) data format. It was derived from [JavaScript](https://en.wikipedia.org/wiki/JavaScript), but many modern [programming languages](https://en.wikipedia.org/wiki/Programming_language) include code to generate and [parse](https://en.wikipedia.org/wiki/Parsing) JSON-format data. JSON filenames use the extension .json .

## ASYNCIO

asyncio is a library to write concurrent code using the async/await syntax.asyncio is used as a foundation for multiple Python asynchronous frameworks that provide high-performance network and web-servers, database connection libraries, distributed task queues, etc.asyncio is often a perfect fit for IO-bound and high-level structured network code.asyncio provides a set of high-level APIs to:

* [run Python coroutines](https://docs.python.org/3/library/asyncio-task.html#coroutine) concurrently and have full control over their execution;
* perform [network IO and IPC](https://docs.python.org/3/library/asyncio-stream.html#asyncio-streams);
* control [subprocesses](https://docs.python.org/3/library/asyncio-subprocess.html#asyncio-subprocess);
* distribute tasks via [queues](https://docs.python.org/3/library/asyncio-queue.html#asyncio-queues);
* [synchronize](https://docs.python.org/3/library/asyncio-sync.html#asyncio-sync) concurrent code;

## FOLIUM

Folium is a powerful Python library that helps you create several types of Leaflet maps. By default, Folium creates a map in a separate HTML file. Since Folium results are interactive, this library is very useful for dashboard building. You can also create inline Jupyter maps in Folium.Folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Using Folium, you can manipulate your data in Python, then visualize it in a Leaflet map.Folium enables you to generate a base map of specified width and height with either default tilesets (i.e., map styles) or a custom tileset URL. The following tilesets are available by default with Folium:

* + - * OpenStreetMap
      * Mapbox Bright
      * Mapbox Control Room
      * Stamen (incl. Terrain, Toner, and Watercolor)
      * Cloudmade
      * Mapbox

## URLLIB

Urllib package is the URL handling module for python. It is used to fetch URLs (Uniform Resource Locators). It uses the *urlopen* function and is able to fetch URLs using a variety of different protocols.

Urllib is a package that collects several modules for working with URLs, such as:

* + - * urllib.request for opening and reading.
      * urllib.parse for parsing URLs
      * urllib.error for the exceptions raised
      * urllib.robotparser for parsing robot.txt files

urllib.request

This module helps to define functions and classes to open URLs (mostly HTTP). One of the most simple ways to open such URLs is : *urllib.request.urlopen(url)*

## REGEX

A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching with a particular pattern, and also can split a pattern into one or more sub- patterns. Python provides a re module that supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none.Regular expressions use the backslash character ('\') to indicate special forms or to allow special characters to be used without invoking their special meaning. This collides with Python’s usage of the same character for the same purpose in string literals.

## ALGORITHM

#### SERVER

Step 1: Scan for all the publically available IP address

Step 2: Check if the specified port is open or not for that IP while scanning Step 3: If open collect the information about the IP

Step 4: After scanning all the IP upload all the data to the database

**CLIENT**

Step 1: User can view the map or search information about a particular IP Step 2: If user want to view an IP he/she has to enter the IP

# CHAPTER 6

**SYSTEM IMPLEMENTATION**

31

#### CLIENT -SIDE CODING SPLIT.PY

from flask import Flask, render\_template, request, jsonify from pymongo import MongoClient

import urllib import json

from branca.element import Template, MacroElement import urllib

import folium

app = Flask( name )

html\_page = """<!DOCTYPE HTML>

<html>

<head>

<title>Rough AJAX Test</title>

<script>

function loadXMLDoc()

{

var req = new XMLHttpRequest() req.onreadystatechange = function()

{

if (req.readyState == 4)

{

if (req.status != 200)

{

//error handling code here

}

else

{

var response = JSON.parse(req.responseText)

var output ="<h1> OUTPUT DATA</h1>"+"<br>" var dat ="<br>"

for (key in response){

var value=response[key];

if(typeof value === 'response'){ console.log('{'); logRecursive(value) console.log('}');

}else{

console.log(JSON.stringify(value, null, 2));

for (keyy in value){

var valuee=value[keyy]; if(typeof value === 'value'){

console.log('{'); logRecursive(valuee) console.log('}');

}else{

console.log(JSON.stringify(valuee, null, 2));

output +=keyy+" : "+JSON.stringify(valuee, null, 2) + "<br>"

}

}

output+="<br><br>"

}

}

document.getElementById('myDiv').innerHTML = output

}

}

}

req.open('POST', '/searching')

req.setRequestHeader("Content-type", "application/x-www-form-urlencoded") var un = document.getElementById('scname').value

var sec = document.getElementById('secret').value

var postVars = 'username='+un+'&secret='+sec req.send(postVars)

return false

}

</script>

</head>

<body>

<h1>Flask AJAX Test</h1>

<form action="" method="POST">

<input type="text" name="scname" id="scname">

<input type="hidden" name="secret" id="secret" value="shhh">

<input type="button" value="Submit" onclick="return loadXMLDoc()">

</form>

<div id="myDiv"></div>

</body>

</html>"""

@app.route('/search',methods = ['POST']) def search\_index():

return html\_page

@app.route('/searching', methods = ['POST']) def ajax\_request():

out = request.form['username']

# FTP try:

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/FTP\_DB?retryWrites=true&w=majority") db = client["FTP\_DB"]

ftp = db['ftp']

a = ftp.find\_one() a.pop("\_id")

aa = {"\_ftp\_":"the following is ftp data"} if out in a:

aa.update(a[out]) else:

x={"ftp":"no FTP data on this IP"} aa.update(x)

except:

pass

# RDP try:

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/RDP\_DB?retryWrites=true&w=majority") db = client["RDP\_DB"]

rdp = db['rdp']

b = rdp.find\_one() b.pop("\_id")

bb={"\_rdp\_":"the following data is rdp"} if out in b:

bb.update(b[out]) else:

x={"rdp":"no RDP data on this IP"} bb.update(x)

except:

pass

# SMB try:

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMB\_DB?retryWrites=true&w=majority") db = client["SMB\_DB"]

smb = db['smb']

c = smb.find\_one() c.pop("\_id")

cc={"\_smb\_":"the following data is smb"} if out in c:

cc.update(c[out]) else:

x={"smb":"no SMB data on this IP"} cc.update(x)

except:

pass

# SMTP

try:

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMTP\_DB?retryWrites=true&w=majority") db = client["SMTP\_DB"]

smtp = db['smtp']

d = smtp.find\_one() d.pop("\_id")

dd={"\_smtp\_":"the following data is smtp"} if out in d:

dd.update(d[out]) else:

x={"smb":"no SMB data on this IP"} dd.update(x)

except:

pass

# TELNET

try:

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/TELNET\_DB?retryWrites=true&w=majority") db = client["TELNET\_DB"]

telnet = db['telnet'] e = telnet.find\_one() e.pop("\_id")

ee={"\_telnet\_":"the following data is telnet"} if out in e:

ee.update(e[out]) else:

x={"smb":"no TELNET data on this IP"} ee.update(x)

except:

pass z={"FTP":aa,"RDP":bb,"SMB":cc,"SMTP":dd,"TELNET":ee}

#z=str(z) #return str(z) return jsonify(z)

#marker color{'blue', 'lightred', 'darkpurple', 'pink', 'lightgray', 'cadetblue', 'beige', 'lightgreen', 'lightblue', 'black', 'darkred', 'orange', 'darkgreen', 'purple', 'gray', 'red', 'green', 'white', 'darkblue'}.

@app.route('/map',methods = ['POST']) def map\_index():

#folium\_map = folium.Map((0,0), zoom\_start=2)

ftp\_db\_mark() rdp\_db\_mark() smb\_db\_mark() smtp\_db\_mark() telnet\_db\_mark() templates = """

{% macro html(this, kwargs) %}

<!doctype html>

<html lang="en">

<head>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<title>jQuery UI Draggable - Default functionality</title>

<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">

<script src="https://code.jquery.com/jquery-1.12.4.js"></script>

<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>

<script>

$( function() {

$( "#maplegend" ).draggable({

start: function (event, ui) {

$(this).css({

right: "auto",

top: "auto", bottom: "auto"

});

});

}

});

</script>

</head>

<body>

<div id='maplegend' class='maplegend'

style='position: absolute; z-index:9999; border:2px solid grey; background-color:rgba(255, 255, 255, 0.8);

border-radius:6px; padding: 10px; font-size:14px; right: 20px; bottom: 200px;'>

<div class='legend-scale'>

<ul class='legend-labels'>

<li><span style='background:blue;opacity:0.7;'></span>FTP</li>

<li><span style='background:green;opacity:0.7;'></span>RDP</li>

<li><span style='background:#FF7F7F;opacity:0.7;'></span>SMB</li>

<li><span style='background:red;opacity:0.7;'></span>SMTP</li>

<li><span style='background:#e75480;opacity:0.7;'></span>Telnet</li>

</ul>

</div>

</body>

</html>

<style type='text/css'>

.maplegend .legend-title { text-align: left;

margin-bottom: 5px; font-weight: bold; font-size: 90%;

}

.maplegend .legend-scale ul { margin: 0;

margin-bottom: 5px; padding: 0;

float: left;

list-style: none;

}

.maplegend .legend-scale ul li {

font-size: 80%; list-style: none; margin-left: 0; line-height: 18px;

margin-bottom: 2px;

}

.maplegend ul.legend-labels li span { display: block;

float: left; height: 16px; width: 30px;

margin-right: 5px; margin-left: 0;

border: 1px solid #999;

}

.maplegend .legend-source { font-size: 80%;

color: #777; clear: both;

}

.maplegend a { color: #777;

}

</style>

{% endmacro %}"""

macro = MacroElement() macro.\_template = Template(templates) folium\_map.get\_root().add\_child(macro)

return folium\_map.\_repr\_html\_()

def ftp\_db\_mark(): a\_logged\_in=[] a\_connection\_reset=[] a\_lat=[]

a\_long=[] a\_country=[]

a\_city=[] a\_region=[] a\_ip=[] client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/FTP\_DB?retryWrites=true&w=majority") db = client["FTP\_DB"]

ftp = db['ftp']

a = ftp.find\_one() a.pop("\_id")

for key in a: x=a[key] a\_ip.append(key)

a\_logged\_in.append(x["logged\_in"]) a\_connection\_reset.append(x['connection\_reset']) a\_lat.append(x['latitude']) a\_long.append(x['longitude']) a\_region.append(x['region']) a\_country.append(x['country']) a\_city.append(x['city'])

print(len(a\_lat))

for i in range(len(a\_lat)): folium.Marker([str(a\_lat[i]),str(a\_long[i])],

popup="IP: "+str(a\_ip[i])+" city:"+str(a\_city[i])+" region:"+str(a\_region[i])+" country:"+str(a\_country[i])+" logged in:"+str(a\_logged\_in[i])+" connection reset:"+str(a\_connection\_reset[i])).add\_to(folium\_map)

def rdp\_db\_mark(): b\_scan\_result=[] b\_lat=[] b\_long=[] b\_country=[] b\_city=[] b\_region=[] b\_ip=[]

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/RDP\_DB?retryWrites=true&w=majority") db = client["RDP\_DB"]

rdp = db['rdp']

b = rdp.find\_one()

b.pop("\_id") for key in b:

x=b[key] b\_ip.append(key)

b\_scan\_result.append(x["scan\_result"]) b\_lat.append(x['latitude']) b\_long.append(x['longitude']) b\_region.append(x['region']) b\_country.append(x['country']) b\_city.append(x['city'])

for i in range(len(b\_lat)): folium.Marker([str(b\_lat[i]),str(b\_long[i])],popup="IP: "+str(b\_ip[i])+"

city:"+str(b\_city[i])+" region:"+str(b\_region[i])+" country:"+str(b\_country[i])+" scan\_result:"+str(b\_scan\_result[i]),

icon=folium.Icon(color="green")).add\_to(folium\_map)

def smb\_db\_mark(): c\_scan\_result=[] c\_lat=[] c\_long=[] c\_country=[] c\_city=[] c\_region=[] c\_ip=[]

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMB\_DB?retryWrites=true&w=majority") db = client["SMB\_DB"]

smb = db['smb']

c = smb.find\_one() c.pop("\_id")

for key in c: x=c[key] c\_ip.append(key)

c\_scan\_result.append(x["scan\_result"]) c\_lat.append(x['latitude']) c\_long.append(x['longitude']) c\_region.append(x['region']) c\_country.append(x['country']) c\_city.append(x['city'])

for i in range(len(c\_lat)):

folium.Marker([str(c\_lat[i]),str(c\_long[i])],popup="IP: "+str(c\_ip[i])+" city:"+str(c\_city[i])+" region:"+str(c\_region[i])+" country:"+str(c\_country[i])+" scan\_result:"+str(c\_scan\_result[i]),

icon=folium.Icon(color="lightred")).add\_to(folium\_map)

def smtp\_db\_mark(): d\_logged\_in=[] d\_connection\_reset=[] d\_lat=[]

d\_long=[] d\_country=[] d\_city=[] d\_region=[] d\_ip=[] client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMTP\_DB?retryWrites=true&w=majority") db = client["SMTP\_DB"]

smtp= db['smtp']

d = smtp.find\_one() d.pop("\_id")

for key in d: x=d[key] d\_ip.append(key)

d\_logged\_in.append(x["logged\_in"]) d\_connection\_reset.append(x['connection\_reset']) d\_lat.append(x['latitude']) d\_long.append(x['longitude']) d\_region.append(x['region']) d\_country.append(x['country']) d\_city.append(x['city'])

for i in range(len(d\_lat)): folium.Marker([str(d\_lat[i]),str(d\_long[i])],popup="IP: "+str(d\_ip[i])+"

city:"+str(d\_city[i])+" region:"+str(d\_region[i])+" country:"+str(d\_country[i])+" logged\_in:"+str(d\_logged\_in[i])+" connection\_reset:"+str(d\_connection\_reset[i]),

icon=folium.Icon(color="red")).add\_to(folium\_map)

def telnet\_db\_mark():

e\_logged\_in=[] e\_connection\_reset=[] e\_lat=[]

e\_long=[] e\_country=[] e\_city=[] e\_region=[] e\_ip=[] client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/TELNET\_DB?retryWrites=true&w=majority") db = client["TELNET\_DB"]

ftp = db['telnet'] e = ftp.find\_one() e.pop("\_id")

for key in e: x=e[key] e\_ip.append(key)

e\_logged\_in.append(x["logged\_in"]) e\_connection\_reset.append(x['connection\_reset']) e\_lat.append(x['latitude']) e\_long.append(x['longitude']) e\_region.append(x['region']) e\_country.append(x['country']) e\_city.append(x['city'])

for i in range(len(e\_lat)): folium.Marker([str(e\_lat[i]),str(e\_long[i])],

popup="IP: "+str(e\_ip[i])+" city:"+str(e\_city[i])+" region:"+str(e\_region[i])+" country:"+str(e\_country[i])+" logged in:"+str(e\_logged\_in[i])+" connection reset:"+str(e\_connection\_reset[i]),

icon=folium.Icon(color="pink")).add\_to(folium\_map)

@app.route('/') def index():

return render\_template("index.html")

if name == " main ":

folium\_map = folium.Map((0,0), zoom\_start=2) app.run(debug = True)

#### INDEX.HTML

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style> body {

font-family: Arial; color: white;

}

.split {

height: 100%;

width: 50%; position: fixed; z-index: 1;

top: 0;

overflow-x: hidden; padding-top: 20px;

}

.left { left: 0;

background-color: #111;

}

.right { right: 0;

background-color: red;

}

.centered { position: absolute; top: 50%;

left: 50%;

transform: translate(-50%, -50%); text-align: center;

}

.centered img {

width: 150px; border-radius: 50%;

}

</style>

</head>

<body>

<div class="split left">

<div class="centered">

<h2>Search for IP</h2>

<form method="POST" action = "/search">

<input type='submit' id='btnTest1' value='search'>

</form>

</div>

</div>

<div class="split right">

<div class="centered">

<h2>Visualize all the data</h2>

<form method="POST" action = "/map">

<input type='submit' id='btnTest2' value='map'>

</form>

</div>

</div>

</body>

</html>

#### CLIENT-SIDE CODING [FTP.PY](ftp://ftp.py/)

import asyncio

from ftplib import FTP import json

import ftplib import os import time

from ip2geotools.databases.noncommercial import DbIpCity from geopy.geocoders import Nominatim

from pymongo import MongoClient import urllib

import re

b="2M" p=21 n=300

i=0 ip\_data={}

async def scan(full\_ip):

proc = await asyncio.create\_subprocess\_exec( 'zmap','-p','21',full\_ip,'-o',str(full\_ip), stdout=asyncio.subprocess.PIPE, stderr=asyncio.subprocess.PIPE)

print (" ip : {0} scan completed ".format(full\_ip))

async def scan2(full\_ip):

print(" scan "+full\_ip+" ")

if (os.path.isfile("{}".format(full\_ip)) == True):

if (os.stat(full\_ip).st\_size != 0):

geolocator = Nominatim(user\_agent="app") print("Port is open "+full\_ip)

reset="False" logged\_in="False" country="" region=""

city="" latitude="" longitude="" try:

print("in ftp ")

s = ftplib.FTP(full\_ip, "anonymous" , "anonymous",timeout=2) s.quit()

logged\_in="True"

response = DbIpCity.get(full\_ip, api\_key='free') country=response.country region=response.region

city=re.sub(r" ?\([^)]+\)", "", response.city) loc = geolocator.geocode(city).raw

latitude=loc['lat'] longitude=loc['lon']

a={full\_ip:{"logged\_in":logged\_in,"connection\_reset":reset,"latitude":latitude,"longitude":longit ude,"city":city,"region":region,"country":country}}

ip\_data.update(a)

except ftplib.all\_errors as e: print(e)

reset="True"

response = DbIpCity.get(full\_ip, api\_key='free') country=response.country

city=re.sub(r" ?\([^)]+\)", "", response.city) region=response.region

loc = geolocator.geocode(city).raw latitude=loc['lat'] longitude=loc['lon']

a={full\_ip:{"logged\_in":logged\_in,"connection\_reset":reset,"latitude":latitude,"longitude":longit ude,"city":city,"region":region,"country":country}}

ip\_data.update(a) os.remove(str(full\_ip)) else:

os.remove(str(full\_ip))

def loop2(i1,i2,i3): i4=0

try:

while(i4<256 and i3<256 and i2<256): full\_ip=str(i1)+"."+str(i2)+"."+str(i3)+"."+str(i4) cmd="zmap "+"-B "+b+" -p "+str(p)+" "+full\_ip loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan2(full\_ip)) i4=i4+1

except Exception as e: print(e)

def start(): i=0 f2=0 ip1=111

ip2=87 ip3=224

ip4=0 try:

while(ip1<256):

if(ip4<256 and ip3<256 and ip2<256): full\_ip=str(ip1)+"."+str(ip2)+"."+str(ip3)+"."+str(ip4) loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan(full\_ip) )

ip4=ip4+1 elif(ip4>255):

loop2(ip1,ip2,ip3) ip4=0

ip3=ip3+1 elif(ip3>255):

ip3=0 ip2=ip2+1

elif(ip2>255): ip2=0 ip1=ip1+1

except Exception as e: print(e)

if name == ' main ': try:

start()

except KeyboardInterrupt: pass

client = MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/FTP\_DB?retryWrites=true&w=majority") db = client["FTP\_DB"]

ftp = db['ftp']

ftp.insert\_one(ip\_data)

"""

with open ('output.json','w') as f: json.dump(ip\_data,f,indent=4) """

#### RDP.PY

import asyncio import json import ftplib import os import time

from geopy.geocoders import Nominatim

from ip2geotools.databases.noncommercial import DbIpCity from pymongo import MongoClient

import urllib import re

b="2M" p=21 n=300

i=0 ip\_data={}

async def scan(full\_ip):

proc = await asyncio.create\_subprocess\_exec( 'zmap','-p','3389',full\_ip,'-o',str(full\_ip), stdout=asyncio.subprocess.PIPE, stderr=asyncio.subprocess.PIPE)

print (" ip : {0} scan completed ".format(full\_ip))

async def scan2(full\_ip):

print(" scan "+full\_ip+" ")

if (os.path.isfile("{}".format(full\_ip)) == True): if (os.stat(full\_ip).st\_size != 0):

print("Port is open "+full\_ip)

geolocator = Nominatim(user\_agent="app") country=""

region="" city=""

latitude="" longitude=""

response = DbIpCity.get(full\_ip, api\_key='free') country=response.country

city=re.sub(r" ?\([^)]+\)", "", response.city) region=response.region

loc = geolocator.geocode(city).raw latitude=loc['lat'] longitude=loc['lon'] a={full\_ip:{"scan\_result":"port

open","latitude":latitude,"longitude":longitude,"city":city,"region":region,"country":country }} ip\_data.update(a)

os.remove(str(full\_ip)) else:

os.remove(str(full\_ip))

def loop2(i1,i2,i3): i4=0

try:

while(i4<256 and i3<256 and i2<256): full\_ip=str(i1)+"."+str(i2)+"."+str(i3)+"."+str(i4) cmd="zmap "+"-B "+b+" -p "+str(p)+" "+full\_ip loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan2(full\_ip)) i4=i4+1

except Exception as e: print(e)

def start(): i=0 f2=0 ip1=121 ip2=196

ip3=14 ip4=0 try:

while(ip1<256):

if(ip4<256 and ip3<256 and ip2<256): full\_ip=str(ip1)+"."+str(ip2)+"."+str(ip3)+"."+str(ip4) loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(scan(full\_ip) ) ip4=ip4+1

elif(ip4>255): loop2(ip1,ip2,ip3) print("back") ip4=0

ip3=ip3+1 elif(ip3>255):

ip3=0 ip2=ip2+1

elif(ip2>255): ip2=0 ip1=ip1+1

except Exception as e: print(e)

if name == ' main ': try:

start()

except KeyboardInterrupt: client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/RDP\_DB?retryWrites=true&w=majority") db = client["RDP\_DB"]

ftp = db['rdp'] ftp.insert\_one(ip\_data) """

with open ('output.json','w') as f: json.dump(ip\_data,f,indent=4) """

#### SMB.PY

import asyncio

from ip2geotools.databases.noncommercial import DbIpCity import json

import os import time

from pymongo import MongoClient import urllib

from geopy.geocoders import Nominatim import re

b="2M" p=21 n=300

i=0 ip\_data={}

async def scan(full\_ip):

proc = await asyncio.create\_subprocess\_exec( 'zmap','-p','445',full\_ip,'-o',str(full\_ip), stdout=asyncio.subprocess.PIPE, stderr=asyncio.subprocess.PIPE)

print (" ip : {0} scan completed ".format(full\_ip))

async def scan2(full\_ip):

print(" scan "+full\_ip+" ")

if (os.path.isfile("{}".format(full\_ip)) == True): if (os.stat(full\_ip).st\_size != 0):

print("Port is open "+full\_ip) try:

geolocator = Nominatim(user\_agent="app") country=""

region="" city="" latitude="" longitude=""

proc = await asyncio.create\_subprocess\_shell( 'smbclient '+' -L '+full\_ip+' -N', stdout=asyncio.subprocess.PIPE, stderr=asyncio.subprocess.PIPE)

stdout, stderr = await proc.communicate() if stdout:

response = DbIpCity.get(full\_ip, api\_key='free')

country=response.country region=response.region

city=re.sub(r" ?\([^)]+\)", "", response.city) loc = geolocator.geocode(city).raw latitude=loc['lat']

longitude=loc['lon'] a={full\_ip:{"scan\_result":"data

available","latitude":latitude,"longitude":longitude,"city":city,"region":region,"country":country}

}

ip\_data.update(a) else:

response = DbIpCity.get(full\_ip, api\_key='free') region=response.region country=response.country

city=re.sub(r" ?\([^)]+\)", "", response.city) loc = geolocator.geocode(city).raw latitude=loc['lat']

longitude=loc['lon'] a={full\_ip:{"scan\_result":"data not

available","latitude":latitude,"longitude":longitude,"city":city,"region":region,"country":country}

}

ip\_data.update(a)

except Exception as e: print(e)

os.remove(str(full\_ip)) else:

os.remove(str(full\_ip))

def loop2(i1,i2,i3): i4=0

try:

while(i4<256 and i3<256 and i2<256): full\_ip=str(i1)+"."+str(i2)+"."+str(i3)+"."+str(i4) loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan2(full\_ip)) i4=i4+1

except Exception as e: print(e)

def start():

ip1=213 ip2=191 ip3=222

ip4=0 try:

while(ip1<256):

if(ip4<256 and ip3<256 and ip2<256): full\_ip=str(ip1)+"."+str(ip2)+"."+str(ip3)+"."+str(ip4) "smbclient -L "+full\_ip+" -N"

cmd="zmap "+"-B "+b+" -p "+str(p)+" "+full\_ip loop = asyncio.get\_event\_loop()

#await asyncio.wait\_for(initial(cmd,full\_ip),timeout=2) loop.run\_until\_complete(scan(full\_ip) ) #time.sleep(0.3) #loop.run\_until\_complete(scan2(full\_ip))

ip4=ip4+1 elif(ip4>255):

loop2(ip1,ip2,ip3) ip4=0

ip3=ip3+1 elif(ip3>255):

ip3=0 ip2=ip2+1

elif(ip2>255): ip2=0 ip1=ip1+1

except Exception as e: print(e)

if name == ' main ': try:

start()

except KeyboardInterrupt: client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMB\_DB?retryWrites=true&w=majority")

db = client["SMB\_DB"] ftp = db['smb'] ftp.insert\_one(ip\_data) """

with open ('output.json','w') as f: json.dump(ip\_data,f,indent=4) """

#### SMTP.PY

import asyncio import telnetlib import json import os import time

from ip2geotools.databases.noncommercial import DbIpCity from pymongo import MongoClient

import urllib

from geopy.geocoders import Nominatim import re

b="2M" p=25

ip\_data={}

async def scan(full\_ip):

proc = await asyncio.create\_subprocess\_exec( 'zmap','-p','25',full\_ip,'-o',str(full\_ip), stdout=asyncio.subprocess.PIPE, stderr=asyncio.subprocess.PIPE)

print (" ip : {0} scan completed ".format(full\_ip))

async def scan2(full\_ip): logged\_in=False reset=False

print(" scan "+full\_ip+" ")

if (os.path.isfile("{}".format(full\_ip)) == True): if (os.stat(full\_ip).st\_size != 0):

print("Port is open "+full\_ip) try:

geolocator = Nominatim(user\_agent="app") country=""

region="" city="" latitude="" longitude=""

telnet = telnetlib.Telnet() telnet.open(full\_ip, p,timeout=1) telnet.close()

logged\_in=True

response = DbIpCity.get(full\_ip, api\_key='free') country=response.country region=response.region

city=re.sub(r" ?\([^)]+\)", "", response.city) loc = geolocator.geocode(city).raw latitude=loc['lat']

longitude=loc['lon']

a={full\_ip:{"logged\_in":logged\_in,"connection\_reset":reset,"latitude":latitude,"longitude":longit ude,"city":city,"region":region,"country":country}}

ip\_data.update(a)

except Exception as e: print(e)

reset=True

response = DbIpCity.get(full\_ip, api\_key='free') country=response.country region=response.region

city=re.sub(r" ?\([^)]+\)", "", response.city) loc = geolocator.geocode(city).raw latitude=loc['lat']

longitude=loc['lon']

a={full\_ip:{"logged\_in":logged\_in,"connection\_reset":reset,"latitude":latitude,"longitude":longit ude,"city":city,"region":region,"country":country}}

ip\_data.update(a) os.remove(str(full\_ip)) else: os.remove(str(full\_ip))

def loop2(i1,i2,i3): i4=0

try:

while(i4<256 and i3<256 and i2<256): full\_ip=str(i1)+"."+str(i2)+"."+str(i3)+"."+str(i4) cmd="zmap "+"-B "+b+" -p "+str(p)+" "+full\_ip loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan2(full\_ip)) i4=i4+1

except Exception as e: print(e)

def start(): ip1=8 ip2=37 ip3=55 ip4=0 try:

while(ip1<256):

if(ip4<256 and ip3<256 and ip2<256): full\_ip=str(ip1)+"."+str(ip2)+"."+str(ip3)+"."+str(ip4) loop = asyncio.get\_event\_loop() loop.run\_until\_complete(scan(full\_ip) )

ip4=ip4+1 elif(ip4>255):

loop2(ip1,ip2,ip3) ip4=0

ip3=ip3+1 elif(ip3>255):

ip3=0 ip2=ip2+1

elif(ip2>255): ip2=0 ip1=ip1+1

except Exception as e: print(e)

if name == ' main ': try:

start()

except KeyboardInterrupt: client =

MongoClient("mongodb+srv://randw:"+urllib.parse.quote("Cyber@Sec1995")+"@cluster0.uq5r7

.mongodb.net/SMTP\_DB?retryWrites=true&w=majority") db = client["SMTP\_DB"]

ftp = db['smtp'] ftp.insert\_one(ip\_data) """

with open ('output.json','w') as f: json.dump(ip\_data,f,indent=4) """

# CHAPTER 7

**CONCLUSION AND FUTURE ENHANCEMENT**

59

#### CONCLUSION

In our project, We scan the IP address and update the data to the public whether the IP address is vulnerable or not. The IP addresses are scanned and then stored in a database (Mongodb).When the user enters a specific IP address and submits the IP address,the scanned data is retrieved from the database. The compiled data from the database can be viewed in the map and it shows whether the IP is vulnerable or not.At last it displays the message that the scan is completed.

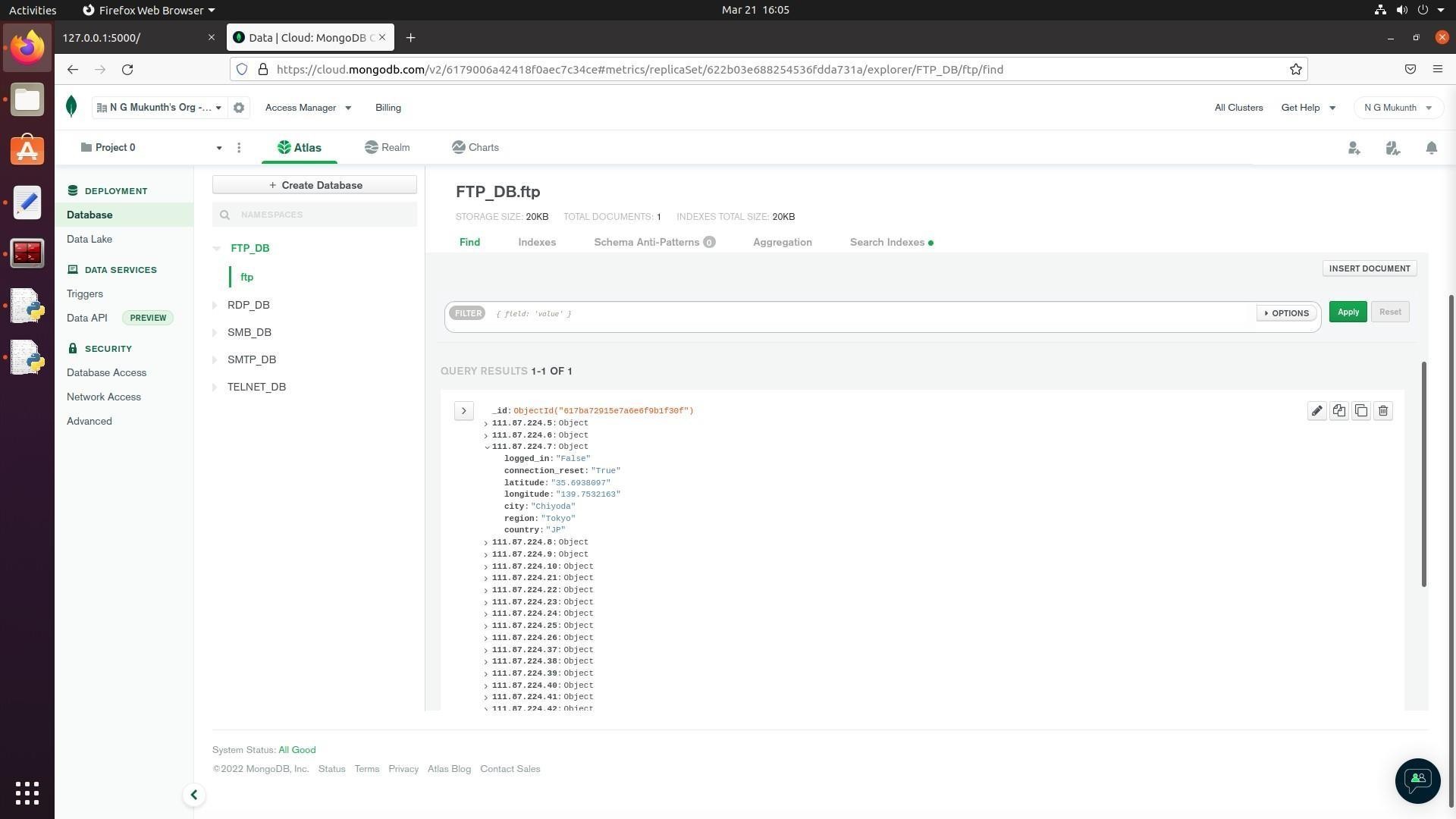
#### FUTURE ENHANCEMET

The future work of our application is to develop for more vulnerability scanning and improved scanning techniques.

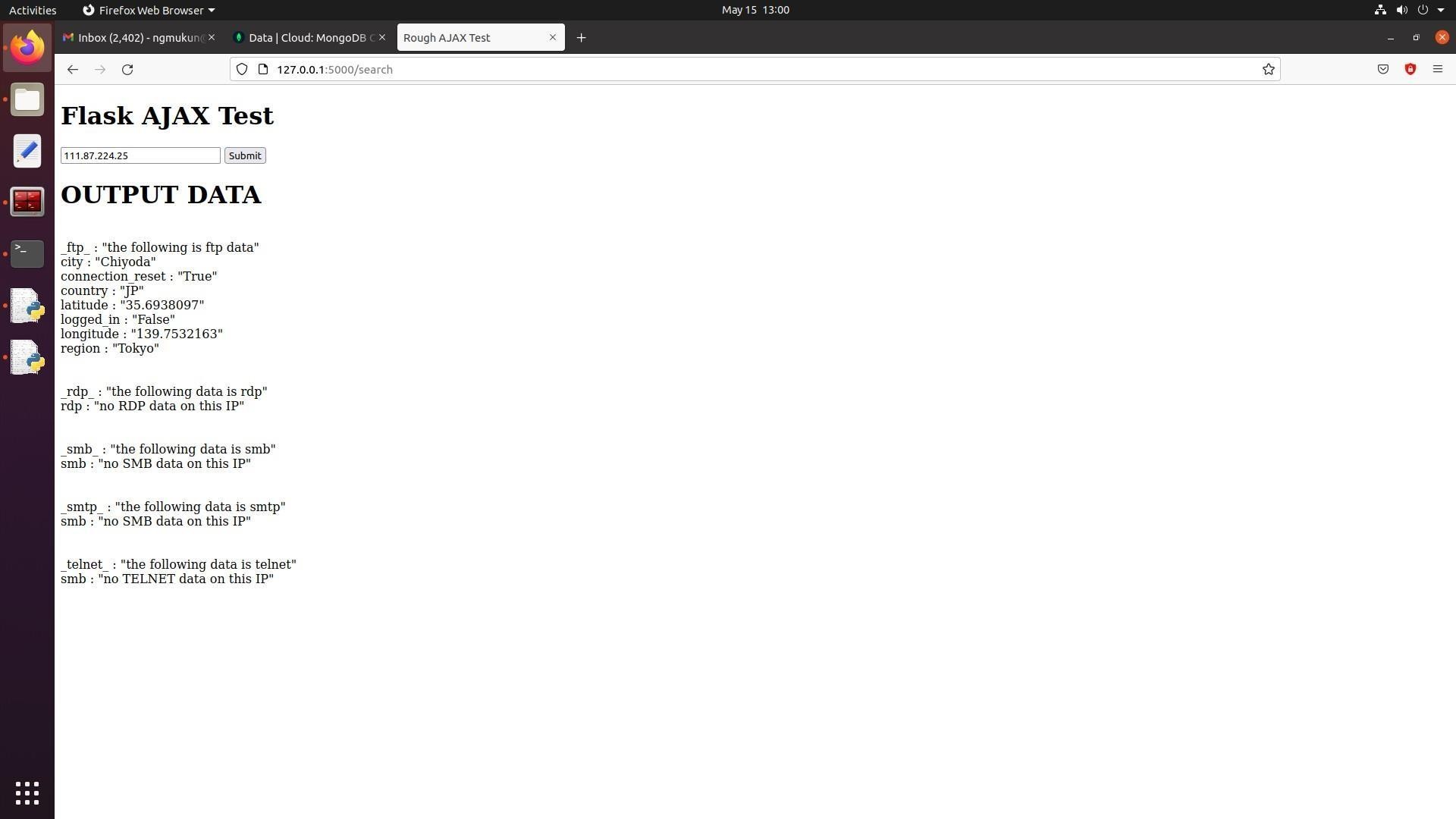
60

#### APPENDICES

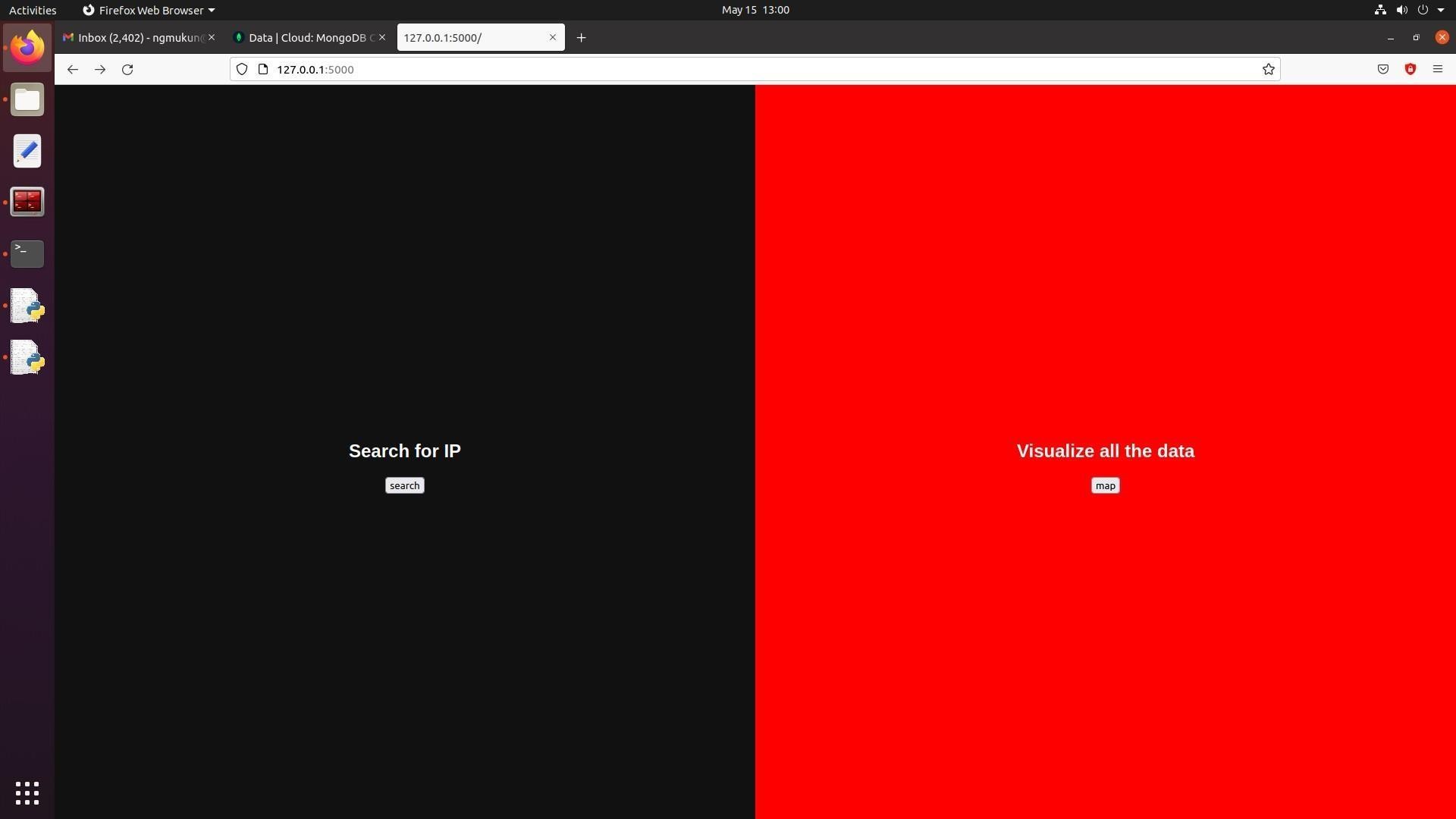
* 1. **SAMPLE SCREENS**



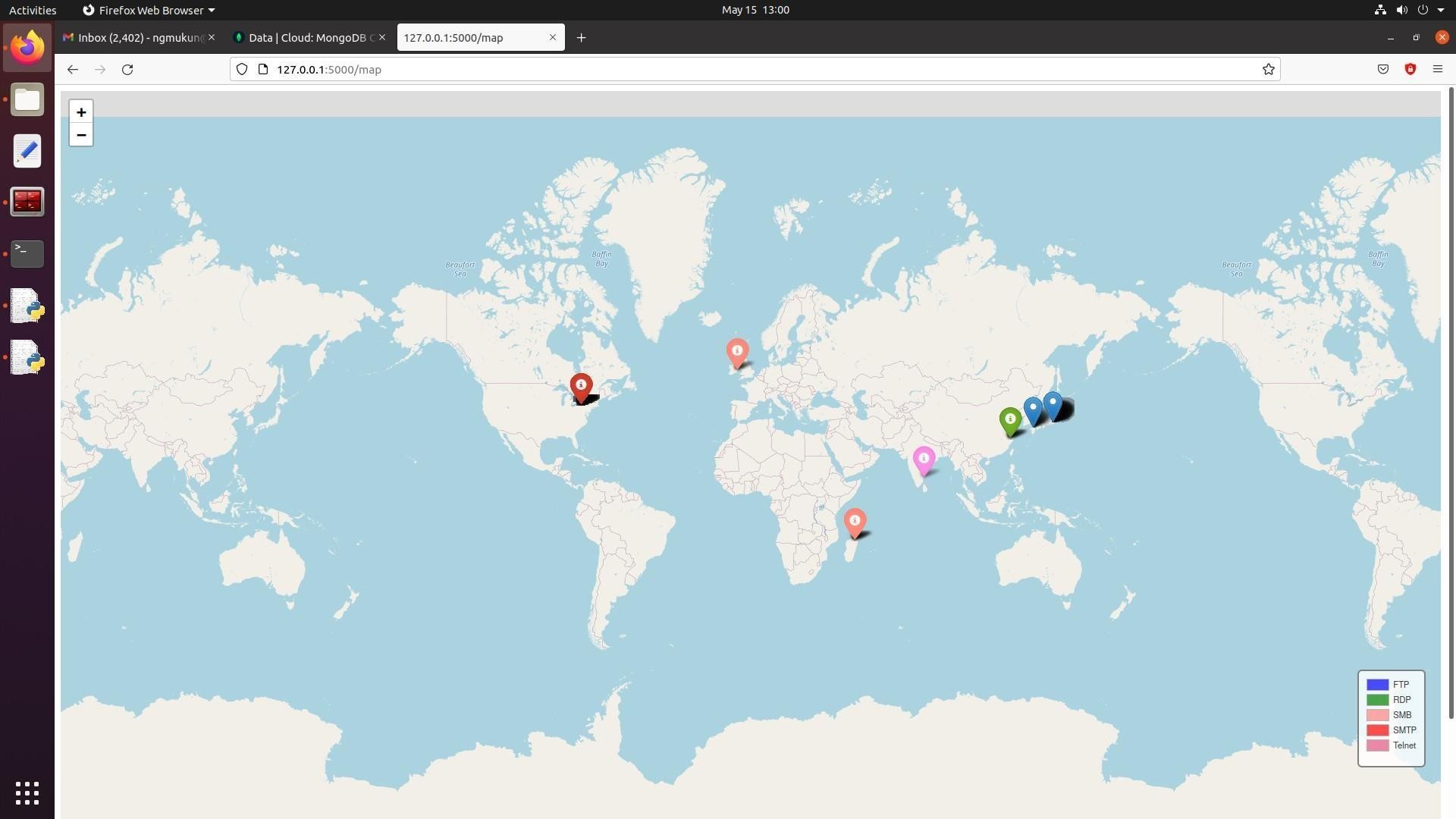
61



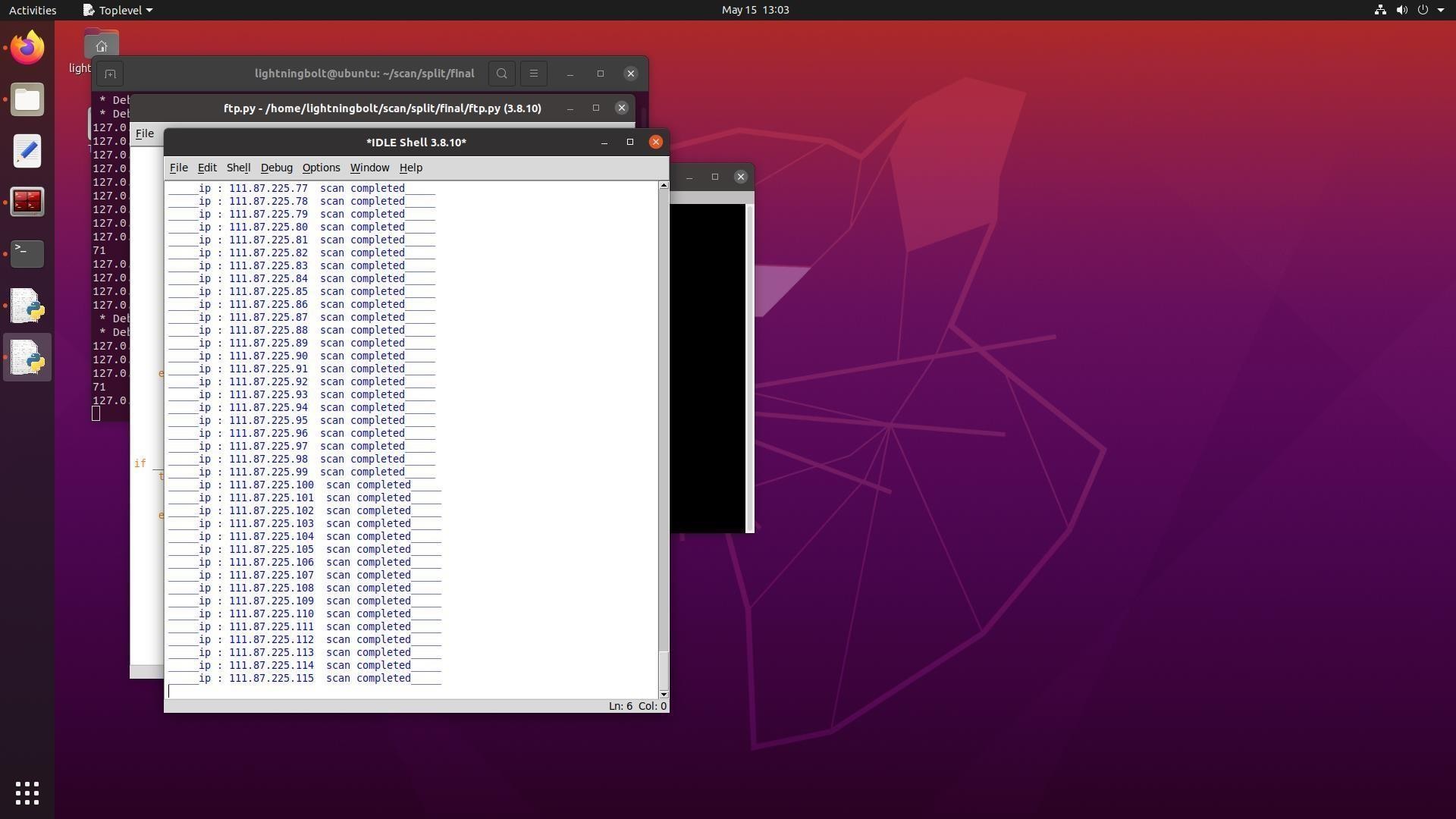
62



63



64



65

**REFERENCES:**

1. Indian Computer Emergency Response Team (CERT-In), “Annual Report-2017”, Ministry of Electronics & Information Technology, Government of India, 2018.

2. Z. Li, et Al., “VulPecker: an automated vulnerability detection system based on code similarity analysis”, ACM, Proc. of the 32 Annual Conference on Computer Security Applications, pp. 201- 213, 2016.

3. S. O. Uwagbole, W. J. Buchanan, & L. Fan, “Applied machine learning predictive analytics to SQL injection attack detection and prevention”, IEEE, Symposium on Integrated Network and Service Management (IM), 2017 IFIP/IEEE, pp. 1087-1090, 2017.

4. I. Medeiros, N. Neves, & M. Correia, “Detecting and removing application vulnerabilities with static analysis and data mining”, IEEE, IEEE Transactions on Reliability, Vol 65, Issue 1, pp. 54-69, 2016.

5. S. S. Alqahtani, E. E. Eghan, & J. Rilling, “SV-AF—A Security Vulnerability Analysis Framework”, IEEE, 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 219- 229, 2016.

6. R. V. Bhor, & H. K. Khanuja, “Analysis of security mechanism and Attack Detection using Vulnerability injection technique”, IEEE, International Conference on Computing Communication Control and automation (ICCUBEA), pp. 1-6, 2016.

7. M. Zachara, “Identification of possible attack attempts against utilizing collective assessment of suspicious requests”, Springer, In Transactions on Computational Collective Intelligence XXII, Berlin, Heidelberg, pp. 45-59, 2016.

8. A. Razzaq, et. Al., “Semantic security against web application attacks”, Information Sciences, Vol. 254, pp. 19-38, 2014.

66