A Major Project Report on

BAT: Deep Learning Methods for IDS in Wireless Networks

Submitted in partial fulfillment for the award of the degree of Bachelor of Technology in the Department of Information Technology

by

Jureddy Akanksha **18321A1204**

Gubba Gayathri **18321A1221**

Manogna Tummanepally **18321A1248**

Under the esteemed guidance of

Internal Guide

Maya B Dhone

Assistant Professor, Information Technology

Bhoj Reddy Engineering College for Women

Department of Information Technology

(Sponsored by Sangam Laxmibai Vidyapeet, approved by AICTE & affiliated to JNTUH)

Vinaynagar, IS Sadan Crossroads, Saidabad, Hyderabad-500 059, Telangana

Ph: +91-40-2453-7282, Website: www.brecw.ac.in, Email: principal@brecw.ac.in

2021-2022

ACKNOWLEDGMENT

It is our pleasure to express our whole-hearted thanks to our internal guide **Mrs Maya B Dhone**, **Assistant Professor, Information Technology Department**, for her extreme guidance and support in completing this project successfully.

We are also thankful to our project coordinator **Mrs M Sandhya Rani, Associate Professor**, **Information Technology Department**, for giving us the opportunity to do this project.

We are also thankful to **Dr C Murugamani**, **Head of the Department**, for providing the necessary facilities to undergo this project.

We would also like to thank **Dr J Madhavan, Principal of Bhoj Reddy Engineering College for Women** for encouragement in carrying out the major project successfully.

We express our deep sense of gratitude to our external project guide, for having permitted us to carry out the project work in their institute and for his/her valuable guidance and supervision at every stage.

We are also thankful to the staff members of the Information Technology department, my friends, and our parents who helped us in completing this project successfully.

By

Jureddy Akanksha **18321A1204**

Gubba Gayathri **18321A1221**

Manogna Tummanepally **18321A1248**

Contents

List of Figures

List of Tables

Abbreviations

- BLSTM : Bidirectional Long Short Term Memory
- NSL-KDD : NSL- Knowledge Discovery Dataset
- BAT : Bidirectional Attention Layer
- MCL : Multi-Convolutional Layer
- RNN : Recurrent Neural Network
- SVM : Support Vector Machine
- TP : True Positive
- FP : False Positive
- TPR : True Positive Rate
- FPR : False Positive Rate

ABSTRACT

Intrusion detection can identify unknown attacks from network traffics and has been an effective means of network security. Nowadays, existing methods for network anomaly detection are usually based on traditional machine learning models, such as KNN, SVM, etc. Although these methods can obtain some outstanding features, they get a relatively low accuracy and rely heavily on the manual design of traffic features, which has been obsolete in the age of big data. To solve the problems of low accuracy and feature engineering in intrusion detection, a traffic anomaly detection model BAT is proposed. The BAT model combines BLSTM (Bidirectional Long Short-term memory) and attention mechanism. The attention mechanism is used to screen the network flow vector composed of packet vectors generated by the BLSTM model, which can obtain the key features for network traffic classification. In addition, we adopt multiple convolutional layers to capture the local features of traffic data. As multiple convolutional layers are used to process data samples, we refer BAT model as BAT-MC. The softmax classifier is used for network traffic classification. The proposed end-to-end model does not use any feature engineering skills and can automatically learn the key features of the hierarchy. It can well describe the network traffic behavior and improve the ability of anomaly detection effectively. We test our model on a public benchmark dataset, and the experimental results demonstrate our model has better performance than other comparison methods.

Keywords: Network traffic, intrusion detection, deep learning, BLSTM, attention mechanism.

1. INTRODUCTION

INTRODUCTION

With the development and improvement of Internet technology, the Internet is providing various convenient services for people. However, we are also facing various security threats. Network viruses, eavesdropping, and malicious attacks are on the rise, causing network security to become the focus of attention of society and government departments. Fortunately, these problems can be well solved via intrusion detection. Intrusion detection plays an important part in ensuring network information security. However, with the explosive growth of Internet business, traffic types in the network are increasing day by day, and network behavior characteristics are becoming increasingly complex, which brings great challenges to intrusion detection. How to identify various malicious network traffics, especially unexpected malicious network traffics, is a key problem that cannot be avoided. The associate editor coordinating the review of this manuscript and approving it for publication was Fan Zhang. In fact, network traffic can be divided into two categories (normal traffics and malicious traffics). Furthermore, network traffic can also be divided into five categories: Normal, DoS (Denial of Service attacks), R2L (Root to Local attacks), U2R (User to Root attack), and Probe (Probing attacks). Hence, intrusion detection can be considered a classification problem. By improving the performance of classifiers in effectively identifying malicious traffics, intrusion detection accuracy can be largely improved. Machine learning methods have been widely used in intrusion detection to identify malicious traffic. However, these methods belong to shallow learning and often emphasize feature engineering and selection. They have difficulty in feature selection and cannot effectively solve the massive intrusion data classification problem, which leads to low recognition accuracy and a high false alarm rate. In recent years, intrusion detection methods based on deep learning have been proposed successively. The authors propose a malware traffic classification method based on convolutional VOLUME 8, 2020 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/ 29575 T. Su et al.: BAT: Deep Learning Methods on Network Intrusion Detection Using NSL-KDD Dataset neural network with traffic data as an image. This method does not need manual design features and directly takes the original traffic as the input data to the classifier. The authors provide an analysis of the viability of Recurrent Neural Networks (RNN) to detect the behavior of network traffic by modeling it as a sequence of states that change over time. The authors verify the performance of the Long Short-Term Memory

(LSTM) network in classifying intrusion traffics. Experimental results show that LSTM can learn all the attack classes hidden in the training data. All the above methods treat the entire network traffic as a whole consisting of a sequence of traffic bytes. They don't make full use of domain knowledge of network traffic. For example, CNN converts continuous network traffic into images for processing, which is equivalent to treating traffic as independent and ignoring the internal relations of network traffic. Firstly, network traffic is a hierarchical structure. Specifically, network traffic is a traffic unit composed of multiple data packets. A Data packet is a traffic unit composed of multiple bytes. Secondly, traffic features in the same and different packets are significantly different. Sequential features between different packets need to be extracted independently. In other words, not all traffic features are equally important for traffic classification in the process of extracting features on certain network traffic.

1.1 Purpose:

The intrusion detection technology can be divided into three major categories: pattern matching methods, traditional machine learning methods, and deep learning methods. In the beginning, people mainly use pattern matching algorithms for intrusion detection. Pattern matching algorithm [14], [15] is the core algorithm of an intrusion detection system based on feature matching. Most algorithms have been considered for use in the past. In [16], the authors make a summary of pattern matching algorithms in Intrusion Detection Systems KMP algorithm, BM algorithm, BMH algorithm, BMHS algorithm, AC algorithm and AC-BM algorithm. Experiments show that the improved algorithm can accelerate the matching speed and has a good time performance. In [17], Naive approach, Knuth-MorrisPratt algorithm and RabinKarp Algorithm are compared to check which of them is most efficient in pattern/intrusion detection. Pcap files have been used as datasets in order to determine the efficiency of the algorithm by taking into consideration their running times respectively.

1.2 Existing System

An Intrusion Detection System (IDS) is a mechanism that detects intrusions or attacks against a system or a network by analyzing the activity of the network and the system. Such intruders can be internal or external. Internal intruders are users inside the network that attempt to raise their access privileges to misuse non-authorized privileges while external intruders are users outside the target network attempting to gain

unauthorized access to the network. The IDS monitors the operations of a host or a network, alerting the system administrator when it detects a security violation.

1.2.1 Problems in Existing System

- We review the technologies involved in the Intrusion Detection System to handle security issues in IoT environments.

1.3 Proposed System

The BAT-MC model consists of five components, including the input layer, multiple convolutional Layers, BSLTM layer, attention layer, and output layer, from bottom to top. At the input layer, the BAT-MC model converts each traffic byte into a one-hot data format. Each traffic byte is encoded as an n-dimensional vector. After the traffic byte is converted into a numerical form, we perform normalization operations. At the multiple convolutional layers, we convert the numerical data into traffic images. Convolutional operation is used as a feature extractor that takes an image representation of a data packet. At the BLSTM layer, the BLSTM model which connects the forward LSTM, and the backward LSTM is used to extract features on the traffic bytes of each packet. BLSTM model can learn the sequential characteristics within the traffic bytes because BLSTM is suitable for the structure of network traffic. In the attention layer, the attention mechanism is used to analyze the important degree of packet vectors to obtain fine-grained features which are more salient for malicious traffic detection. At the output layer, the features generated by the attention mechanism are then imported into a fully connected layer for feature fusion, which obtains the key features that accurately characterize network traffic behavior. Finally, the fused features are fed into a classifier to get the final recognition results.

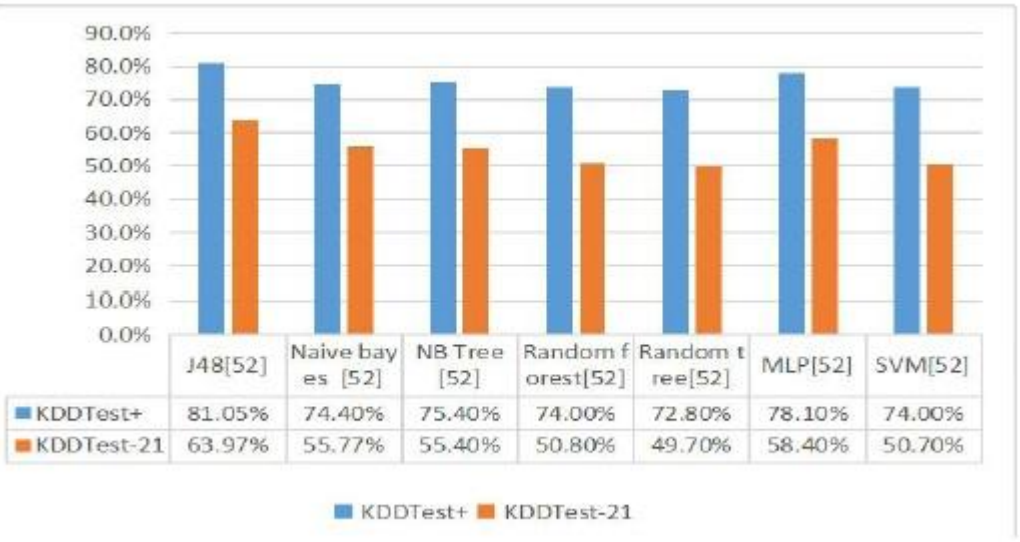


Figure 1.3.1: Performance of BAT-MC model and other Machine Learning models

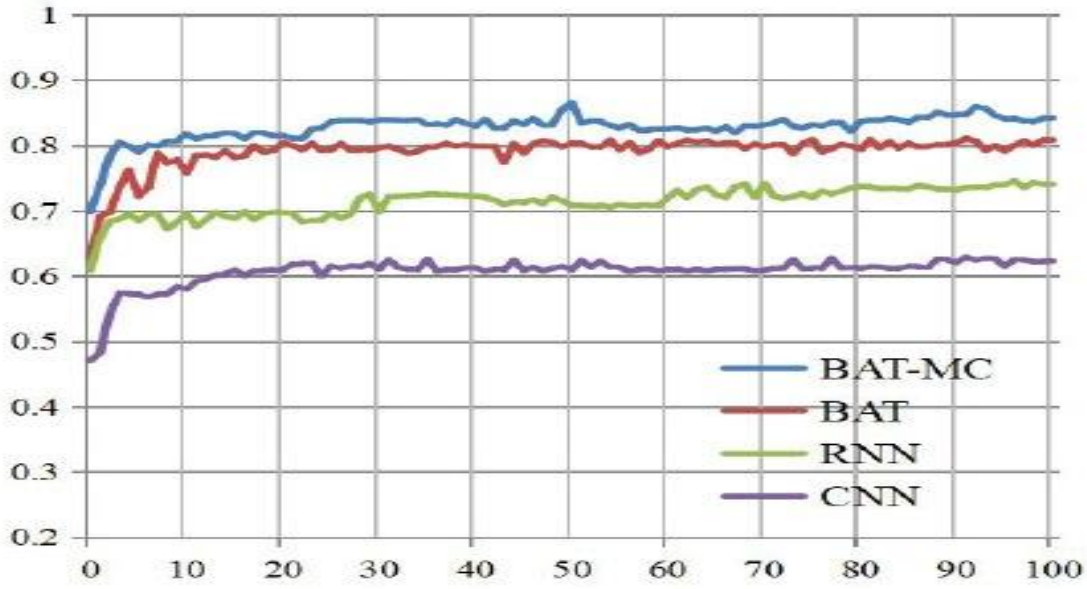


Figure 1.3.2: Comparison of Accuracy with different models

1.3.1 Advantages of Proposed System:

- We propose an end-to-end deep learning model BAT-MC that is composed of BLSTM and attention mechanisms. BAT-MC can well solve the problem of intrusion detection and provide a new research method for intrusion detection
- We introduce the attention mechanism into the BLSTM model to highlight the key input. The attention mechanism conducts feature learning on sequential data

composed of data package vectors. The obtained feature information is reasonable and accurate.

- We compare the performance of BAT-MC with traditional deep learning methods, the BAT-MC model can extract information from each packet. By making full use of the structure information of network traffic, the BAT-MC model can capture features more comprehensively.
- We evaluate our proposed network with a real NSL-KDD dataset. The experimental results show that the performance of BAT-MC is better than the traditional methods.

2. LITERATURE SURVEY

LITERATURE SURVEY

This paper addresses the Internet of Things. The main enabling factor of this promising paradigm is the integration of several technologies and communications solutions. Identification and tracking technologies, wired and wireless sensor and actuator networks, enhanced communication protocols (shared with the Next Generation Internet), and distributed intelligence for smart objects are just the most relevant. As one can easily imagine, any serious contribution to the advance of the Internet of Things must necessarily be the result of synergetic activities conducted in different fields of knowledge, such as telecommunications, informatics, electronics, and social science. In such a complex scenario, this survey is directed to those who want to approach this complex discipline and contribute 30 to its development. Different visions of this Internet of Things paradigm are reported 31 and enabling technologies reviewed. What emerges is that still major issues shall be faced 32 by the research community. The most relevant among them are addressed in detail. The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, can interact with each other and cooperate with their neighbors to reach common goals. Unquestionably, the main strength of the IoT idea is the 48 high impact it will have on several aspects of everyday life and the behavior of potential users. From the point of view of a private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. In this context, demotics, assisted living, e-health, and enhanced learning are only a few examples of possible appli54 cation scenarios in which the new paradigm will play a leading role in the near future. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, and intelligent transportation of people and goods. By starting from the considerations above, it should not be surprising that IoT is included by the US National Intelligence Council in the list of six ‘‘Disruptive Civil Technologies” with potential impacts on US national power. NIC foresees that ‘‘by 2025 Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more”. It highlights future opportunities that will arise, starting from the idea that ‘‘popular demand

combined with technology advances could drive wide- 68 spread diffusion of an Internet of Things (IoT) that could, like the present Internet, contribute invaluably to economic development". The possible threats deriving from widespread adoption of such a technology are also stressed. Indeed, it is emphasized that ''to the extent that everyday objects become information security risks, the IoT could distribute those risks far more widely than the Internet has to date". Many challenging issues still need to be addressed and both technological, as well as social knots, have to be untied before the IoT idea is widely accepted. Central issues are making full interoperability of inter81 connected devices possible, providing them with an always higher degree of smartness by enabling their adaptation and autonomous behavior, while guaranteeing trust, privacy, and security. Also, the IoT idea poses several new problems concerning the networking aspects. In fact, the things composing the IoT will be characterized by low resources in terms of both computation and energy capacity. Accordingly, the proposed solutions need to pay special attention to resource efficiency besides the obvious scalability problems. Several industrial, standardization and research bodies are currently involved in the activity of developing solutions to fulfill the highlighted technological requirements. This survey gives a picture of the current state of the art on the IoT. More specifically, it provides the readers with a description of the different visions of the Internet of Things paradigm coming from different scientific communities; reviews the enabling technologies and illustrates which are the major benefits of the spread of this paradigm in 101 everyday lives; offers an analysis of the major research issues the scientific community still has to face. The main objective is to give the reader the opportunity of understanding what has been done (protocols, algorithms, proposed solutions) and what still remains to be addressed, as well as what are the enabling factors of this evolutionary process and what are its weaknesses and risk factors. The remainder of the paper is organized as follows. In Section 2, we introduce and compare the different visions of the IoT paradigm, which are available from the literature. The IoT main enabling technologies are the subject of Section 3, while the description of the principal applications, which in the future will benefit from the full deployment of the IoT idea, are addressed in Section 4. Section 5 gives a glance at the open issues on which research should focus more, by stressing topics such as addressing, networking, security, privacy, and standardization efforts. Conclusions and future research hints are given in Section 6. One paradigm, many visions Manifold definitions of Internet of Things traceable within the research community testify to the strong interest in the IoT

issue and to the vivacity of the debates on it. By browsing the literature, an interested reader might experience real difficulty in understanding what IoT really means, which basic ideas stand behind this concept, and which social, economic, and technical implications the full deployment of IoT will have. The reason for today's apparent fuzziness around this term is a consequence of the name ''Internet of Things" itself, which syntactically is composed of two terms. The first one pushes towards a network-oriented vision of IoT, while the second one moves the focus on generic ''objects" to be integrated into a common framework. Differences, sometimes substantial, in the IoT visions, raise from the fact that stakeholders, business alliances, research and standardization bodies start approaching the issue from either an Internet-oriented" or a ''Things oriented" perspective, depending on their specific interests, finalities, and backgrounds. It shall not be forgotten, anyway, that the words ''Internet" and ''Things", when put together, assume a meaning which introduces a disruptive level of innovation into today's ICT world. In fact, ''Internet of Things" semantically means ''worldwide network of interconnected objects uniquely addressable, based on standard communication protocols". This implies a huge number of (heterogeneous) objects involved in the process. The object's unique addressing and the representation and storing of the exchanged information become the most challenging issues, bringing directly to a third, ''Semantic oriented", the perspective of IoT. In Fig. 1, the main concepts, technologies, and standards are highlighted and classified with reference to the IoT vision/s they contribute to characterize best. From such an illustration, it clearly appears that the IoT paradigm shall be the result of the convergence of the three main visions addressed above. The very first definition of IoT derives from a ''Things oriented" perspective; the considered things were very simple items: Radio-Frequency Identification (RFID) tags. The terms ''Internet of Things" is, in fact, attributed to The Auto-ID Labs, a worldwide network of academic research laboratories in the field of networked RFID and emerging sensing technologies. These institutions, since their establishment, have been targeted to architect the IoT, together with EPC global. Their focus has primarily been on the development of the Electronic Product Code™ (EPC) to support the spread use of RFID in worldwide modern trading networks and to create the industry-driven global standards for the EPC global Network™. These standards are mainly designed to improve object visibility (i.e., the traceability of an object and the awareness of its status, current location, etc.). This is undoubtedly a key component of the path to the full deployment of the IoT vision, but it is not the only one. In a broader sense,

IoT cannot be just a global EPC system in which the only objects are RFIDs; they are just a part of the full story! And the same holds for the alternative Unique/Universal/Ubiquitous Identifier (UID) architecture, whose main idea is still the development of (middleware-based) solutions for global visibility of objects in an IoT vision. It is the authors' opinion that starting from RFID-centric solutions may be positive as the main aspects stressed by RFID technology, namely item traceability, and addressability shall be addressed also by the IoT. Notwithstanding, alternative and somehow more complete, IoT visions recognize that the term IoT implies a much wider vision than the idea of a mere object's identification

> Consider writing, perhaps the first information technology: The ability to capture a symbolic representation of spoken language for long-term storage freed information from the limits of individual memory. Today this technology is ubiquitous in industrialized countries. Not only do books, magazines, and newspapers convey written information, but so do street signs, billboards, shop signs, and even graffiti. Candy wrappers are covered in writing. The constant background presence of these products of "literacy technology" does not require active attention, but the information to be conveyed is ready for use at a glance. It is difficult to imagine modern life otherwise. Silicon-based information technology, in contrast, is far from having become part of the environment. More than 50 million personal computers have been sold, and nonetheless, the computer remains largely in a world of its own. It is approachable only through complex jargon that has nothing to do with the tasks for which people actually use computers. The state of the art is perhaps analogous to the period when scribes had to know as much about making ink or baking clay as they did about writing. The arcane aura that surrounds personal computers is not just a "user interface" problem. My colleagues and I at PARC think that the idea of a "personal" computer itself is misplaced and that the vision of laptop machines, Dynabook, and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computing an integral, invisible part of the way people live their lives. Therefore, we are trying to conceive a new way of thinking about computers in the world, one that considers the natural human environment and allows the computers themselves to vanish into the background. Such a disappearance is a fundamental consequence not of technology, but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading. Computer scientist, economist, and

Nobelist Herb Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist TK Gibson calls it "visual invariants"; philosophers Georg Gadamer and Martin Heidegger call it "the horizon" and the "ready-to-hand", John Seely Brown at PARC calls it the "periphery". All say, in essence, that only when things disappear in this way are, we freed to use them without thinking and so to focus beyond them on new goals. The idea of integrating computers seamlessly into the world at large runs counter to a number of present-day trends. "Ubiquitous computing" in this context does not just mean computers that can be carried to the beach, jungle, or airport. Even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box. By analogy to writing, carrying a super-laptop is like owning just one very important book. Customizing this book, even writing millions of other books, does not begin to capture the real power of literacy. Furthermore, although ubiquitous computers may employ sound and video in addition to text and graphics, that does not make them "multimedia computers." Today's multimedia machine makes the computer screen a demanding focus of attention rather than allowing it to fade into the background. Perhaps most diametrically opposed to our vision is the notion of "virtual reality," which attempts to make a world inside the computer. Users don special goggles that project an artificial scene on their eyes; they wear gloves or even body suits that sense their motions and gestures so that they can move about and manipulate virtual objects. Although it may have its purpose in allowing people to explore realms otherwise inaccessible -- the insides of cells, the surfaces of distant planets, the information web of complex databases – the virtual reality is only a map, not a territory. It excludes desks, offices, other people not wearing goggles and body suits, weather, grass, trees, walks, chance encounters, and in general the infinite richness of the universe. Virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists. Indeed, the opposition between the notion of virtual reality and ubiquitous, invisible computing is so strong that some of us use the term "embodied virtuality" to refer to the process of drawing computers out of their electronic shells. The "virtuality" of computer-readable data -- all the different ways in which it can be altered, processed, and analyzed -- is brought into the physical world. How do technologies disappear into the background? The vanishing of electric motors may serve as an instructive example: At the turn of the century, a typical workshop or factory contained a single-engine that drove dozens or hundreds of different machines

through a system of shafts and pulleys. Cheap, small, efficient electric motors made it possible first to give each machine or tool its own source of the motive force, then to put many motors into a single machine. A glance through the shop manual of a typical automobile, for example, reveals twenty-two motors and twenty-five more solenoids. They start the engine, clean the windshield, lock and unlock the doors, and so on. By paying careful attention it might be possible to know whenever one activated a motor, but there would be no point to it. Most of the computers that participate in embodied virtuality will be invisible in fact as well as in metaphor. Already computers in light switches, thermostats, stereos, and ovens help to activate the world. These machines and more will be interconnected in a ubiquitous network. As computer scientists, however, my colleagues and I have focused on devices that transmit and display information more directly. We have found two issues of crucial importance: location and scale. Little is more basic to human perception than physical juxtaposition and so ubiquitous computers must know where they are. (Today's computers, in contrast, have no idea of their location and surroundings.) If a computer merely knows what room it is in, it can adapt its behavior in significant ways without requiring even a hint of artificial intelligence. Ubiquitous computers will also come in different sizes, each suited to a particular task. My colleagues and I have built what we call tabs, pads, and boards: inch-scale machines that approximate active Post-It notes, foot-scale ones that behave something like a sheet of paper (or a book or a magazine), and yard-scale displays that are the equivalent of a blackboard or bulletin board. How many tabs, pads, and board-sized writing and display surfaces are there in a typical room? Look around you: at the inch, the scale includes wall notes, titles on book spines, labels on controls, thermostats, and clocks, as well as small pieces of paper. Depending upon the room you may see more than a hundred tabs, ten or twenty pads, and one or two boards. This leads to our goals for initially deploying the hardware of embodied virtuality: hundreds of computers per room. Hundreds of computers in a room could seem intimidating at first, just as hundreds of volts coursing through wires in the walls did at one time. But like the wires in the walls, these hundreds of computers will come to be invisible to common awareness. People will simply use them unconsciously to accomplish everyday tasks. Tabs are the smallest components of embodied virtuality. Because they are interconnected, tabs will expand on the usefulness of existing inch-scale computers such as the pocket calculator and the pocket organizer. Tabs will also take on functions that no computer performs today. For example, Olivetti Cambridge Research

Labs pioneered active badges, and now computer scientists at PARC and other research laboratories around the world are working with these clip-on computers roughly the size of an employee ID card. These badges can identify themselves to receivers placed throughout a building, thus making it possible to keep track of the people or objects to which they are attached. In our experimental embodied virtuality, doors open only to the right badge wearer, rooms greet people by name, telephone calls can be automatically forwarded to wherever the recipient may be, receptionists know where people are, and computer terminals retrieve the preferences of whoever is sitting at them, and appointment diaries write themselves. No revolution in artificial intelligence is needed--just the proper embedding of computers into the everyday world. The automatic diary shows how such a simple thing as knowing where people are can yield complex dividends: meetings, for example, consist of several people spending time in the same room, and the subject of a meeting is most likely the files called up on that room's display screen while the people are there. My colleague Roy Want has designed a tab incorporating a small display that can serve simultaneously as an active badge, calendar, and diary. It will also act as an extension to computer screens: instead of shrinking a program window down to a small icon on the screen, for example, a user will be able to shrink the window onto a tab display. This will leave the screen free for information and let people arrange their computer-based projects in the area around their terminals, much as they now arrange paper-based projects in piles on desks and tables. Carrying a project to a different office for discussion is as simple as gathering up its tabs; the associated programs and files can be called up on any terminal. The next step up in size is the pad, something of a cross between a sheet of paper and current laptop and palmtop computers. Bob Krivacic at PARC has built a prototype pad that uses two microprocessors, a workstation-sized display, a multi-button stylus, and a radio network that can potentially handle hundreds of devices per person per room. Pads differ from conventional portable computers in one crucial way. Whereas portable computers go everywhere with their owners, the pad that must be carried from place to place is a failure. Pads are intended to be "scrap computers" (analogous to scrap paper) that can be grabbed and used anywhere; they have no individualized identity or importance. One way to think of pads is as an antidote to windows. Windows were invented at PARC and popularized by Apple in the Macintosh as a way of fitting several different activities onto the small space of a computer screen at the same time. In twenty years, computer screens have not grown

much larger. Computer window systems are often said to be based on the desktop metaphor--but who would ever use a desk whose surface area is only 9" by 11"? Pads, in contrast, use a real desk. Spread many electronic pads around on the desk, just as you spread out papers. Have many tasks in front of you and use the pads as reminders. Go beyond the desk to drawers, shelves, and coffee tables. Spread the many parts of the many tasks of the day out in front of you to fit both the task and the reach of your arms and eyes, rather than to fit the limitations of CRT glassblowing. Someday pads may even be as small and light as actual paper, but meanwhile, they can fulfill many more of paper's functions than can computer screens. Yard-size displays (boards) serve several purposes: in the home, video screens and bulletin boards; in the office, bulletin boards, whiteboards, or flip charts. A board might also serve as an electronic bookcase from which one might download texts to a pad or tab. For the time being, however, the ability to pull out a book and place it comfortably on one's lap remains one of the many attractions of paper. Similar objections apply to using a board as a desktop; people will have to get used to using pads and tabs on a desk as an adjunct to computer screens before taking embodied virtuality even further. Boards built by Richard Bruce and Scott Elrod at PARC currently measure about 40 by 60 inches and display 1024x768 black-and-white pixels. To manipulate the display, users pick up a piece of wireless electronic "chalk" that can work either in contact with the surface or from a distance. Some researchers, using themselves and their colleagues as guinea pigs, can hold electronically mediated meetings or engage in other forms of collaboration around a liveaboard. Others use the boards as testbeds for improved display hardware, new "chalk" and interactive software. For both obvious and subtle reasons, the software that animates a large, shared display and its electronic chalk is not the same as that for a workstation. Switching back and forth between chalk and keyboard may involve walking several steps, and so the act is qualitatively different from using a keyboard and mouse. In addition, the body size is an issue -- not everyone can reach the top of the board, so a Macintosh-style menu bar may not be a good idea. We have built enough liveaboards to permit casual use: they have been placed in ordinary conference rooms and open areas, and no one needs to sign up or give advance notice before using them. By building and using these boards, researchers start to experience and so understand a world in which computer interaction casually enhances every room. Liveaboards can usefully be shared across rooms as well as within them. In experiments instigated by Paul Dourish of EuroPARC and Sara Bly and Frank Halasz of PARC,

groups at widely separated sites gathered around boards -- each displaying the same image -- and jointly composed pictures and drawings. They have even shared two boards across the Atlantic. Liveaboards can also be used as bulletin boards. There is already too much data for people to read and comprehend of it, and so Marvin Theimer and David Nichols at PARC have built a prototype system that attunes its public information to the people reading it. Their "scoreboard" requires little or no interaction from the user other than to look and to wear an active badge. Prototype tabs, pads, and boards are just the beginning of ubiquitous computing. The real power of the concept comes not from any one of these devices; it emerges from the interaction of all of them. The hundreds of processors and displays are not a "user interface" like a mouse and windows, just a pleasant and effective "place" to get things done. What will be most pleasant and effective is that tabs can animate objects previously inert. They can beep to help locate mislaid papers, books, or other items. File drawers can open and show the desired folder -- no searching. Tabs in library catalogs can make active maps to any book and guide searchers to it, even if it is off the shelf and on a table from the last reader. In presentations, the size of text on overhead slides, the volume of the amplified voice, and even the amount of ambient light can be determined not by accident or guess but by the desires of the listeners in the room at that moment. Software tools for instant votes and consensus checking are already in specialized use in electronic meeting rooms of large corporations; tabs can make them widespread. The technology required for ubiquitous computing comes in three parts: cheap, low-power computers that include equally convenient displays, a network that ties them all together, and software systems implementing ubiquitous applications. Current trends suggest that the first requirement will easily be met. Flat-panel displays containing 640x480 black-and-white pixels are now common. This is the standard size for PPC sand and is also about right for television. As long as laptops, palmtops, and notebook computers continue to grow in popularity, display prices will fall, and resolution and quality will rise. By the end of the decade, a 1000x800-pixel high-contrast display will be a fraction of a centimeter thick and weigh perhaps 100 grams. A small battery will provide several days of continuous use. Larger displays are a somewhat different issue. If an interactive computer screen is to match a whiteboard in usefulness, it must be viewable from arm's length as well as from across a room. For close viewing, the density of picture elements should be no worse than on a standard computer screen, about 80 per inch. Maintaining a density of 80 pixels per inch over an area several feet on a side implies

displaying tens of millions of pixels. The biggest computer screen made today has only about one-fourth of this capacity. Such large displays will probably be expensive, but they should certainly be available. Central-processing unit speeds, meanwhile, reached a million instructions per second in 1986 and continue to double each year. Some industry observers believe that this exponential growth in raw chip speed may begin to level off about 1994, but that other measures of performance, including power consumption and auxiliary functions, will still improve. The 100-gram flat-panel display, then, might be driven by a single microprocessor chip that executes a billion operations per second and contains 16 megabytes of onboard memory along with sound, video, and network interfaces. Such a processor would draw, on average, a few percent of the power required by the display.

Internet of Things:

In the Internet of Things (IoT), resources constrained tiny sensors and devices could be connected to unreliable and untrusted networks. Nevertheless, securing IoT technology is mandatory, due to the relevant data handled by these devices. An intrusion Detection System (IDS) is the most efficient technique to detect attackers with high accuracy when cryptography is broken. This is achieved by combining the advantages of anomaly and signature detection, which are high detection and low false-positive rates, respectively. To achieve a high detection rate, the anomaly detection technique relies on a learning algorithm to model the normal behavior of a node and when a new attack pattern (often known as signature) is detected, it will be modeled with a set of rules. This latter is used by the signature detection technique for attack confirmation. However, the activation of anomaly detection for low-resource IoT devices could generate a high-energy consumption, specifically when this technique is activated all the time. Using game theory and with the help of Nash equilibrium, anomaly detection is activated only when a new attack's signature is expected to occur. This will make a balance between accuracy detection and energy consumption. Simulation results show that the proposed anomaly detection approach requires a low energy consumption to detect the attacks with high accuracy (i.e., high detection and low false-positive rates). Index Terms— Anomaly detection, Low-resources devices, Game theory, Nash equilibrium. The Internet of Things (IoT) can incorporate transparently a large number of heterogeneous devices such as, for instance, cameras, wireless sensor networks (WSN), smart meters, vehicles, etc while providing open access to a variety of data generated by such devices to provide new

services to citizens and companies. Due to the services that IoT technology affords, it finds applications in many different domains such as medical aids, automotive, smart grid, and many others. The relevant data exchanged between IoT devices are more vulnerable to attacks since they are often deployed in a hostile and insecure environment. Therefore, security solutions are mandatory to protect IoT devices from intruder attacks. In this paper, our aim is to secure low resources IoT devices such as smart meters and sensors against any malicious behaviors. The Intrusion Detection System (IDS) is very effective to protect IoT devices against intruders since it has the capability to detect both internal and external attacks with high accuracy. To monitor and detect malicious devices, detection techniques can be classified into two main approaches:

(i) Signature-based detection (or Misuse detection), which is based on the detection of the attack type by comparing the behavior of the analyzed target to a set of predefined rules related to each attack signature. Such a technique aims to reduce the false positive and it requires a low computation overhead to model the normal behavior of a device. Nevertheless, the drawback of this technique is that it can only detect known attacks, described by a set of signatures.

(ii) Anomaly detection, which uses a supervised learning algorithm, such as data mining, support vector machine (SVM), and neural networks (NNs), to build the normal behavior. The advantage of such a technique is its high detection rate since it can detect new attacks that have never occurred before. However, the main drawback is the high computation overhead required to model the normal behavior. According to several research works, the combination of anomaly and signature detection techniques (defined as a hybrid intrusion detection system) incurs high detection and low false-positive rates. However, the activation of anomaly detection for low-resource IoT devices could generate a high energy consumption due to the computational cost leading to a rapid decrease in the network lifetime, specifically when this technique is activated all the time. Thereby, our aim in this paper is to make a dilemma between energy consumption and accuracy detection (i.e., high detection and low false-positive rates) by activating the anomaly detection only when a new attack pattern (i.e., signature) is expected to occur. This dilemma is achieved, thanks to a proposed security game model, where we modeled the security strategy as a game formulation between the intruder attack and the IDS agent embedded in the IoT device. With

the help of Nash Equilibrium, we determine 1 the equilibrium state that allows the IDS agent to activate its anomaly detection technique to detect a new attack pattern. This paper is organized as follows: In section II, we put a highlight in context with the related work in this area. In Section III, we explain our anomaly detection technique based on game theory. Finally, we conclude our work and give directions for future works.

II. RELATED WORK IDS provides effective protection to IoT networks against both external and internal intruders and acts as a second wall of defense when cryptography is broken. In recent research works, the authors use an anomaly detection technique to monitor the smart grid's IoT devices such as smart meters and identify any external or internal attack that targets the grid. According to their simulation results, the anomaly detection technique, which is based on a learning algorithm, exhibits a high detection rate (i.e., above 90%). However, embedding this heavy detection technique for low-resource IoT devices could incur a high computation overhead and subsequently degrades the smart grid performances. The authors propose a hybrid intrusion detection framework for a heterogonous WSN, where a signature detection technique runs at each sensor node and an anomaly detection technique runs at a powerful node, e.g., cluster-head, or base station. The anomaly detection technique computes a rule related to each attack's signature that it detects and forwards this new rule to sensor nodes (located within its range). The sensor adds the rule into its database and compares the behavior of a monitored node with the stored rules (related to each signature). If a match occurs, the analyzed node is defined as an attacker. Such hybrid detection incurs a high communication overhead since a huge number of signatures are forwarded to sensor nodes, specifically when the number of attackers is higher in the network. Both anomaly detection and signature-based detection techniques run at the same sensor node. According to their simulation results, their hybrid intrusion detection system generates a high detection rate with a low false-positive rate. However, the major drawback of this work is that a heavy machine-learning algorithm is activated in a permanent fashion at each sensor to build intrusion rules. Therefore, a high computation overhead could be generated leading to a rapid decrease in the network lifetime. The anomaly detection technique can detect almost all the attacks that occur in a network. However, a permanent activation (i.e., does not switch to idle time) of this technique for low-resource IoT devices could decrease rapidly during their lifetimes. Thereby, in this paper, we make a dilemma between constrained energy resources and

accuracy detection by activating the anomaly detection only when a new attack's signature will be expected to occur.

3. REQUIREMENT ANALYSIS

REQUIREMENT ANALYSIS

As a basis, an article on all the different requirements for software development was taken into account during this process. We divide the requirements into 2 types: Functional and Non-functional requirements.

Requirement analysis in system engineering and software engineering encompasses those tasks that determine conditions to meet for a new product, taking account of possibly conflicting requirements of the various stakeholders such as users.

3.1 Functional Requirements

In software engineering and system engineering, a functional requirement defines a function of a system or component. A function is described as a set of inputs, behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish behavioral requirements describe all the cases where the system uses the functional requirements are captured in use cases.

- Firstly, we obtain the NSL-KDD dataset from an open source.
- Then we perform data preprocessing by cleaning the unnecessary attributes and filtering out the missing data.
- We now build a BATMC Model by making use of the BLSTM algorithm.
- Evaluation of the BATMC model is done using the evaluation metrics like Accuracy(A), True positive Rate (TPR), and False Positive Rate (FPR) with the help of training and testing datasets.
- By using the Localhost (IP Address:127.0.0.1) we pass the URLs to that localhost to check whether that site is malicious or not.

3.2 Non-Functional Requirements

Describe user-visible aspects of the system that are not directly related to the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e., how fast the system reacts to user commands.) or accuracy.

- Accuracy & Performance
- Recoverability

- Security & Privacy
- Testability
- Availability
- Usability
- Reliability
- Interoperability
- Maintainability

3.3 Computational Resources

3.3.1 Hardware Requirements

- Processor : Intel I3 or above.
- Ram : 4GB or above.
- Hard Disk : 250GB or above.

3.3.2 Software Requirements

Software requirements establish the agreement between your team and the customer on what the application is supposed to do. Without a description of what features will be included and details on how the features will work, the users of the software can't determine if the software will meet their needs. The key software requirements required for the project are:

- PLATFORM : Python
- OPERATING SYSTEM : Windows10
- IDE : Python IDE 3.7v

➢ Pandas

➢ Scikit learn

➢ Matplotlib

➢ Numpy

- **Python:**

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

- **Pandas:**

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Pandas is a Python library used for working with data sets.

- **Scikit Learn:**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. Rather than focusing on loading, manipulating, and summarizing data, the Scikit-Learn library is focused on modeling the data. Some of the most popular groups of models provided by Sklearn are as follows:

➢ Supervised Learning algorithms

➢ Unsupervised Learning algorithms

➢ Clustering

➢ Ensemble methods

➢ Feature extraction

- **Matplotlib:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt

Formatting the style of your plot:

For every x, y pair of arguments, there is an optional third argument which is the format string that indicates the color and line type of the plot. The letters and symbols of the

format string are from MATLAB, and you concatenate a color string with a line-style string. The default format string is 'b-', which is a solid blue line.

- **Numpy:**

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It also discusses the various array functions, types of indexing, etc.

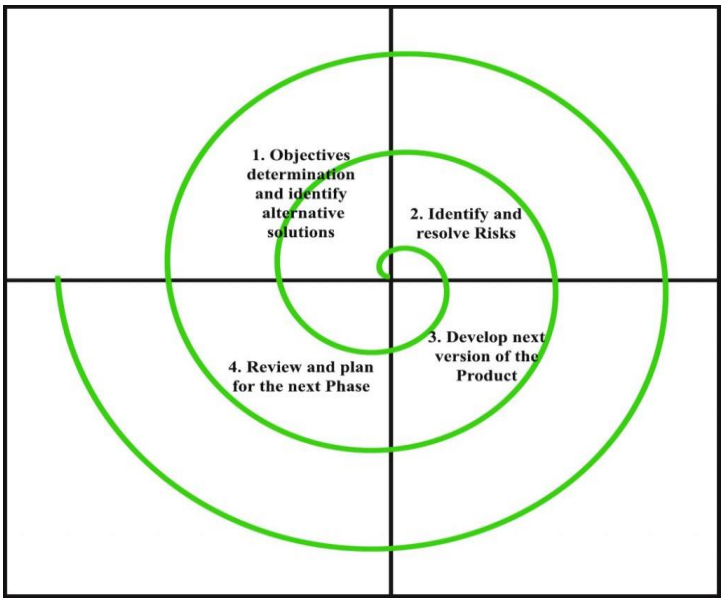
3.4 Life Cycle Model

3.4.1 Spiral Model

The spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

The below diagram shows the different phases of the Spiral Model: –



3.41 Spiral Model

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

- Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analyzed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
- Develop the next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.
- Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so-far-developed version of the software. In the end, planning for the next phase is started.

1. Introduction

1. Introduction

With the development and improvement of Internet technology, the Internet is providing various convenient services for people. However, we are also facing various security threats. Network viruses, eavesdropping, and malicious attacks are on the rise, causing network security to become the focus of attention of society and government departments. Fortunately, these problems can be well solved via intrusion detection. Intrusion detection plays an important part in ensuring network information security. However, with the explosive growth of Internet business, traffic types in the network are increasing day by day, and network behaviour characteristics are becoming increasingly complex, which brings great challenges to intrusion detection.

How to identify various malicious network traffics, especially unexpected malicious network traffics, is a key problem that cannot be avoided. In fact, network traffic can be divided into two categories (normal traffics and malicious traffics). Hence, intrusion detection can be considered a classification problem. By improving the performance of classifiers in effectively identifying malicious traffics, intrusion detection accuracy can be largely improved. Machine learning methods have been widely used in intrusion detection to identify malicious traffic.

However, these methods belong to shallow learning and often emphasize feature engineering and selection. They have difficulty in feature selection and cannot effectively solve the massive intrusion data classification problem, which leads to low recognition accuracy and a high false alarm rate. In recent years, intrusion detection methods based on deep learning have been proposed successively. We proposed a malware traffic classification method, BAT: Deep Learning Methods on IDS in Wireless neural networks with traffic data as an image. This method does not need manual design features and directly takes the original traffic as the input data to the classifier. Here analysis of the viability of Recurrent Neural Networks (RNN) to detect the behaviour of network traffic by modeling it as a sequence of states that change over time. Long Short-Term Memory (LSTM) network in classifying intrusion traffics. Experimental results show that LSTM can learn all the attack classes hidden in the training data. All the above methods treat the entire network traffic as a whole consisting of a sequence of traffic bytes. They don’t make full use of domain knowledge of network traffic.

Firstly, network traffic is a traffic unit composed of multiple data packets. A Data packet is a traffic unit composed of multiple bytes. Secondly, traffic features in the same and different packets are significantly different. Sequential features between different packets need to be extracted independently. In other words, not all traffic features are equally important for traffic classification in the process of extracting features on certain network traffic.

1.1 Purpose

The intrusion detection technology can be divided into three major categories: pattern matching methods, traditional machine learning methods, and deep learning methods. In the beginning, people mainly use pattern matching algorithms for intrusion detection. The pattern matching algorithm is the core algorithm of an intrusion detection system based on feature matching. Most algorithms have been considered for use in the past. A summary of pattern matching algorithms in Intrusion Detection System: KMP algorithm, BM algorithm, BMH algorithm, BMHS algorithm, AC algorithm, and AC-BM algorithm. Experiments show that the improved algorithm can accelerate the matching speed and has a good time performance. The naive approach, Knuth-MorrisPratt algorithm, and Rabin-Karp Algorithm are compared to check which of them is most efficient in pattern/intrusion detection. Pcap files have been used as datasets in order to determine the efficiency of the algorithm by taking into consideration their running times respectively.

1.2 Existing System

An Intrusion Detection System (IDS) is a mechanism that detects intrusions or attacks against a system or a network by analyzing the activity of the network and the system. Such intruders can be internal or external. Internal intruders are users inside the network that attempt to raise their access privileges to misuse non-authorized privileges while external intruders are users outside the target network attempting to gain unauthorized access to the network. The IDS monitors the operations of a host or a network, alerting the system administrator when it detects a security violation. There are mainly three components of the IDS.

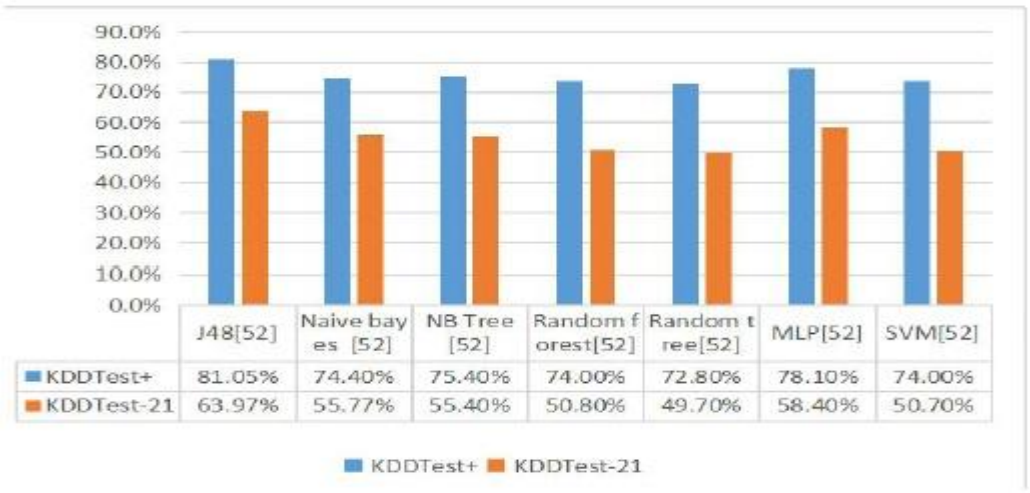
1.3 Problems in Existing System

- We review the technologies involved in the Intrusion Detection System to handle security issues in IoT environments.

1.4 Proposed System:

The BAT-MC model consists of five components, including the input layer, multiple convolutional Layers, BSLTM layer, attention layer and output layer, from bottom to top. At the input layer, BAT-MC model converts each traffic byte into a one-hot data format. Each traffic byte is encoded as an n-dimensional vector. After traffic byte is converted into a numerical form, we perform normalization operations. At the multiple convolutional layers, we convert the numerical data into traffic images. Convolutional operation is used as a feature extractor that takes an image

representation of data packet. At the BLSTM layer, BLSTM model which connects the forward LSTM, and the backward LSTM is used to extract features on the traffic bytes of each packet. BLSTM model can learn the sequential characteristics within the traffic bytes because BLSTM is suitable to the structure of network traffic. In the attention layer, attention mechanism is used to analyse the important degree of packet vectors to obtain fine-grained features which are more salient for malicious traffic detection. At the output layer, the features generated by attention mechanism are then imported into a fully connected layer for feature fusion, which obtains the key features that accurately characterize network traffic behaviour. Finally, the fused features are fed into a classifier to get the final recognition results.



	J48[52]	Naive bayes [52]	NB Tree [52]	Random forest[52]	Random tree[52]	MLP[52]	SVM[52]
KDDTest+	81.05%	74.40%	75.40%	74.00%	72.80%	78.10%	74.00%
KDDTest-21	63.97%	55.77%	55.40%	50.80%	49.70%	58.40%	50.70%

Fig 1.4.1 Performance of BAT-MC model and other machine learning models.

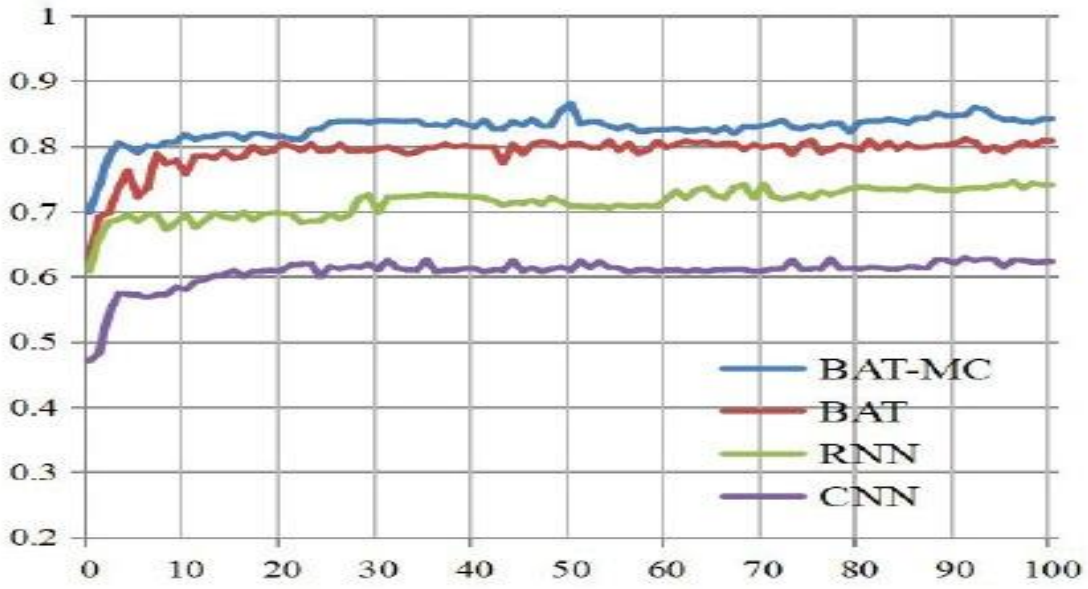


Fig 1.4.2 Comparison of Accuracy with different models.

1.5 Advantages of Proposed System:

- We propose an end-to-end deep learning model BAT-MC that is composed of BLSTM and attention mechanisms. BAT-MC can well solve the problem of intrusion detection and provide a new research method for intrusion detection
- We introduce the attention mechanism into the BLSTM model to highlight the key input. The attention mechanism conducts feature learning on sequential data composed of data package vectors. The obtained feature information is reasonable and accurate.
- We compare the performance of BAT-MC with traditional deep learning methods, the BAT-MC model can extract information from each packet. By making full use of the structure information of network traffic, the BAT-MC model can capture features more comprehensively.
- We evaluate our proposed network with a real NSL-KDD dataset. The experimental results show that the performance of BAT-MC is better than the traditional methods.

2. Literature Survey

2. Literature Survey

This paper addresses the Internet of Things. The main enabling factor of this promising paradigm is the integration of several technologies and communications solutions. Identification and tracking technologies, wired and wireless sensor and actuator networks, enhanced communication protocols (shared with the Next Generation Internet), and distributed intelligence for smart objects are just the most relevant. As one can easily imagine, any serious contribution to the advance of the Internet of Things must necessarily be the result of synergetic activities conducted in different fields of knowledge, such as telecommunications, informatics, electronics, and social science. In such a complex scenario, this survey is directed to those who want to approach this complex discipline and contribute to its development. Different visions of this Internet of Things paradigm are reported and enabling technologies reviewed. What emerges is that still major issues shall be faced by the research community. The most relevant among them are addressed in detail.

Introduction

The Internet of Things (IoT) is a novel paradigm that is rapidly gaining ground in the scenario of modern wireless telecommunications. The basic idea of this concept is the pervasive presence around us of a variety of things or objects – such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. – which, through unique addressing schemes, can interact with each other and cooperate with their neighbours to reach common goals [1]. Unquestionably, the main strength of the IoT idea is the high impact it will have on several aspects of everyday life and the behaviour of potential users. From the point of view of a private user, the most obvious effects of the IoT introduction will be visible in both working and domestic fields. In this context, demotics, assisted living, e-health, and enhanced learning are only a few examples of possible application scenarios in which the new paradigm will play a leading role soon. Similarly, from the perspective of business users, the most apparent consequences will be equally visible in fields such as automation and industrial manufacturing, logistics, business/process management, and intelligent transportation of people and goods. By starting from the considerations above, it should not be surprising that IoT is included by the US National Intelligence Council in the list of six ‘‘Disruptive Civil Technologies” with potential impacts on US national power [2]. NIC foresees that ‘‘by Internet nodes may reside in everyday things – food packages, furniture, paper documents, and more”. It highlights future opportunities that will arise, starting from the idea that ‘‘popular demand combined with technology advances could drive wide-spread diffusion of an Internet of Things (IoT) that could, like the present Internet, contribute invaluably to economic development”. The possible threats deriving from widespread adoption of such a technology are also stressed. Indeed, it is emphasized that ‘‘to the extent that everyday objects

become information security risks, the IoT could distribute those risks far more widely than the Internet has to date". Many challenging issues still need to be addressed and both technological, as well as social knots, must be untied before the IoT idea is widely accepted. Central issues are making full interoperability of interconnected devices possible, providing them with an always higher degree of smartness by enabling their adaptation and autonomous behaviour, while guaranteeing trust, privacy, and security. Also, the IoT idea poses several new problems concerning the networking aspects. In fact, the things composing the IoT will be characterized by low resources in terms of both computation and energy capacity. Accordingly, the proposed solutions need to pay special attention to resource efficiency besides the obvious scalability problems. Several industrial, standardization and research bodies are currently involved in the activity of developing solutions to fulfil the highlighted technological requirements. This survey gives a picture of the current state of the art on the IoT. More specifically, it provides the readers with a description of the different visions of the Internet of Things paradigm coming from different scientific communities; reviews the enabling technologies and illustrates which are the major benefits of the spread of this paradigm in everyday lives; offers an analysis of the major research issues the scientific community still has to face. The main objective is to give the reader the opportunity of understanding what has been done (protocols, algorithms, proposed solutions) and what remains to be addressed, as well as what are the enabling factors of this evolutionary process and what are its weaknesses and risk factors. The remainder of the paper is organized as follows. In Section 2, we introduce and compare the different visions of the IoT paradigm, which are available from the literature. The IoT main enabling technologies are the subject of Section 3, while the description of the principal applications, which in the future will benefit from the full deployment of the IoT idea, are addressed in Section 4. Section 5 gives a glance at the open issues on which research should focus more, by stressing topics such as addressing, networking, security, privacy, and standardization efforts. Conclusions and future research hints are given in Section 6.

One paradigm, many visions

Manifold definitions of Internet of Things traceable within the research community testify to the strong interest in the IoT issue and to the vivacity of the debates on it. By browsing the literature, an interested reader might experience real difficulty in understanding what IoT really means, which basic ideas stand behind this concept, and which social, economic, and technical implications the full deployment of IoT will have. The reason for today's apparent fuzziness around this term is a consequence of the name ''Internet of Things" itself, which syntactically is composed of two terms. The first one pushes towards a network-oriented vision of IoT, while the second one moves the focus on generic ''objects" to be integrated into a common framework. Differences, sometimes substantial, in the IoT visions, raise from

the fact that stakeholders, business alliances, research and standardization bodies start approaching the issue from either an Internet-oriented" or a ''Things oriented" perspective, depending on their specific interests, finalities, and backgrounds. It shall not be forgotten, anyway, that the words ''Internet" and ''Things", when put together, assume a meaning which introduces a disruptive level of innovation into today's ICT world. In fact, ''Internet of Things" semantically means ''worldwide network of interconnected objects uniquely addressable, based on standard communication protocols" [3]. This implies a huge number of (heterogeneous) objects involved in the process. The object's unique addressing and the representation and storing of the exchanged information become the most challenging issues, bringing directly to a third, ''Semantic oriented", the perspective of IoT. In Fig. 1, the main concepts, technologies, and standards are highlighted and classified with reference to the IoT vision/s they contribute to characterize best. From such an illustration, it clearly appears that the IoT paradigm shall be the result of the convergence of the three main visions addressed above. The very first definition of IoT derives from a ''Things oriented" perspective; the considered things were very simple items: Radio-Frequency Identification (RFID) tags. The terms ''Internet of Things" is, in fact, attributed to The Auto-ID Labs [4], a worldwide network of academic research laboratories in the field of networked RFID and emerging sensing technologies. These institutions, since their establishment, have been targeted to architect the IoT, together with EPC global [5]. Their focus has primarily been on the development of the Electronic Product Code™ (EPC) to support the spread use of RFID in worldwide modern trading networks and to create the industry-driven global standards for the EPC global Network™. These standards are mainly designed to improve object visibility (i.e., the traceability of an object and the awareness of its status, current location, etc.). This is undoubtedly a key component of the path to the full deployment of the IoT vision, but it is not the only one. In a broader sense, IoT cannot be just a global EPC system in which the only objects are RFIDs; they are just a part of the full story! And the same holds for the alternative Unique/Universal/Ubiquitous Identifier (UID) architecture [6], whose main idea is still the development of (middleware-based) solutions for global visibility of objects in an IoT vision. It is the authors' opinion that starting from RFID-centric solutions may be positive as the main aspects stressed by RFID technology, namely item traceability, and addressability shall be addressed also by the IoT. Notwithstanding, alternative and somehow more complete, IoT visions recognize that the term IoT implies a much wider vision than the idea of a mere object's identification.

Enabling technologies

Actualization of the IoT concept into the real world is possible through the integration of several enabling technologies. In this section we discuss the most relevant ones. Note that it is not our purpose

to provide a comprehensive survey of each technology. Our major aim is to provide a picture of the role they will likely play in the IoT. Interested readers will find references to technical publications for each specific technology.

Identification, sensing and communication technologies

‘‘Anytime, anywhere, any media” has been for a long time the vision pushing forward the advances in communication technologies. In this context, wireless technologies have played a key role and today the ratio between radios and humans is nearing the 1 to 1 value [7]. However, the reduction in terms of size, weight, energy consumption, and cost of the radio can take us to a new era where the above ratio increases of orders of magnitude. This will allow us to integrate radios in almost all objects and thus, to add the world ‘‘anything” to the above vision, which leads to the IoT concept. In this context, key components of the IoT will be RFID systems [8], which are composed of one or more reader(s) and several RFID tags. Tags are characterized by a unique identifier and are applied to objects (even persons or animals). Readers trigger the tag transmission by generating an appropriate signal, which represents a query for the possible presence of tags in the surrounding area and for the reception of their IDs. Accordingly, RFID systems can be used to monitor objects in real-time, without the need of being in line-of-sight; this allows for mapping the real world into the virtual world. Therefore, they can be used in an incredibly wide range of application scenarios, spanning from logistics to e-health and security.

Applications

Potentialities offered by the IoT make possible the development of a huge number of applications, of which only a very small part is currently available to our society. Many are the domains and the environments in which new applications would likely improve the quality of our lives: at home, while travelling, when sick, at work, when jogging and at the gym, just to cite a few. These environments are now equipped with objects with only primitive intelligence, most of times without any communication capabilities. Giving these objects the possibility to communicate with each other and to elaborate the information perceived from the surroundings imply having different environments where a very wide range of applications can be deployed. These can be grouped into the following domains: Transportation and logistics domain, Healthcare domain, Smart environment (home, office, plant) domain, Personal and social domain.

Computers in 21st Century

Whenever people learn something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading. Computer scientist, economist, and Nobelist Herb Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist TK Gibson calls it "visual invariants"; philosophers Georg Gadamer and Martin Heidegger call it "the horizon" and the "ready-to-hand", John Seely Brown at PARC calls it the "periphery". All say, in essence, that only when things disappear in this way are, we freed to use them without thinking and so to focus beyond them on new goals. The idea of integrating computers seamlessly into the world at large runs counter to a number of present-day trends. "Ubiquitous computing" in this context does not just mean computers that can be carried to the beach, jungle, or airport. Even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box. By analogy to writing, carrying a super-laptop is like owning just one very important book. Today's multimedia machine makes the computer screen a demanding focus of attention rather than allowing it to fade into the background. Perhaps most diametrically opposed to our vision is the notion of "virtual reality," which attempts to make a world inside the computer. Users don special goggles that project an artificial scene on their eyes; they wear gloves or even body suits that sense their motions and gestures so that they can move about and manipulate virtual objects. Although it may have its purpose in allowing people to explore realms otherwise inaccessible -- the insides of cells, the surfaces of distant planets, the information web of complex databases – the virtual reality is only a map, not a territory. It excludes desks, offices, other people not wearing goggles and body suits, weather, grass, trees, walks, chance encounters, and in general the infinite richness of the universe. Virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists. Indeed, the opposition between the notion of virtual reality and ubiquitous, invisible computing is so strong that some of us use the term "embodied virtuality" to refer to the process of drawing computers out of their electronic shells. The "virtuality" of computer-readable data -- all the different ways in which it can be altered, processed, and analysed -- is brought into the physical world. How do technologies disappear into the background? The vanishing of electric motors may serve as an instructive example: At the turn of the century, a typical workshop or factory contained a single-engine that drove dozens or hundreds of different machines through a system of shafts and pulleys. Cheap, small, efficient electric motors made it possible first to give each machine or tool its own source of the motive force, then to put many motors into a single machine. A glance through the shop manual of a typical automobile, for example, reveals twenty-two motors and twenty-five more solenoids. They start

the engine, clean the windshield, lock and unlock the doors, and so on. By paying careful attention it might be possible to know whenever one activated a motor, but there would be no point to it. Most of the computers that participate in embodied virtuality will be invisible in fact as well as in metaphor. Already computers in light switches, thermostats, stereos, and ovens help to activate the world. These machines and more will be interconnected in a ubiquitous network. As computer scientists, however, my colleagues and I have focused on devices that transmit and display information more directly. We have found two issues of crucial importance: location and scale. If a computer merely knows what room it is in, it can adapt its behaviour in significant ways without requiring even a hint of artificial intelligence. Ubiquitous computers will also come in different sizes, each suited to a particular task. How many tabs, pads, and board-sized writing and display surfaces are there in a typical room? Look around you: at the inch, the scale includes wall notes, titles on book spines, labels on controls, thermostats, and clocks, as well as small pieces of paper. Depending upon the room you may see more than a hundred tabs, ten or twenty pads, and one or two boards. This leads to our goals for initially deploying the hardware of embodied virtuality: hundreds of computers per room. Hundreds of computers in a room could seem intimidating at first, just as hundreds of volts coursing through wires in the walls did at one time. But like the wires in the walls, these hundreds of computers will come to be invisible to common awareness. People will simply use them unconsciously to accomplish everyday tasks. Tabs are the smallest components of embodied virtuality. Because they are interconnected, tabs will expand on the usefulness of existing inch-scale computers such as the pocket calculator and the pocket organizer. Tabs will also take on functions that no computer performs today. For example, Olivetti Cambridge Research Labs pioneered active badges, and now computer scientists at PARC and other research laboratories around the world are working with these clip-on computers roughly the size of an employee ID card. These badges can identify themselves to receivers placed throughout a building, thus making it possible to keep track of the people or objects to which they are attached. In our experimental embodied virtuality, doors open only to the right badge wearer, rooms greet people by name, telephone calls can be automatically forwarded to wherever the recipient may be, receptionists know where people are, and computer terminals retrieve the preferences of whoever is sitting at them, and appointment diaries write themselves. No revolution in artificial intelligence is needed--just the proper embedding of computers into the everyday world.

Intrusion Detection System (IDS)

In the Internet of Things (IoT), resources constrained tiny sensors and devices could be connected to unreliable and untrusted networks. Nevertheless, securing IoT technology is mandatory, due to the relevant data handled by these devices. An intrusion Detection System (IDS) is the most efficient

technique to detect attackers with high accuracy when cryptography is broken. This is achieved by combining the advantages of anomaly and signature detection, which are high detection and low false-positive rates, respectively. To achieve a high detection rate, the anomaly detection technique relies on a learning algorithm to model the normal behaviour of a node and when a new attack pattern (often known as signature) is detected, it will be modelled with a set of rules. This latter is used by the signature detection technique for attack confirmation. However, the activation of anomaly detection for low-resource IoT devices could generate a high-energy consumption, specifically when this technique is activated all the time. Using game theory and with the help of Nash equilibrium, anomaly detection is activated only when a new attack's signature is expected to occur. This will make a balance between accuracy detection and energy consumption. Simulation results show that the proposed anomaly detection approach requires a low energy consumption to detect the attacks with high accuracy (i.e., high detection and low false-positive rates). Index Terms— Anomaly detection, Low-resources devices, Game theory, Nash equilibrium. The Internet of Things (IoT) can incorporate transparently a large number of heterogeneous devices such as, for instance, cameras, wireless sensor networks (WSN), smart meters, vehicles, etc while providing open access to a variety of data generated by such devices to provide new services to citizens and companies [9]. Due to the services that IoT technology affords, it finds applications in many different domains such as medical aids, automotive, smart grid, and many others [10]. The relevant data exchanged between IoT devices are more vulnerable to attacks since they are often deployed in a hostile and insecure environment. Therefore, security solutions are mandatory to protect IoT devices from intruder attacks. The Intrusion Detection System (IDS) is very effective to protect IoT devices against intruders since it has the capability to detect both internal and external attacks with high accuracy [11]. To monitor and detect malicious devices, detection techniques can be classified into two main approaches [11][12][13]:

(i) Signature-based detection (or Misuse detection), which is based on the detection of the attack type by comparing the behaviour of the analysed target to a set of predefined rules related to each attack signature. Such a technique aims to reduce the false positive and it requires a low computation overhead to model the normal behaviour of a device. Nevertheless, the drawback of this technique is that it can only detect known attacks, described by a set of signatures.

(ii) Anomaly detection, which uses a supervised learning algorithm [14], such as data mining, support vector machine (SVM), and neural networks (NNs), to build the normal behaviour. The advantage of such a technique is its high detection rate since it can detect new attacks that have never occurred before. According to several research works [13], the combination of anomaly and signature detection techniques (defined as a hybrid intrusion detection system) incurs high

detection and low false-positive rates. However, the activation of anomaly detection for low-resource IoT devices could generate a high energy consumption due to the computational cost leading to a rapid decrease in the network lifetime, specifically when this technique is activated all the time.

RELATED WORK IDS provides effective protection to IoT networks against both external and internal intruders [11] and acts as a second wall of defence when cryptography is broken. In recent research works [15][13], the authors use an anomaly detection technique to monitor the smart grid's IoT devices such as smart meters and identify any external or internal attack that targets the grid.

Techniques:

According to their simulation results, the anomaly detection technique, which is based on a learning algorithm, exhibits a high detection rate (i.e., above 90%). However, embedding this heavy detection technique for low-resource IoT devices could incur a high computation overhead and subsequently degrades the smart grid performances. In [11][13], the authors propose a hybrid intrusion detection framework for a heterogonous WSN, where a signature detection technique runs at each sensor node and an anomaly detection technique runs at a powerful node, e.g., cluster-head, or base station. The anomaly detection technique computes a rule related to each attack's signature that it detects and forwards this new rule to sensor nodes (located within its range). The sensor adds the rule into its database and compares the behaviour of a monitored node with the stored rules (related to each signature). If a match occurs, the analysed node is defined as an attacker. Such hybrid detection incurs a high communication overhead since a huge number of signatures are forwarded to sensor nodes, specifically when the number of attackers is higher in the network. In both anomaly detection and signature-based detection techniques run at the same sensor node. According to their simulation results, their hybrid intrusion detection system generates a high detection rate with a low false-positive rate. However, the major drawback of this work is that a heavy machine-learning algorithm is activated in a permanent fashion at each sensor to build intrusion rules. Therefore, a high computation overhead could be generated leading to a rapid decrease in the network lifetime. The anomaly detection technique can detect almost all the attacks that occur in a network. However, a permanent activation (i.e., does not switch to idle time) of this technique for low-resource IoT devices could decrease rapidly during their lifetimes. Thereby, in this paper, we make a dilemma between constrained energy resources and accuracy detection by activating the anomaly detection only when a new attack's signature will be expected to occur.

3. Requirement Analysis

3. Requirement Analysis

As a basis, an article on all the different requirements for software development was taken into account during this process. We divide the requirements in 2 types: Functional and Non-functional requirements.

Requirement analysis in system engineering and software engineering encompasses those tasks that determine conditions to meet for a new product, taking account of possibly conflicting requirements of the various stakeholders such as users.

3.1 Functional Requirements

In software engineering and system engineering, a functional requirement defines a function of a system or component. A function is described as a set of inputs, behaviour and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases.

- Firstly, we obtain the NSL-KDD dataset from an open-source.
- Then we perform data preprocessing by cleaning the unnecessary attributes and filtering out the missing data.
- We now build a BATMC Model by making use of the BLSTM algorithm.
- Evaluation of the BATMC model is done using the evaluation metrics like Accuracy(A), True positive Rate (TPR), and False Positive Rate (FPR) with the help of training and testing datasets.
- By using the Localhost (IP Address:127.0.0.1) we pass the URLs to that localhost to check whether that site is malicious or not.

2.2 Non-Functional Requirements

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e., how fast the system reacts to user commands.) or accuracy.

- Accuracy & Performance
- Recoverability
- Security & Privacy
- Testability
- Availability
- Usability

- Reliability
- Interoperability
- Maintainability

3.3 Computational Resources

3.3.1 Hardware Requirements

- Processor : Intel I3 or above.
- Ram : 4GB or above.
- Hard Disk : 250GB or above.

3.3.2 Software Requirements

Software requirements establish the agreement between your team and the customer on what the application is supposed to do. Without a description of what features will be included and details on how the features will work, the users of the software can't determine if the software will meet their needs. The key software requirements required for the project are:

- PLATFORM : Python
- OPERATING SYSTEM : Windows10
- IDE : Python IDE 3.7v
- Pandas
- Scikit learn
- Matplotlib
- Numpy

- **Python:**

  Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

- **Pandas:**

  Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. Pandas is a Python library used for working with data sets.

- **Scikit Learn:**

  Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. Rather than focusing on loading, manipulating, and summarizing data, Scikit-learn library is focused on modelling the data. Some of the most popular groups of models provided by Sklearn are as follows:

  ➢ Supervised Learning algorithms

  ➢ Unsupervised Learning algorithms

  ➢ Clustering

  ➢ Ensemble methods

  ➢ Feature extraction

- **Matplotlib:**

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt

  Formatting the style of your plot:

  For every x, y pair of arguments, there is an optional third argument which is the format string that indicates the colour and line type of the plot. The letters and symbols of the format string are from MATLAB, and you concatenate a colour string with a line-style string. The default format string is 'b-', which is a solid blue line.

- **Numpy:**

  NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. It also discusses the various array functions, types of indexing, etc.

3.4 Life Cycle Model

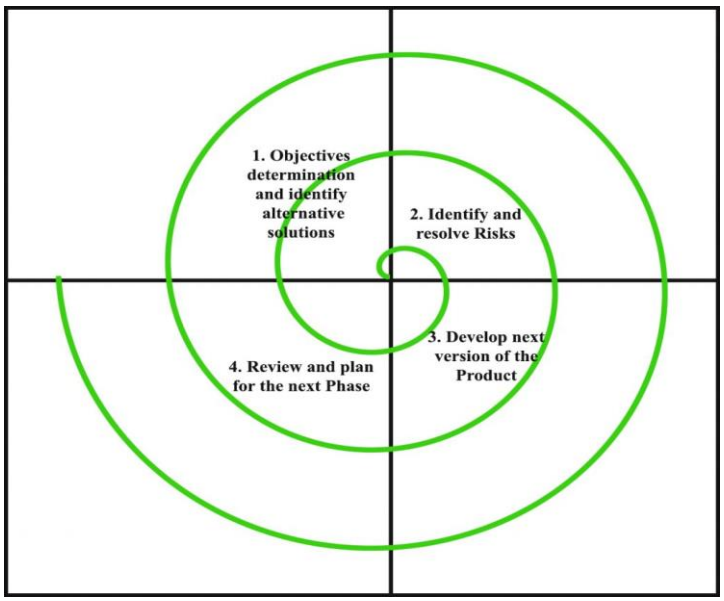
3.4.1 Spiral Model:

The Spiral model is one of the most important Software Development Life Cycle models, which provides support for Risk Handling. In its diagrammatic representation, it looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a Phase of the software development process. The exact number of phases needed to develop the product can be varied by the project manager depending upon the project risks. As the project

manager dynamically determines the number of phases, so the project manager has an important role to develop a product using the spiral model.

The Radius of the spiral at any point represents the expenses(cost) of the project so far, and the angular dimension represents the progress made so far in the current phase.

The below diagram shows the different phases of the Spiral Model: –



3.41 Spiral Model

Each phase of the Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are discussed below-

- Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated, and analysed at the start of every phase. Then alternative solutions possible for the phase are proposed in this quadrant.
- Identify and resolve Risks: During the second quadrant, all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution are identified and the risks are resolved using the best possible strategy. At the end of this quadrant, the Prototype is built for the best possible solution.
- Develop the next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.

- Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so-far-developed version of the software. In the end, planning for the next phase is started.

4. Design

4. Design

4.1 Architecture:

4.1.1 System Architecture:

It describes the structure and behavior of the technology infrastructure of an enterprise, solution, or system. In other words, the system architecture can be described as the flow of the application represented below in the pictorial format.

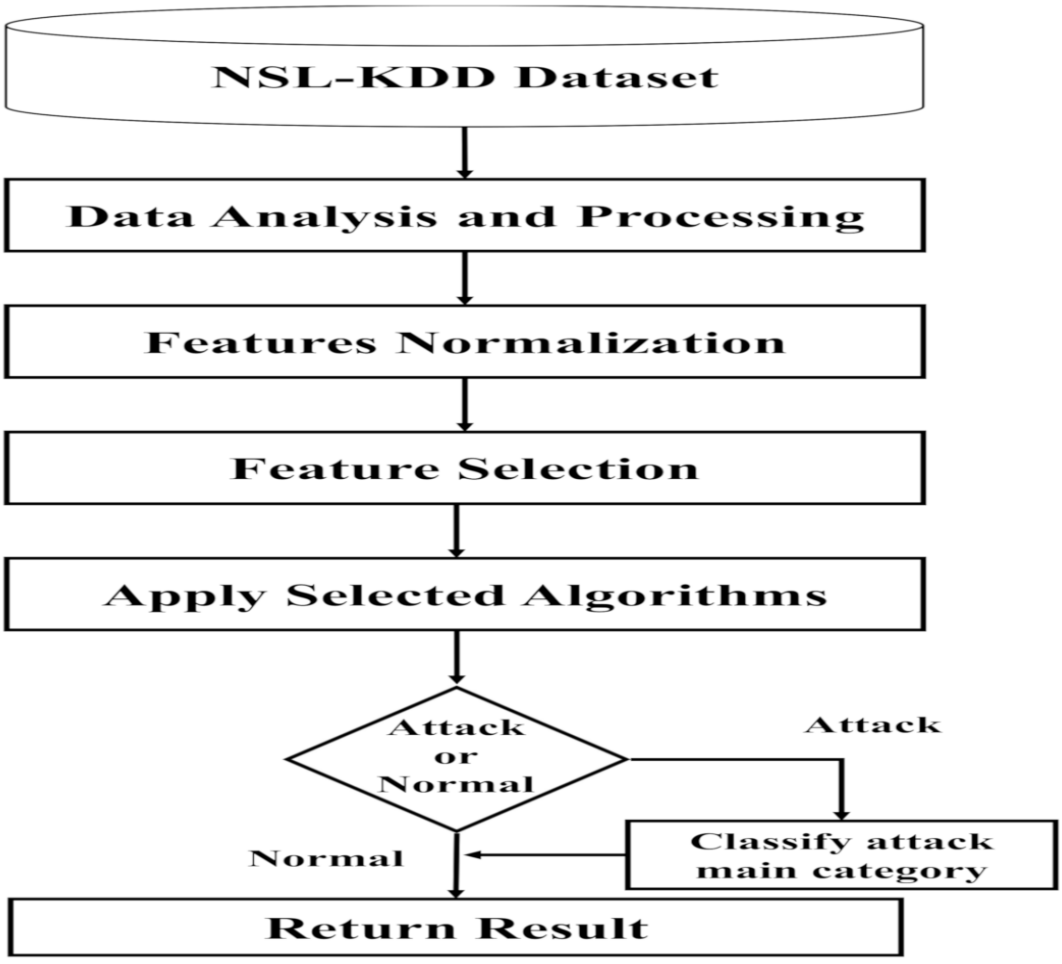


Figure 4.1.1: System Architecture

This flow chart represents the overall process. The process starts with the data source, the NSL-KDD dataset, followed by data analysis and some pre-processing techniques for data cleaning. Feature normalization and feature selection are performed in order to implement the later selected tree-based machine learning algorithm to classify whether there is an attack or not and the types of attack.

4.1.2 Technical Architecture:

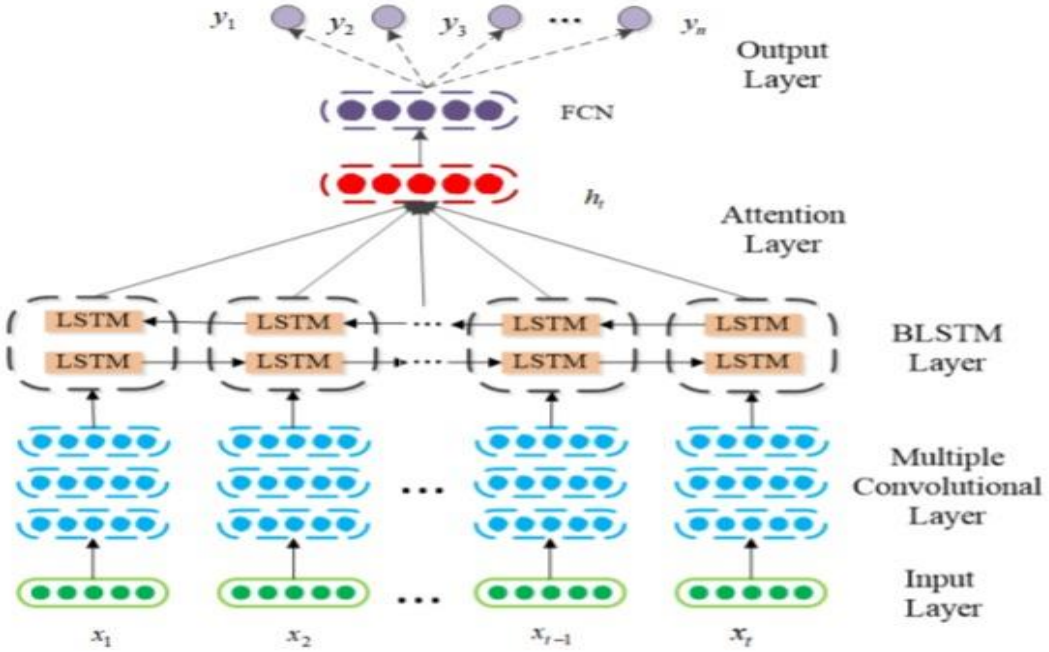


FIGURE 1. The Architecture of BAT-MC model. The whole architecture is divided into five parts.

Figure 4.2.1: Technical Architecture

A. Data Preprocessing Layer

There are three symbolic data types in NSL-KDD data features protocol type, flag, and service. We use a one-hot encoder to map these features into binary vectors. One-Hot Processing: NSL-KDD dataset is processed by a one-hot method to transform symbolic features into numerical features. For example, the second feature of the NSL-KDD data sample is protocol type. The protocol type has three values: tcp, udp, and icmp. One-hot method is processed into a binary code that can be recognized by a computer, where tcp is [1, 0, 0], udp is [0, 1, 0], and icmp is [0, 0, 1]. Normalization Processing: The value of the original data may be too large, resulting in problems such as “large numbers to eat decimals”, data processing overflows, inconsistent weights so on. We use a standard scaler to normalize the continuous data into the range [0, 1]. Normalization processing eliminates the influence of the measurement unit on the model training and makes the training result more dependent on the characteristics of the data itself. The formula is shown in equation (1) and equation (2).

r′cr−rminrmax−rmin,rmax=max{r},(1)(2)

Where r stands for numeric feature value, rmin stands for the minimal value of the feature, rmax stands for the max value, and r′ stands for the value after the normalization.

B. Multiple Convolutional Layers

After the above processing operations, the convolutional layer is used to capture the local features of traffic data. The convolutional layer is the most important part of the CNN, which convolves the input images (or feature maps) with multiple convolutional kernels to create different feature maps. The shallower convolutional layers whose receptive field is narrow can extract local information, while the deeper layers can capture global information with a larger visual field. Hence, as the number of the convolutional layer's increases, the scale of the convolutional feature gradually becomes coarser. In this paper, the input of the convolutional layer can be formulated as a tensor of the size H×W×1, where H and W denote the height and width of data yielded by normalization processing. Suppose we have some N unites layer as input which is followed by a convolutional layer. If we use m width filter w, the convolutional output will be (N−m+1) unites. The convolutional calculation process is as shown in equation (3).

$$x_{l,ji,k}=f\left(b_j+\sum_{a=1}^{m} w_{ja,krl-1,ji+(k-1)\times s+a-1}\right), \quad (3)$$

where xi, ji ,k is one of i th unit of j feature map of the k th section in the l th layer, and s is the range of section. f is a non-linear mapping, it usually uses a hyperbolic tangent function, tanh(·).

C. BLSTM Layer

For the time series data composed of traffic bytes, BLSTM can effectively use the context information of data for feature learning. The BLSTM is used to learn the time series feature in the data packet. Traffic bytes of each data packet are sequentially input into a BLSTM, which finally obtains a packet vector. BLSTM is an enhanced version of LSTM (Long Short-Term Memory). The BLSTM model is used to extract coarse-grained features by connecting forward LSTM and backward LSTM. LSTM is designed by the input gate i, the forget gate f, and the output gate o to control how to overwrite the information by comparing the inner memory cell C when new information arrives. When information enters an LSTM network, we can judge whether it is useful according to relevant rules. Only the information that meets algorithms authentication will remain, and inconsistent information will be forgotten through forget gate. Given an input sequence x=(x0,…,xt) at time t and the hidden states of a BLSTM layer, h=(h0,…, ht) can be derived as follows.

The forget gate will take the output of hidden layer ht−1 at the previous moment and the input xt at the current moment as input to selectively forget in the cell state Ct, which can be expressed as:

$$f_t = \text{sigmoid}\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right), \quad (4)$$

The input gate cooperates with a tanh function together to control the addition of new information. tanh generates a new candidate vector. The input gate generates a value for each item in C˜t from 0 to 1 to control how much new information will be added, which can be expressed as:

Ct=it=C˜t=sigmoid(ft·Ct−1+it·C˜t), sigmoid (Wxixt+Whiht−1+bt), tanh (Wcxt+Wcht−1+bc), (5)(6)(7)

The output gate is used to control how much of the current unit state will be filtered out, which can be expressed as:

ot=sigmoid (Wxoxt+Whoht−1+bo), (8)

For the BLSTM model at time t, the hidden states of the ht that is a packet vector generated from each packet can be defined as the concatenation of h←t and h→t, which can be expressed as:

ht=h→t=h←t=h←t+h→t, tanh (Wxh→xt+Wh→h→h→t−1+bh→),tanh(Wxh←xt+Wh←h←h←t−1+bh←), (9)(10)(11)

where ′.′ means the pointwise product. x represents the input of the heterogeneous time series data. h→t and h←t are the hidden states of the forward LSTM layer and backward LSTM layer at time t. All the matrices W are the connection weights between two units, and b are bias vectors.

D. Attention Layer

BLSTM eventually generates a packet vector for each packet. These packet vectors are arranged in the order of interaction between the two parties in the network stream to form a sequence of packet vectors. The relationships within packet vectors will be learned by the attention layer. similarly, to the attention mechanism is used to adjust the probability of packet vectors so that our model pays more attention to important features. Firstly, the packet vectors ht extracted by the BLSTM model is used to obtain its implicit representation ut through a nonlinear transformation, which can be expressed as:

ut=tanh (Wwht+bw), (12)

We next measure the importance of packet vectors based on the similarity representation ut with a context vector uw and obtain the normalized importance weight coefficient αt. uw is a random initialization matrix that can focus on important information over ut. The weight coefficient for the above coarse-grained features can be expressed as:

αt=exp(uTtuw)∑exp(uTtuw), (13)

Finally, the fine-grained feature s can be computed via the weighted sum of ht based on αt . s can be expressed as:

s=∑αtht, (14)

The fine-grained feature vector s generated from the attention mechanism is used for malicious traffic recognition with a softmax classifier, which can be expressed as:

$$y=softmax(Whs+bh), \quad (15)$$

where Wh represents the weight matrix of the classifier, which can map s to a new vector with length h, h is the number of categories of network traffic.

E. Model Training

Training the proposed network contains a forward pass and a backward pass.

Forward Propagation: The BAT-MC model is mainly composed of BLSTM layer and attention layer, each of which presents different structures and thus plays a different role in the whole model. The forward propagation is conducted from BLSTM layer to the attention layer. The input of the current model is obtained by the processing of the previous model. After the completion of forwarding propagation, the final recognition result is obtained. The NSL-KDD dataset is defined as X. The divided training dataset and testing dataset can be expressed as x1,x2,x3. After a one-hot operation and normalization operation, every sample is converted into a format X'' that can be acceptable to the BAT-MC model. Meanwhile, we set the cell state vector size as State. In summary, the abnormal traffic detection algorithm based on the BAT-MC model is summarized as Algorithm 1. The objective function of our model is the cross-entropy-based cost function. The goal of training this model is to minimize the cross-entropy of the expected and actual outputs for all activities. The formula is shown:

$$C= -\sum i\sum j yjilnaji+(1-yji)ln(1-aji), \quad (16)$$

where i is the index of network traffic. And j is the traffic category. a is the actual category of network traffic and y is the predicted category.

5. Modules

5. Modules

The system after a careful analysis has been identified to be presented with the following modules:

5.1 Data Collection:

The main data sets were collected from the Kaggle deep learning repository which is an open data resource for data mining and for predictive analytics purposes. The acquired data source was a CSV file.

5.2 Data Cleaning:

The data in the CSV files need to be checked to whether it has any missing values, as the data source has missing values every attribute has been checked using the filters, and null values are removed which helps to increase the accuracy level.

5.3 Benchmark Datasets:

The final result of network traffic anomaly detection is closely related to the dataset. The NSL-KDD dataset is an enhanced version of KDD cup 1999 dataset, which is widely used in intrusion detection experiments. The NSL-KDD dataset not only effectively solves the inherent redundant records problems of the KDD Cup 1999 dataset but also makes the number of records reasonable in the training dataset and testing dataset. The NSL-KDD dataset is mainly composed of KDDTrain+ training dataset, KDDTest+ and KDDTest-21 testing dataset, which can make a reasonable comparison with different methods of the experimental results. As shown in Table 1, the NSL-KDD dataset has different normal records and four different types of abnormal records. The KDDTest-21 dataset is a subset of the KDDTest+ and is more difficult to classify.

TABLE 5.3.1 Different Classifications in the NSL-KDD Dataset

	Total	Normal	Dos	Probe	R2L	U2L
KDDTrain+	125973	67343	45927	11656	995	52
KDDTest+	22544	9711	7458	2421	2754	200
KDDTest-21	11850	2152	4342	2402	2754	200

Network traffic is generally collected at fixed time intervals. Essentially, network traffic data is a kind of time-series data. Network traffic is a traffic unit composed of multiple data packets. Each data packet is seen as a whole consisting of a sequence of traffic bytes. There are 41 features from the different data

packets and 1 class label for every data packet. It can be described in the following form: x = (b0,…,bi ,..). bi is the i -th feature in a data packet, and x represents a continuous feature of a data packet. These features include basic features, content features, and traffic features. According to its characteristics, there are four types of attacks in this dataset: DoS (Denial of Service attacks), R2L (Root to Local attacks), U2R (User to Root attack), and Probe (Probing attacks)

5.4 Modelling:

We propose an end-to-end deep learning model BAT-MC that is composed of BLSTM and attention mechanisms. BAT-MC can well solve the problem of intrusion detection and provide a new research method for intrusion detection.

5.5 Evaluation Metrics:

Accuracy (A) is used to evaluate the BAT-MC model. Except for accuracy, false positive rate (TPR) and false positive rate (FPR) are also introduced. These three indicators are commonly used in the research field of network traffic anomaly detection, and the calculation formula is shown as follows. Where True Positive (TP). represents the correct classification of the Intruder. False Positive (FP) represents the incorrect classification of a normal user taken as an intruder. True Negative (NP) represents a normal user classified correctly. False Negative (FN) represents an instance where the intruder is incorrectly classified as a normal user.

Accuracy represents the proportion of correctly classified samples to the total number of samples. The evaluation metric is defined as follows:

accuracy, $A=TP+TN \Big/ TP+FP+FN+TN.$

True Positive Rate (TPR): as the equivalent of the Detection Rate (DR), it represents the percentage of the number of records correctly identified over the total number of anomaly records.

$DR=TPR=TP \Big/ TP+FN.$

False Positive Rate (FPR) represents the percentage of the number of records rejected incorrectly divided by the total number of normal records. The evaluation metric is defined as follows:

$FPR=FP \Big/ FP+TN.$

5.6 Visualization:

By giving train and test data to the BAT-MC model it identifies the malicious URLs from normal URLs. It displays the output in the text format in a short window within a localhost browser.

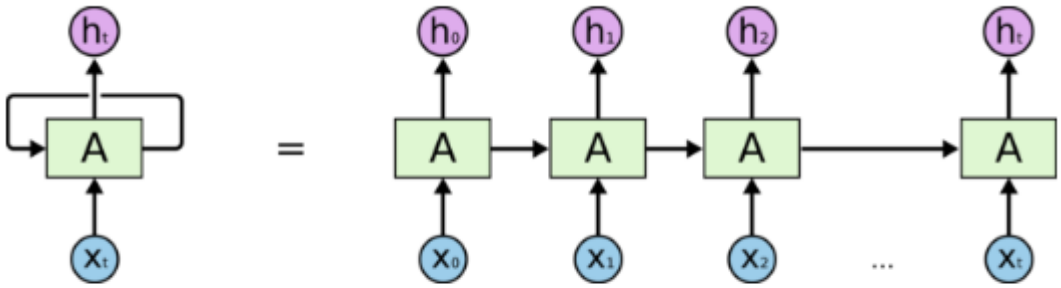
6. Algorithms

6. Algorithms

6.1 Recurrent Neural Network:

A Recurrent Neural Network is a generalization of a feedforward neural network that has internal memory. RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past computation. After producing the output, it is copied and sent back into the recurrent network. For making a decision, it considers the current input and the output that it has learned from the previous input.

Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. In other neural networks, all the inputs are independent of each other. But in RNN, all the inputs are related to each other.



An unrolled recurrent neural network.

Figure 6.1.1: RNN Network

First, it takes the X(0) from the sequence of input, and then it outputs h(0) which together with X(1) is the input for the next step. So, h(0) and X(1) are the input for the next step. Similarly, h(1) from the next is the input with X(2) for the next step and so on. This way, it keeps remembering the context while training.

The formula for the current state is:

$$h_t = f(h_{t-1}, x_t)$$

Applying Activation Function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

W is *weight*, **h** is the *single hidden vector*, **Whh** is *the weight at previous hidden state*, **Whx** is the *weight at current input state*, **tanh** is the *Activation funtion*, that implements a Non-linearity that squashes the activations to the range[-1.1]

Output:

$$y_t = W_{hy}h_t$$

$$y_t = W_{hy}h_t$$

Yt is the *output state*. **Why** is the *weight at the output state.*

6.1.1 Advantages of Recurrent Neural Network:

1. **RNN** can model the sequence of data so that each sample can be assumed to be dependent on previous ones
2. Recurrent neural networks are even used with convolutional layers to extend the effective pixel neighborhood.

6.1.2 Disadvantages of Recurrent Neural Network:

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using *tanh* or *relu* as an activation function.

6.2 Long Short-Term Memory:

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory. The vanishing gradient problem of RNN is resolved here. LSTM is well-suited to classify, process, and predict time series given time lags of unknown duration. It trains the model by using back-propagation. In an LSTM network, three gates are present:

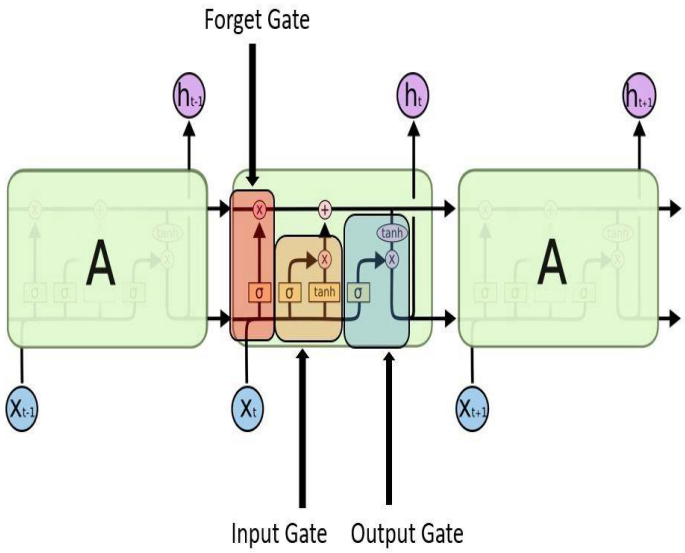


Figure 6.2.1: LSTM gates

1. **Input gate** — discover which value from input should be used to modify the memory. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from-**1** to **1.**

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] \; + \; b_C\right)$$

Input gate

2. Forget gate — discover what details to be discarded from the block. It is decided by the **sigmoid function.** it looks at the previous state(**ht-1**) and the content input (**Xt**) and outputs a number between **0(***omit this*) and **1(***keep this***)** for each number in the cell state **Ct−1**.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget gate

3. Output gate — the input and the memory of the block is used to decide the output. **Sigmoid** function decides which values to let through **0,1.** and **tanh** function gives weightage to the values which are passed deciding their level of importance ranging from-**1** to **1** and multiplied with output of **Sigmoid.**

$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Output Gate

7. Implementation

7. Implementation

7.1 Pseudo Code:

7.1.1 Source Code for Views:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
  def warn(*args, **kwargs):
     pass

  from django.shortcuts import render
  from django.http import HttpResponse, JsonResponse
  from django.db.models import Q
  from .models import *

  # Create your views here.
  def error_404_view(request, exception):
     return render(request,'404.html')

  def getdoc(request):
     return render(request,'ppt.html')

  def index(request):
     try:
        return render(request, 'index.html')
     except:
        return render(request, '404.html')



  def getuserfeedbackform(request):
     try:
        return render(request, 'userfeedbackform.html')
```

```
except:
        return render(request, '404.html')


def saveuserfeedbackform(request):
    try:
        obj = UserFeedBack()
        obj.title = request.GET['usertitle']
        obj.description = request.GET['userdescription']
        obj.save()
        mydict = {'feedback': True}
        return render(request, 'userfeedbackform.html', context=mydict)
    except:
        return render(request, '404.html')


import warnings
warnings.warn = warn
import warnings
import joblib
from lxml import html
from json import dump, loads
from requests import get
import json
from re import sub
from dateutil import parser as dateparser
from time import sleep
from django.http import HttpResponse
from django.shortcuts import render
import os
import pickle
import joblib


import whois
```

```
import datetime


def result(request):


    """try:"""
    #nm=request.GET['url']


    text=request.GET['url']
    if not text.startswith('http'):
        return render(request,"404.html")
    if text.startswith('https://malicious-url-detectorv5.herokuapp.com/') :
        return render(request,'result.html',{'result':'Real-time analysis successfull','f2':'Legtimate','mal': True,'text':text,'name':"The Legions",
                'org':"The Legions",
                'add':"New Delhi",
                'city':"New Delhi",
                'state':"New Delhi",
                'ziip':"201301",
                'country':"India",'emails':"thelegions@gmail.com",
                'dom':"Hidden For Privacy",'rank':"Hidden For Privacy","tags":"Hidden For Privacy","registrar":"Hidden For Privacy","var13":"NA","varab":"NA","var11":"NA","var10":"NA","var5":"NA","var4":"NA","var3":"NA"})



    elif text.startswith('https://www.google.com/search?q='):
        return render(request,'result.html',{'result':'Real-time analysis successfull','f2':'Legtimate','mal': True,'text':text,'name':"NA for google search",
                'org':"NA for google search",
                'add':"NA for google search",
                'city':"NA for google search",
                'state':"NA for google search",
                'ziip':"NA for google search",
                'country':"NA for google search",'emails':"NA for google search",
```

```
'dom':"NA for google search",'rank':"NA for google search","tags":"NA for google search","registrar":"Hidden For Privacy","var13":"NA for google search","varab":"NA for google search","var11":"NA for google search","var10":"NA for google search","var5":"NA for google search","var4":"NA for google search","var3":"NA for google search"})


    elif text.startswith('https://www.youtube.com/watch?v='):
        return render(request,'result.html',{'result':'Real-time analysis successfull','f2':'Legtimate','mal': True,'text':text,'name':"NA for Youtube search",
                'org':"NA for Youtube search",
                'add':"NA for Youtube search",
                'city':"NA for Youtube search",
                'state':"NA for Youtube search",
                'ziip':"NA for Youtube search",
                'country':"NA for Youtube search",'emails':"NA for Youtube search",
                'dom':"NA for Youtube search",'rank':"NA for Youtube search","tags":"NA for Youtube search","registrar":"Hidden For Privacy","var13":"NA for Youtube search","varab":"NA for Youtube search","var11":"NA for Youtube search","var10":"NA for Youtube search","var5":"NA for Youtube search","var4":"NA for Youtube search","var3":"NA for Youtube search"})


    elif (text.startswith('https://www.google.com/search?q=')==False ):


        if text.startswith('https://') or text.startswith('http://'):
            var13="Not Applicable"
            varab="Not Applicable"
            var11="Not Applicable"
            var10="Not Applicable"
            var5="Not Applicable"
            var4="Not Applicable"
            var3="Not Applicable"


            if len(text)<=9:
                return render(request,'errorpage.html')
```

```
aburl=-1
digits="0123456789"
if text[8] in digits:
    oneval=-1
else:
    oneval=1
if len(text)>170:
    secval=-1
else:
    secval=1
if "@" in text:
    thirdval=-1
    var3="@detected"
else:
    thirdval=1
k=text.count("//")
if k>1:
    fourthval=-1
    var4="More Redirects"
else:
    fourthval=1

if "-" in text:
    fifthval=-1
    var5="Prefix-Suffix detected"
else:
    fifthval=1
if "https" in text:
    sixthval=1
else:
    sixthval=-1
temp=text
```

```
temp=temp[6:]
k1=temp.count("https")

if k1 >=1:
   seventhval=-1
else:
   seventhval=1
if "about:blank" in text:
   eighthval=-1
else:
   eighthval=1
if "mail()" or "mailto:" in text:
   ninthval=-1
else:
   ninthval=1
re=text.count("//")
if re>3:
   tenthval=-1
   var10="redirects more than 2"
else:
   tenthval=1

import whois
from datetime import datetime

url=text



"""
try:
   res=whois.whois(url)
   try:
```

```
            a=res['creation_date'][0]
            b=datetime.now()
            c=b-a
            d=c.days
        except:
            a=res['creation_date']
            b=datetime.now()
            c=b-a
            d=c.days
        if d>365:
            eleventhval=1
        else:
            eleventhval=-1
            var11="Domain age working less than a year"
    except:
        aburl=-1
        varab="abnormal url"
        eleventhval=-1
    """
    #code replaced whois
    #
    """try:"""
    d=-1
    try:
        res=whois.whois(url)
    except:
        print("getaddrerrror DNE")
        d=0
        name="Not found in database"
        org="Not found in database"
        add="Not found in database"
        city="Not found in database"
```

```
        state="Not found in database"
        ziip="Not found in database"
        country="Not found in database"
        emails="Not found in database"
        dom="Not Found"
        registrar="Not Found"
    if d!=0:
        try:
            if len(res.creation_date)>1:
                a=res['creation_date'][0]
                b=datetime.now()
                c=b-a
                d=c.days
        except:
            a=res['creation_date']
            b=datetime.now()
            c=b-a
            d=c.days
    """except:
        print("getaddrerrror DNE")
        d=0"""

    if d>365:
        eleventhval=1
        aburl=1
    elif d<=365:
        eleventhval=-1
        aburl=-1
        var11="Domain age working less than a year"

    if aburl==-1:
        twelthval=-1
```

```
        else:
            twelthval=1

        #print (twelthval,eleventhval,aburl,d)
        import urllib.request, sys, re
        import xmltodict, json

        try:
            xml = urllib.request.urlopen('http://data.alexa.com/data?cli=10&dat=s&url={}'.format(text)).read()

            result= xmltodict.parse(xml)

            data = json.dumps(result).replace("@","")
            data_tojson = json.loads(data)
            url = data_tojson["ALEXA"]["SD"][1]["POPULARITY"]["URL"]
            rank= int(data_tojson["ALEXA"]["SD"][1]["POPULARITY"]["TEXT"])
            #print ("rank",rank)
            if rank<=100000:
                thirt=1
            else:
                thirt=-1
                var13="Larger index in alexa database"
            #print (thirt)
        except:
            thirt=-1
            rank=-1
            var13="Larger index in alexa database"
            #print (rank)

        filename = 'phish_trainedv3.sav'

        loaded_model = joblib.load(filename)
```

```
arg=loaded_model.predict(([[oneval,secval,thirdval,fourthval,fifthval,seventhval,eighthval,nin
thval,tenthval,eleventhval,twelthval,thirt]]))
        #print (arg[0])
        import whois
        url=text

        #print (res)
        #res=whois.whois(url)
        if (d!=0):
            name=res.domain_name
            #print (res.domain_name)
            org=res.org
            #print (res.org)
            add=res.address
            #print (res.address)
            city=res.city
            #print (res.city)
            state=res.state
            #print (res.state)
            ziip=res.zipcode
            #print (res.zipcode)
            country=res.country
            #print (res.country)
            emails=res.emails
            #print (res.emails)
            dom=res.domain_name
            #print (res.domain_name)
            registrar=res.registrar
        else:
            name="Not found in database"
            org="Not found in database"
```

```
    add="Not found in database"
    city="Not found in database"
    state="Not found in database"
    ziip="Not found in database"
    country="Not found in database"
    emails="Not found in database"
    dom="Not Found"
    registrar="Not Found"


if dom=="Not Found" and rank==-1 :
    arg[0]=-1
    #phishing

if arg[0]==1:
    te="Legitimate"
else:
    te="Malicious"
if arg[0] == 1:
    mal = True
else:
    mal = False

#print (name,org,add,city,state,ziip,country,emails,dom)


from json.encoder import JSONEncoder
final_entity = { "predicted_argument": [int(arg[0])]}
# directly called encode method of JSON
#print (JSONEncoder().encode(final_entity))
obj = Url()
obj.result = te
```

```
#print (dom,rank)

tags = [name,org,state,add,city,ziip,country,emails,dom,rank]

tags = list(filter(lambda x: x!="Not Found",tags))
tags.append(text)
obj.link = text
obj.add = add
obj.state = state
obj.city = city
#obj.ziip = res['zip_code']
obj.country = country
obj.emails = emails
obj.dom = dom
obj.org = org
obj.rank = rank
obj.registrar=registrar
obj.save()

#print (add)
if add!=None:
    if add and len (add)==1:
        add=add.replace(",","")
    elif len(add)>1:
        add="".join(add)
    #print (add)

name="".join(name)
#print (name)
if emails!=None:
    emails="".join(emails)
if org!=None:
```

```
            org=org.replace(",","")
        #print (org)
        dom="".join(dom)
        #print (dom)
        if registrar:
            registrar=registrar.replace(",","")
        #print (registrar)
        #print (emails)
        #print(city)

        import csv
        with open ('static//dataset.csv','a') as res:
            writer=csv.writer(res)
            s="{},{},{},{},{},{},{},{},{},{},{},{},{}\n".format(text,te,(name),
                org,
                add,
                city,
                state,
                ziip,
                country,emails,
                str(dom),rank,str(registrar))
            res.write(s)

        return render(request,'result.html',{'result':'Real-time analysis
successfull','f2':te,'mal': mal,'text':text,'name':name,
                'org':org,
                'add':add,
                'city':city,
                'state':state,
                'ziip':ziip,
                'country':country,'emails':emails,
```

```
'dom':dom,'rank':rank,'registrar':registrar,"tags":tags,"var13":var13,"varab":varab,"var11":var
11,"var10":var10,"var5":var5,"var4":var4,"var3":var3})


        else:
            return render(request,'404.html')
        """except:
        return render(request,'errorpage.html')  """



def api(request):
    try:
        text=request.GET['query']
        import datetime

        if text.startswith('https://malicious-url-detectorv5.herokuapp.com/'):
            import datetime
            mydict = {
                "query" : text,
                "malware" : False,
                "datetime" : str(datetime.datetime.now())
            }
            response = JsonResponse(mydict)
            return response



        elif text.startswith('https://www.youtube.com/'):
            import datetime
            mydict = {
                "query" : text,
                "malware" : False,
                "datetime" : str(datetime.datetime.now())
            }
```

```
        response = JsonResponse(mydict)
        return response

    elif text.startswith('https://www.google.com/search?q='):
        import datetime
        mydict = {
            "query" : text,
            "malware" : False,
            "datetime" : str(datetime.datetime.now())
        }
        response = JsonResponse(mydict)
        return response


    #if (text.startswith('https://www.google.com/search?q=')==False) :
    else:

        if text.startswith('https://') or text.startswith('http://'):

            if len(text)<=9:
                return render(request,'errorpage.html')

            try:
                res=whois.whois(url)
            except:
                print("getaddrerrror DNE")
                d=0
                name="Not found in database"
                org="Not found in database"
                add="Not found in database"
                city="Not found in database"
                state="Not found in database"
```

```
        ziip="Not found in database"
        country="Not found in database"
        emails="Not found in database"
        dom="Not Found"
    if d!=0:
        try:
            if len(res.creation_date)>1:
                a=res['creation_date'][0]
                b=datetime.now()
                c=b-a
                d=c.days
        except:
            a=res['creation_date']
            b=datetime.now()
            c=b-a
            d=c.days
    """except:
        print("getaddrerrror DNE")
        d=0"""



    if d>365:
        eleventhval=1
        aburl=1
    elif d<=365:
        eleventhval=-1
        aburl=-1
        var11="Domain age working less than a year"

    if aburl==-1:
        twelthval=-1
```

```
        else:
            twelthval=1
        import urllib.request, sys, re
        import xmltodict, json

        try:
            xml = urllib.request.urlopen('http://data.alexa.com/data?cli=10&dat=s&url={}'.format(text)).read()

            result= xmltodict.parse(xml)

            data = json.dumps(result).replace("@","")
            data_tojson = json.loads(data)
            url = data_tojson["ALEXA"]["SD"][1]["POPULARITY"]["URL"]
            rank= int(data_tojson["ALEXA"]["SD"][1]["POPULARITY"]["TEXT"])
            #print ("rank",rank)
            if rank<=100000:
                thirt=1
            else:
                thirt=-1
            #print (thirt)
        except:
            thirt=-1
            rank="Not Indexed by Alexa"
            #print (rank)

        filename = 'phish_trainedv3.sav'

        loaded_model = joblib.load(filename)

arg=loaded_model.predict(([[oneval,secval,thirdval,fourthval,fifthval,seventhval,eighthval,nin
thval,tenthval,eleventhval,twelthval,thirt]]))
```

```
def getdataset(request):
    try:
        return render(request,'getdataset.html')
    except:
        return render(request,'404.html')
```

7.1.2: Source Code for Admin:

```
from django.contrib import admin
from .models import *

# Register your models here.
admin.site.register(UserFeedBack)
admin.site.register(Url)
```

7.1.3: Souce Code for Apps:

```
from django.apps import AppConfig

class MyappConfig(AppConfig):
    name = 'myapp'
```

7.1.4: Source Code for Models:

```
from django.db import models

# Create your models here.
class UserFeedBack(models.Model):
    userid = models.AutoField(primary_key=True)
    title = models.CharField(max_length=100)
    description = models.TextField()
```

```
    reply = models.TextField()
    replied = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)


class Url(models.Model):
    urlid = models.AutoField(primary_key=True)
    link = models.CharField(max_length=1000,null=True,default="Not Found")
    result = models.CharField(max_length=100,null=True,default="Not Found")
    add = models.CharField(max_length=1000,null=True,default="Not Found")
    org = models.CharField(max_length=100,null=True,default="Not Found")
    city = models.CharField(max_length=100,null=True,default="Not Found")
    state = models.CharField(max_length=100,null=True,default="Not Found")
    country = models.CharField(max_length=100,null=True,default="Not Found")
    dom = models.CharField(max_length=100,null=True,default="Not Found")
    emails = models.CharField(max_length=100,null=True,default="Not Found")
    rank = models.IntegerField(null=True,default=0,blank=True)
    registrar = models.CharField(max_length=100,null=True,default="Not Found")
    #rank = models.CharField(max_length=100,null=True,default="N A",blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
```

7.1.5: Source Code for Tests:

```
from django.test import TestCase

# Create your tests here.
```

7.1.6: Source Code for Urls:

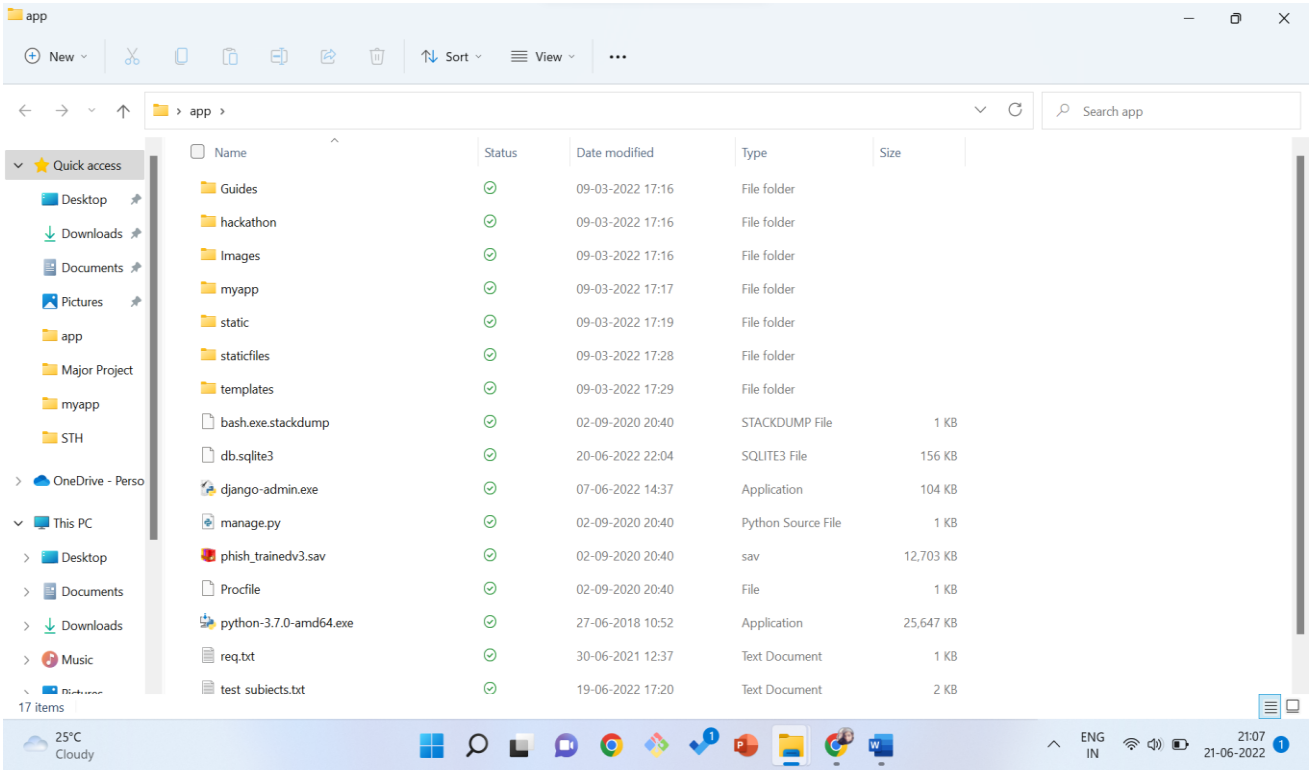
```
from django.urls import path
from . import views

urlpatterns = [
```

```
    path('',views.index,name='index'),
    path('getuserfeedbackform',views.getuserfeedbackform,name="getuserfeedbackform"),
    path('saveuserfeedbackform',views.saveuserfeedbackform,name="saveuserfeedbackform"),
    path('api',views.api,name='api'),
    path('search',views.search,name="search"),
    path('result',views.result,name='result'),
    path('about',views.about,name='about'),
    path('geturlhistory',views.geturlhistory,name="geturlhistory"),
    path('discuss',views.discuss,name="discuss"),
    path('reply/<int:replyid>',views.replyform,name="reply"),
    path('savereply',views.savereply,name="reply"),
    path('searchdiscuss',views.searchdiscuss,name="searchdiscuss"),
    path('getdataset',views.getdataset,name='getdataset'),
    path('getdocs',views.getdoc,name='namedoc')
]
```

8. Screenshots

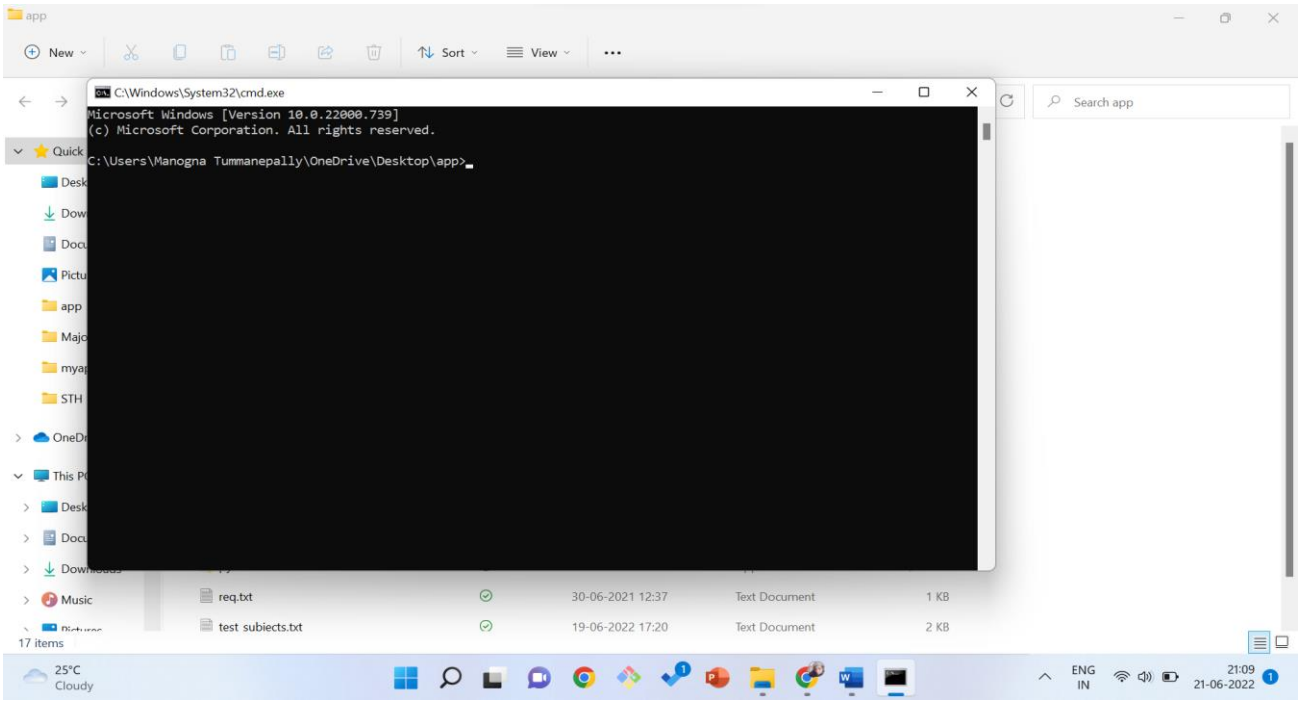
8. Screenshots

8.1 Apps Folder:



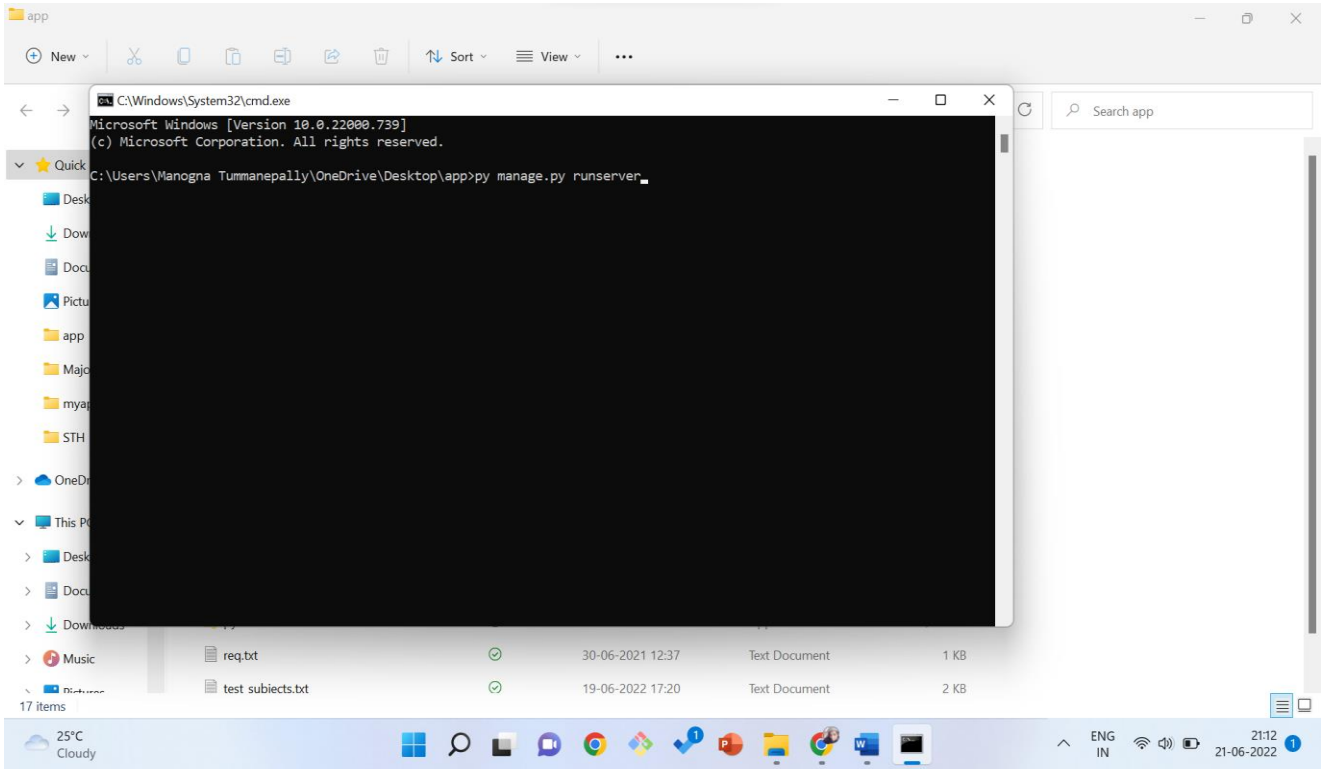
Screenshot 8.1: Apps Folder consisting of Project

8.2: Command Prompt of Apps:



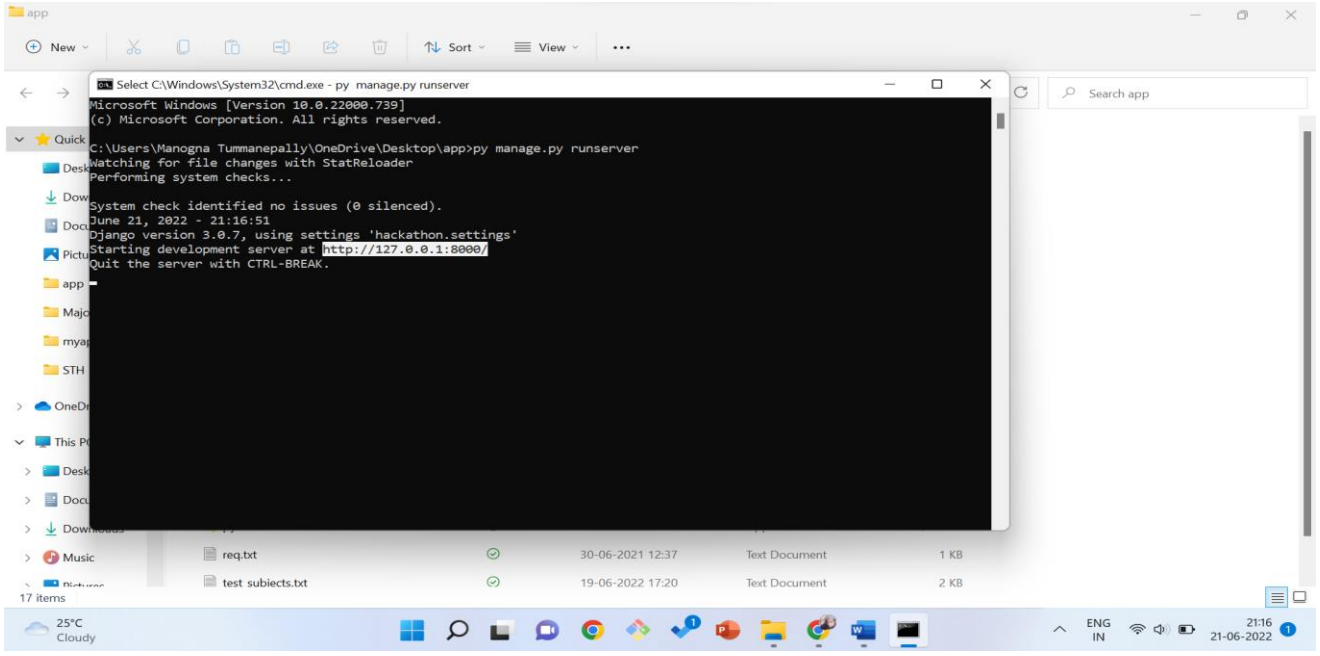
Screenshot 8.2: Command Prompt of Apps

8.3: Command to start the server:



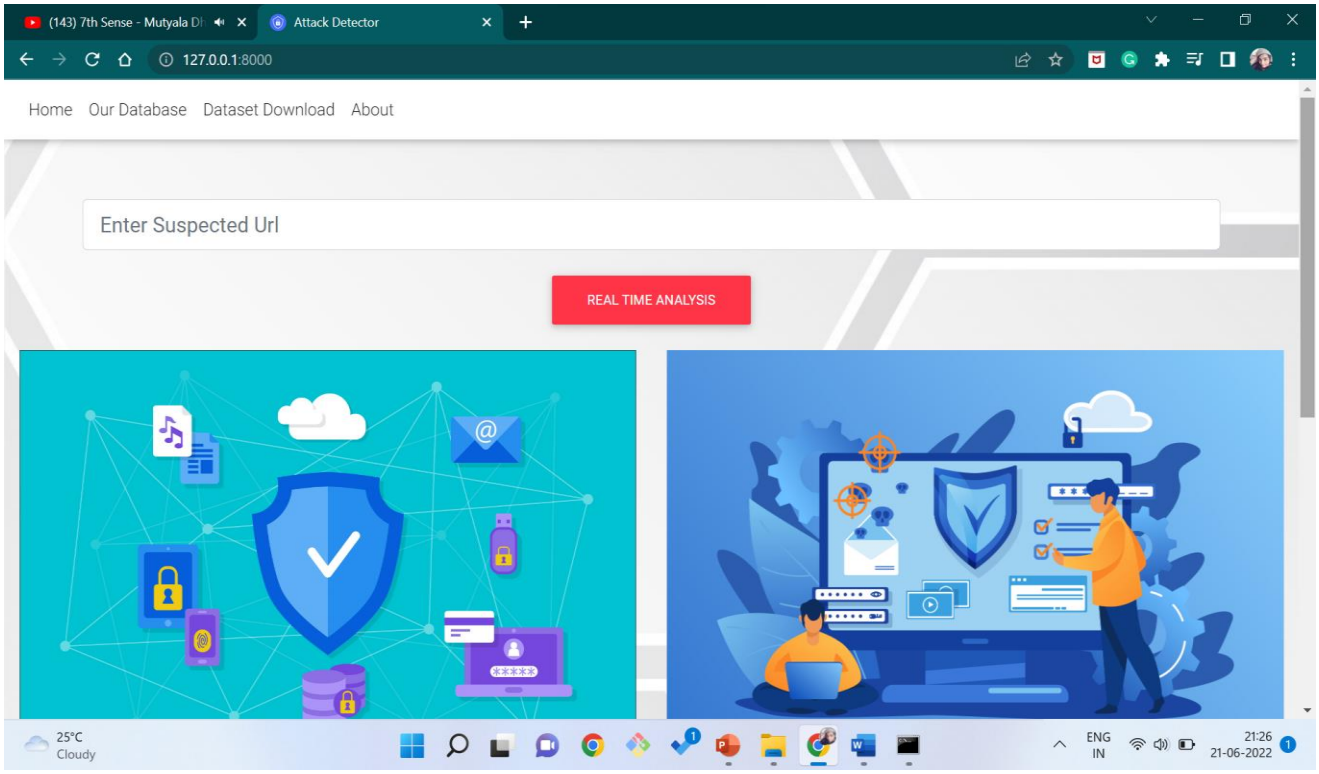
Screenshot 8.3: Command to start the server

8.4: Local Host Address Generated:



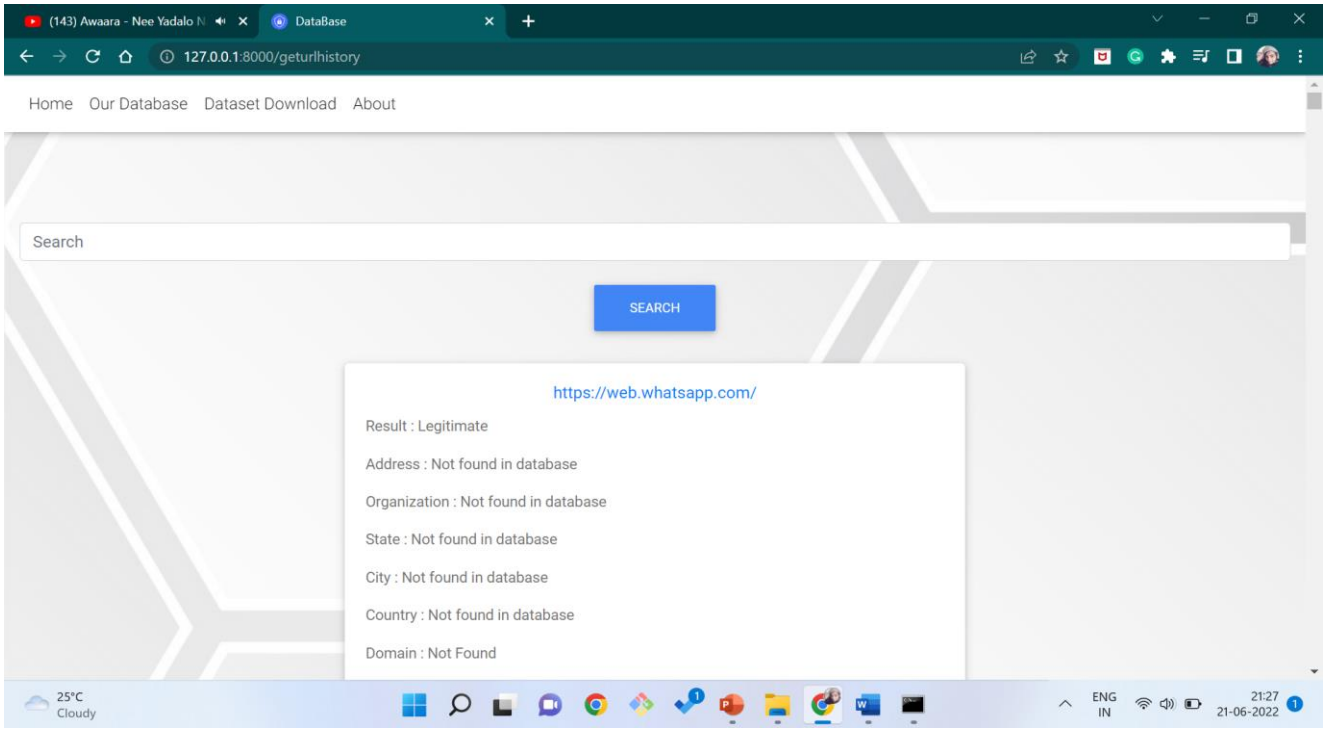
Screenshot 8.4: Local Host Address Generated

8.5: The Website:



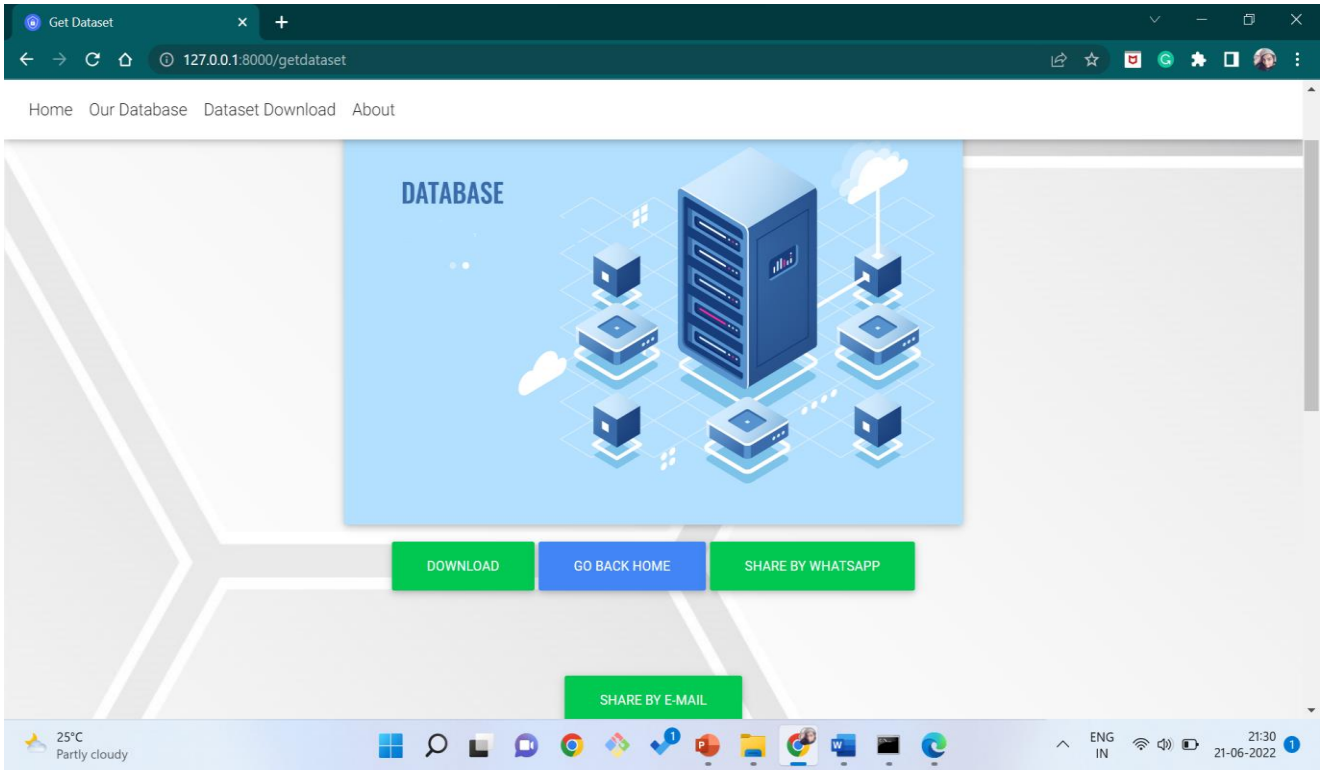
Screenshot 8.5: The Website

8.6: Search History of Website:



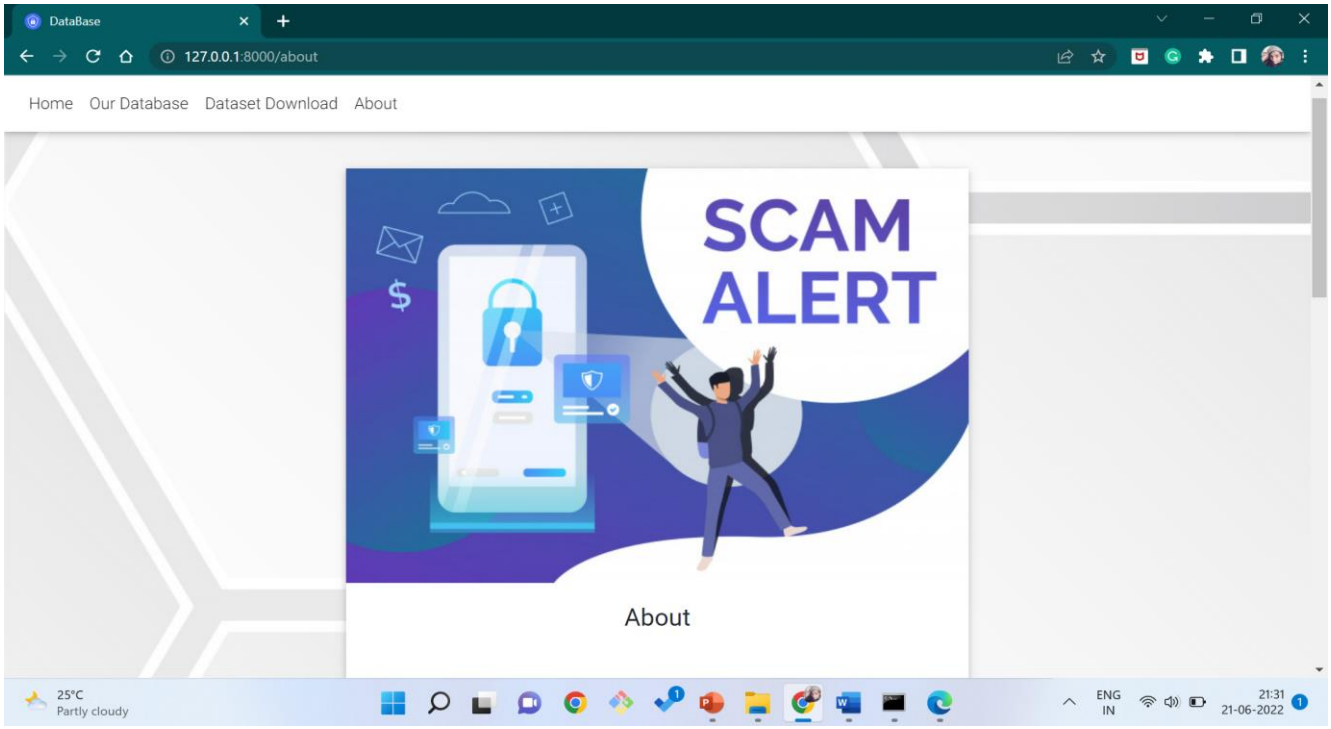
Screenshot 8.6: Search History of Website

8.7: Dataset Download:



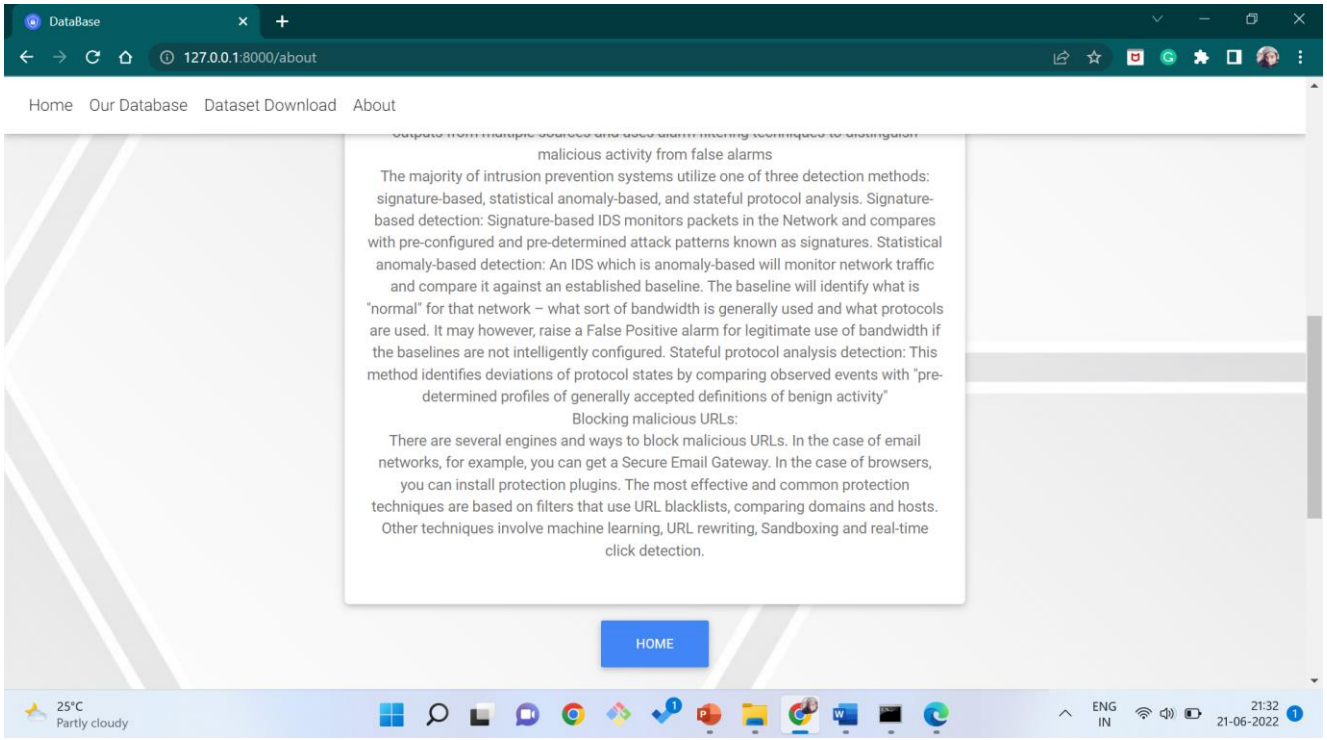
Screenshot 8.7: Dataset Download

8.8: About the Website:



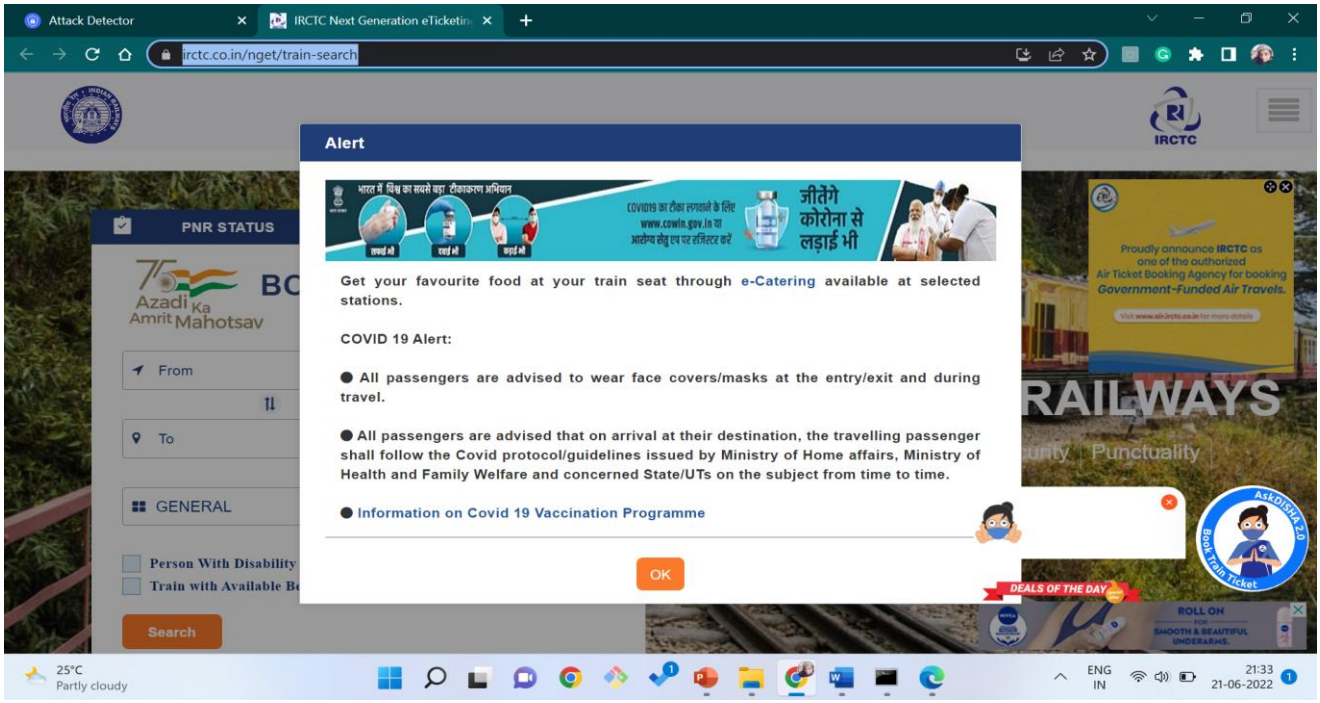
Screenshot 8.8: About the Website

8.9: Home Button to Redirect:



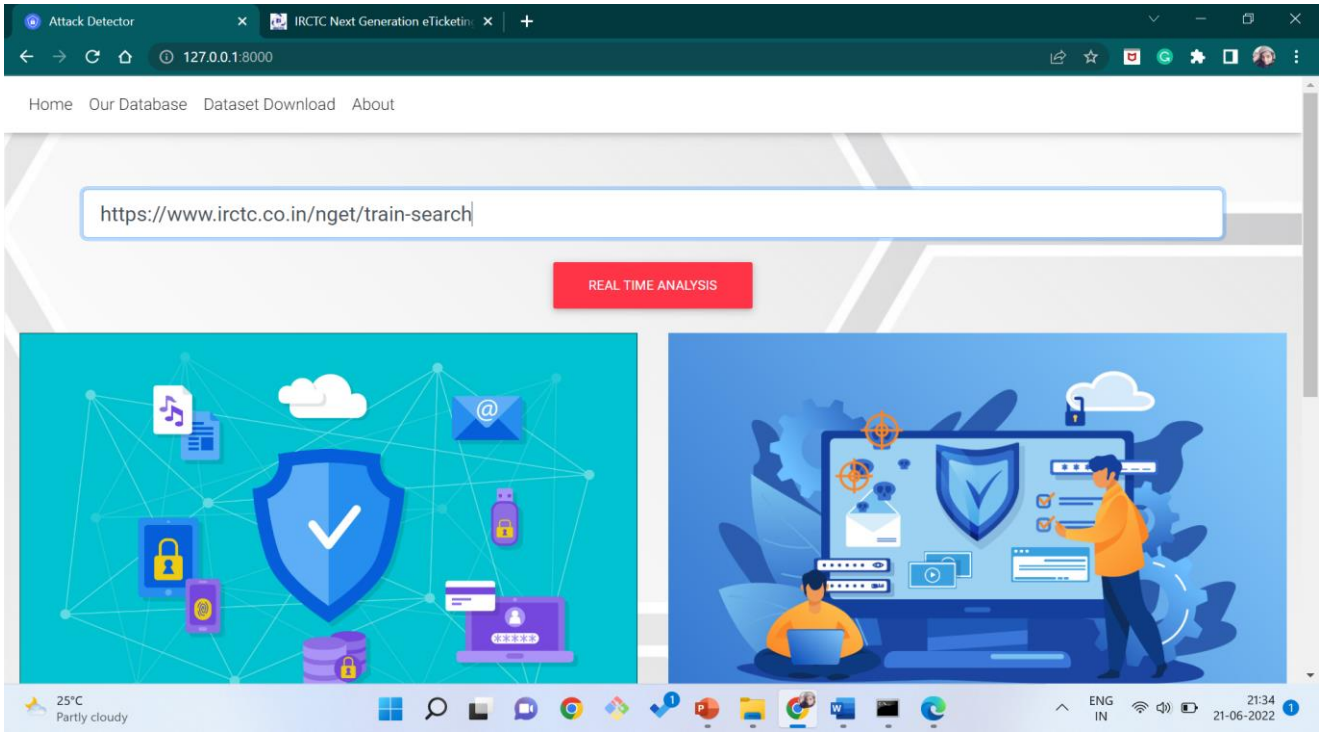
Screenshot 8.9: Home Button to Redirect

8.10: IRCTC Train Booking Website Url:



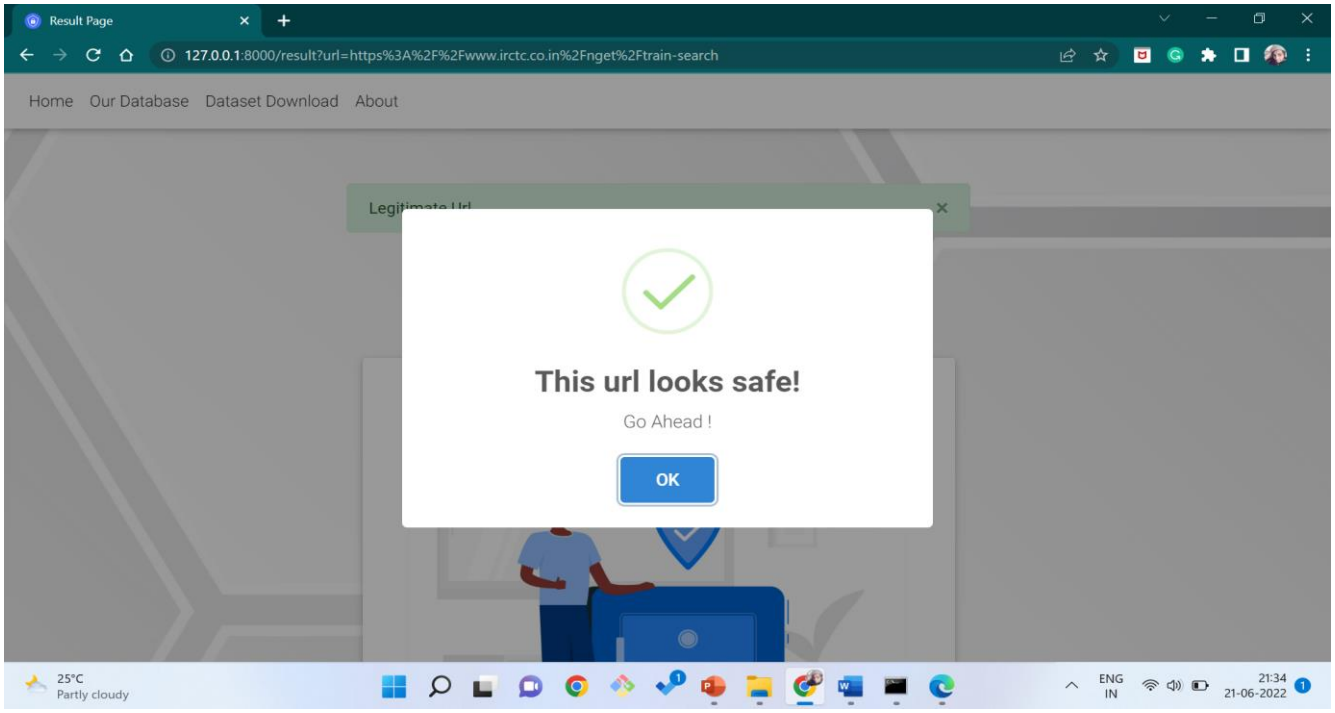
Screenshot 8.10: IRCTC Train Booking Website Url

8.11: Enter the Copies URL in the search space:



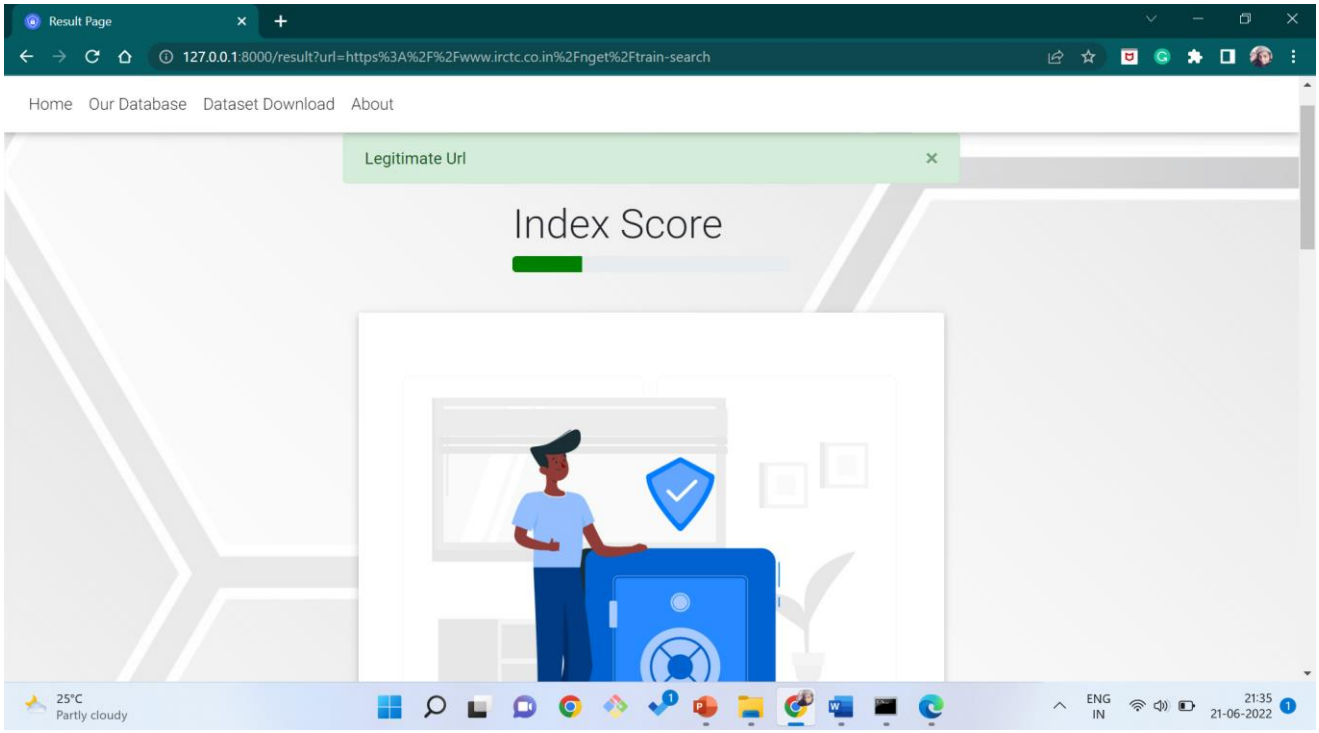
Screenshot 8.11: Enter the Copies URL in the search space

8.12: Real-Time Analysis of the Url:



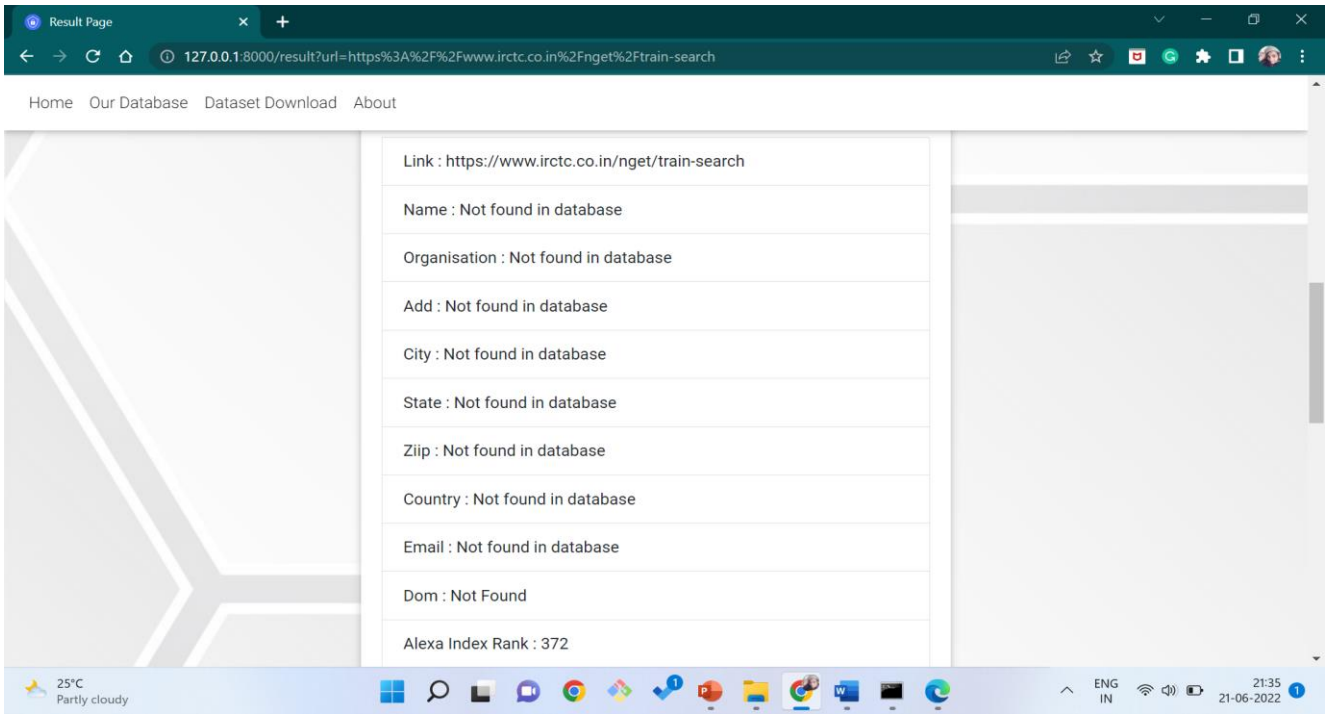
Screenshot 8.12: Real-Time Analysis of the Url

8.13: Real-Time Analysis of the Url:



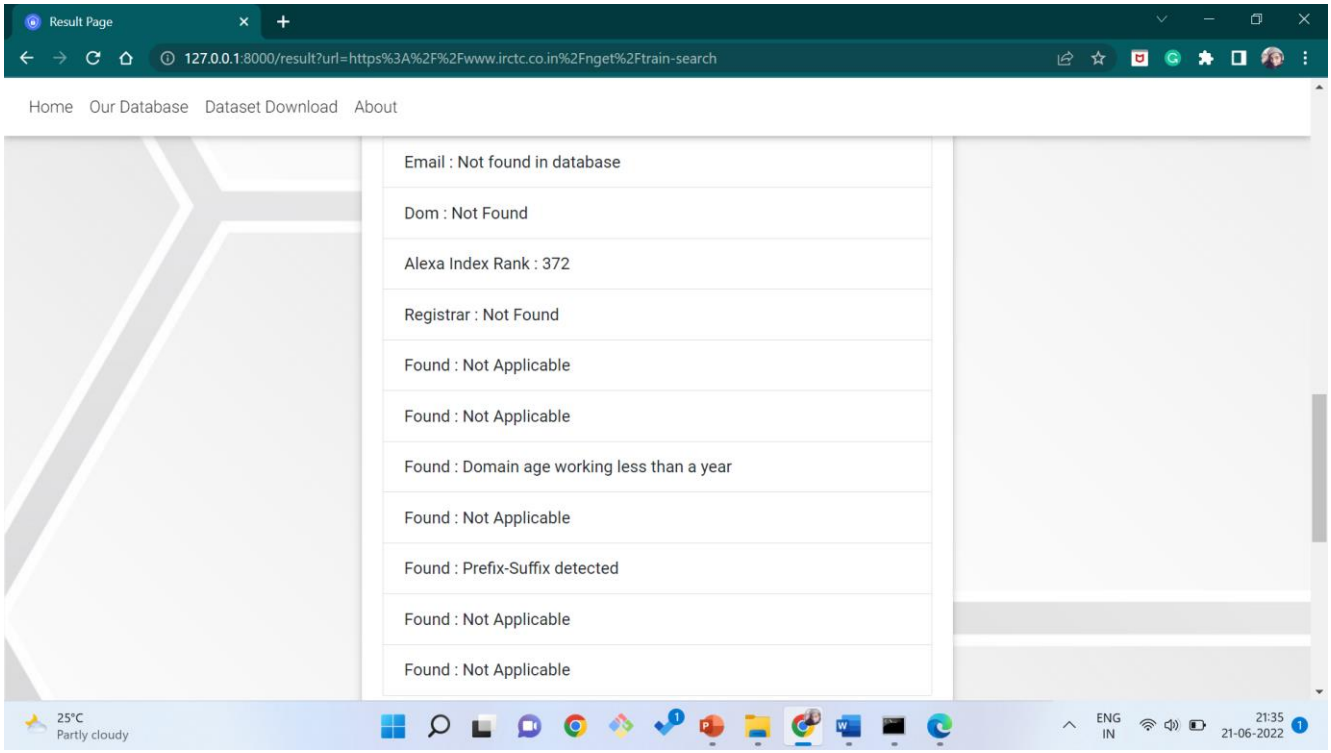
Screenshot 8.13: Real-Time Analysis of the Url

8.14: Attributes of the Url:



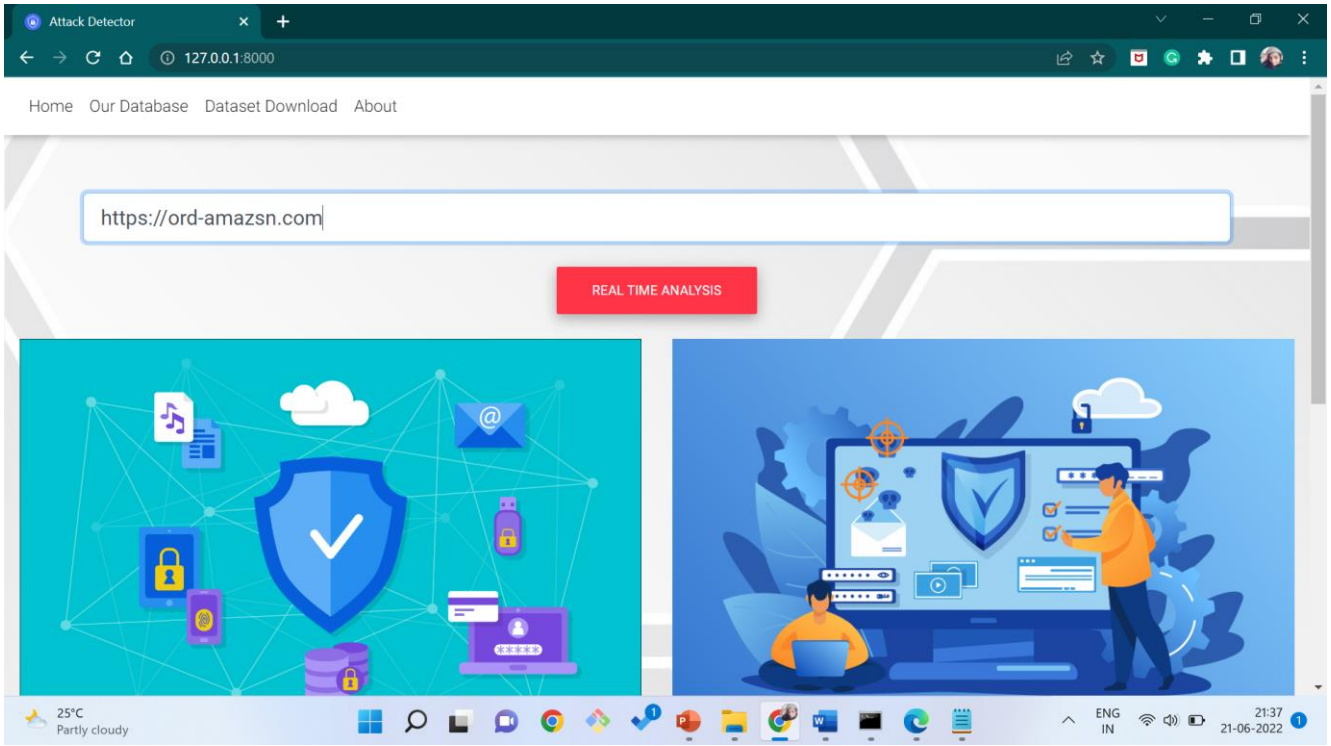
Screenshot 8.14: Attributes of the Url

8.15: Attributes of the Url:



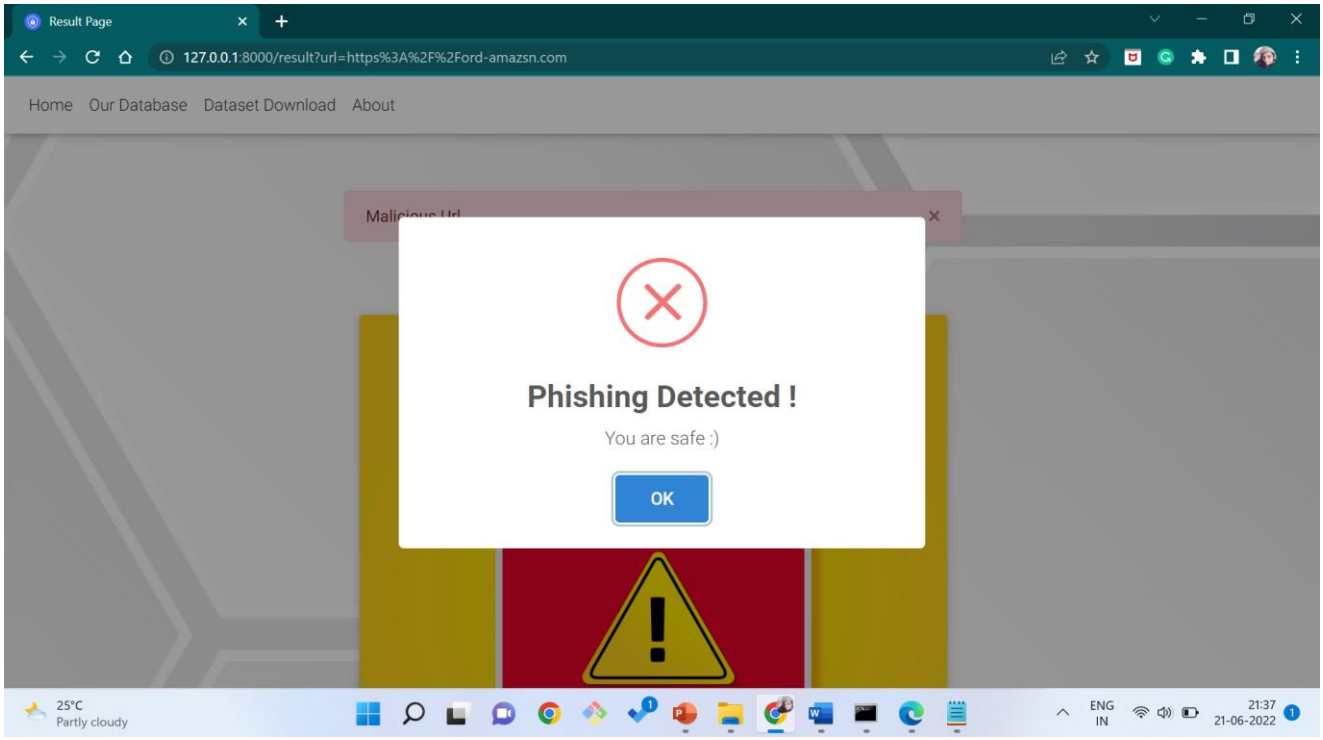
Screenshot 8.15: Attributes of the Url

8.16: Malicious Url link:



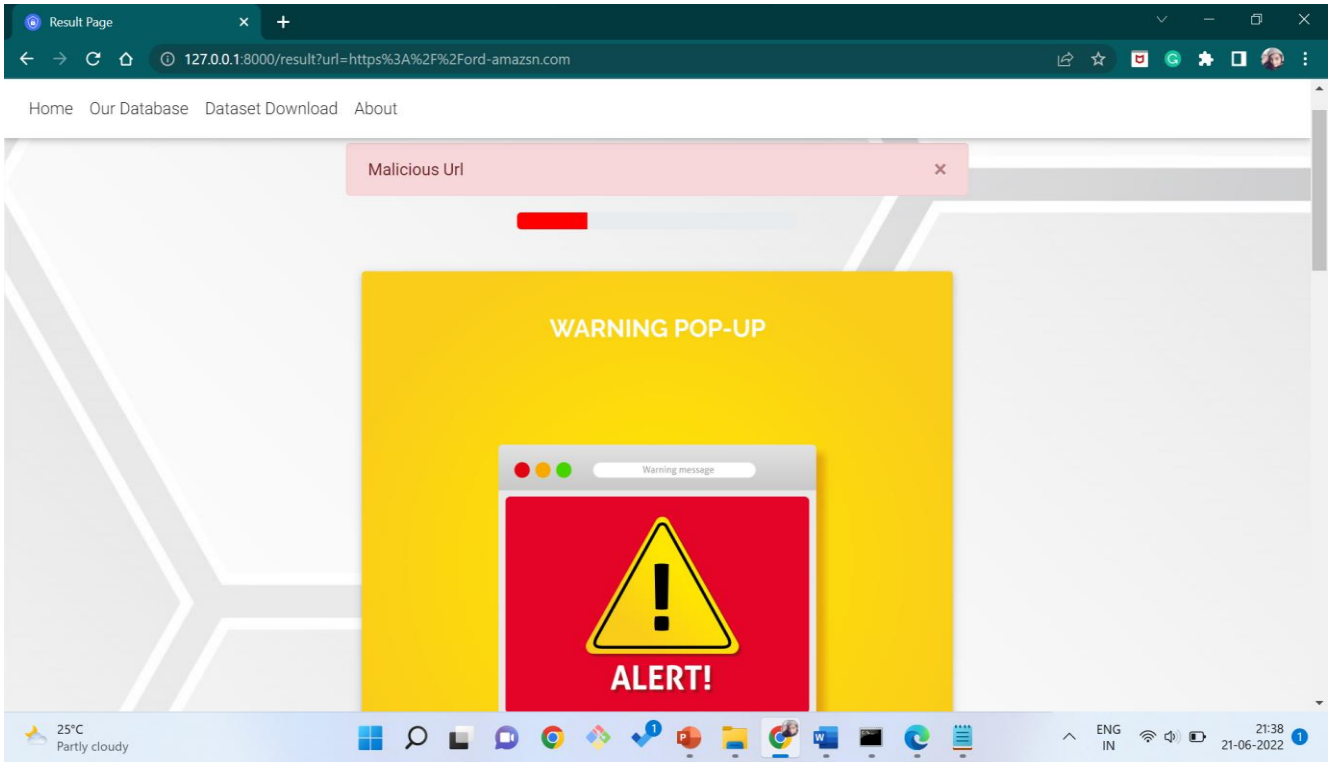
Screenshot 8.16: Malicious Url link

8.17: Pop-up Window after Real-Time Analysis:



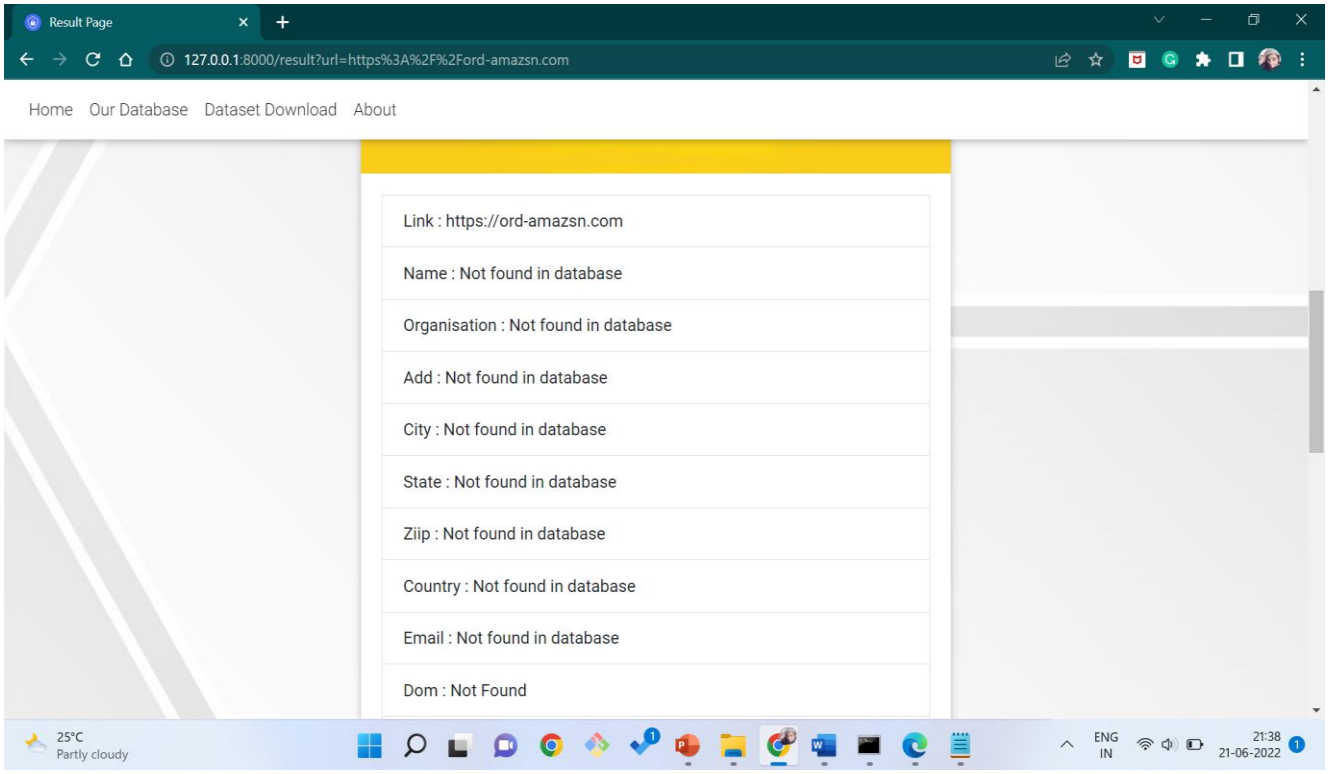
Screenshot 8.17: Pop-up Window after Real-Time Analysis

8.18: Result of the Real-Time Analysis:



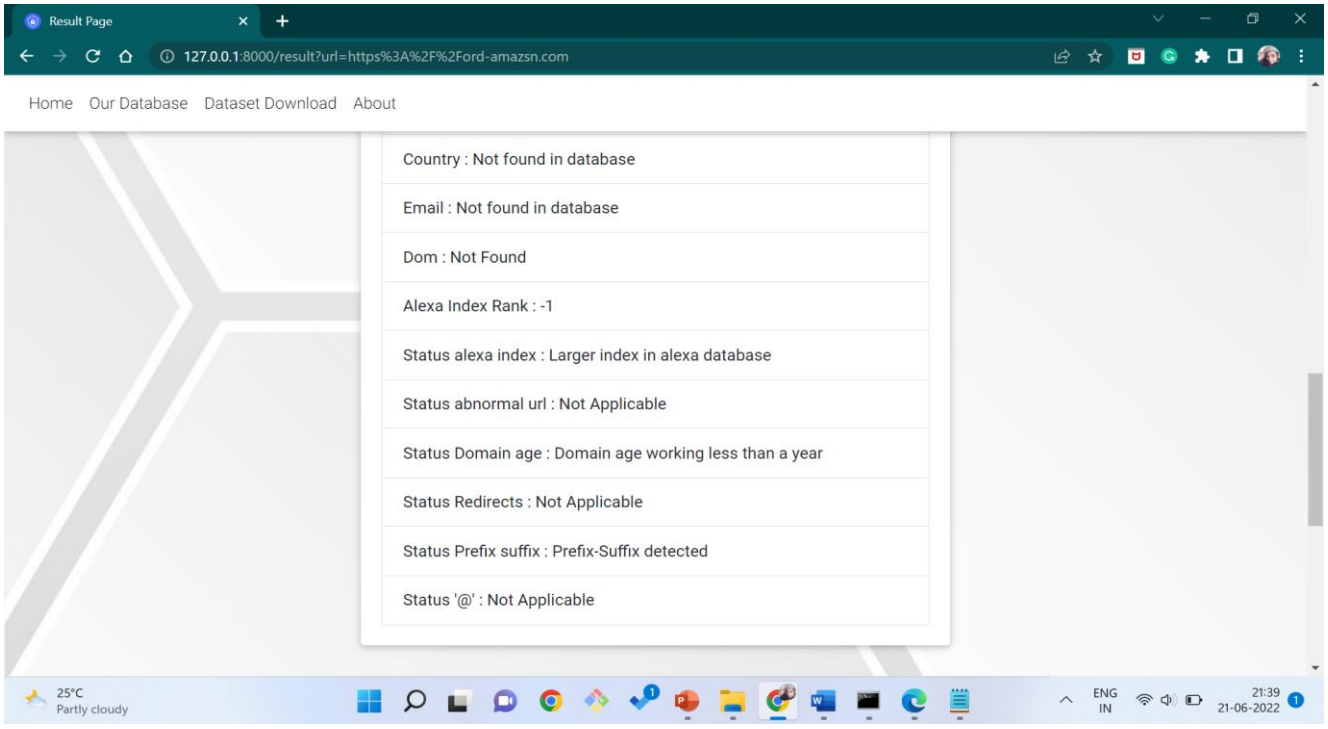
Screenshot 8.18: Result of the Real-Time Analysis

8.19: Attributes of the Malicious Url:



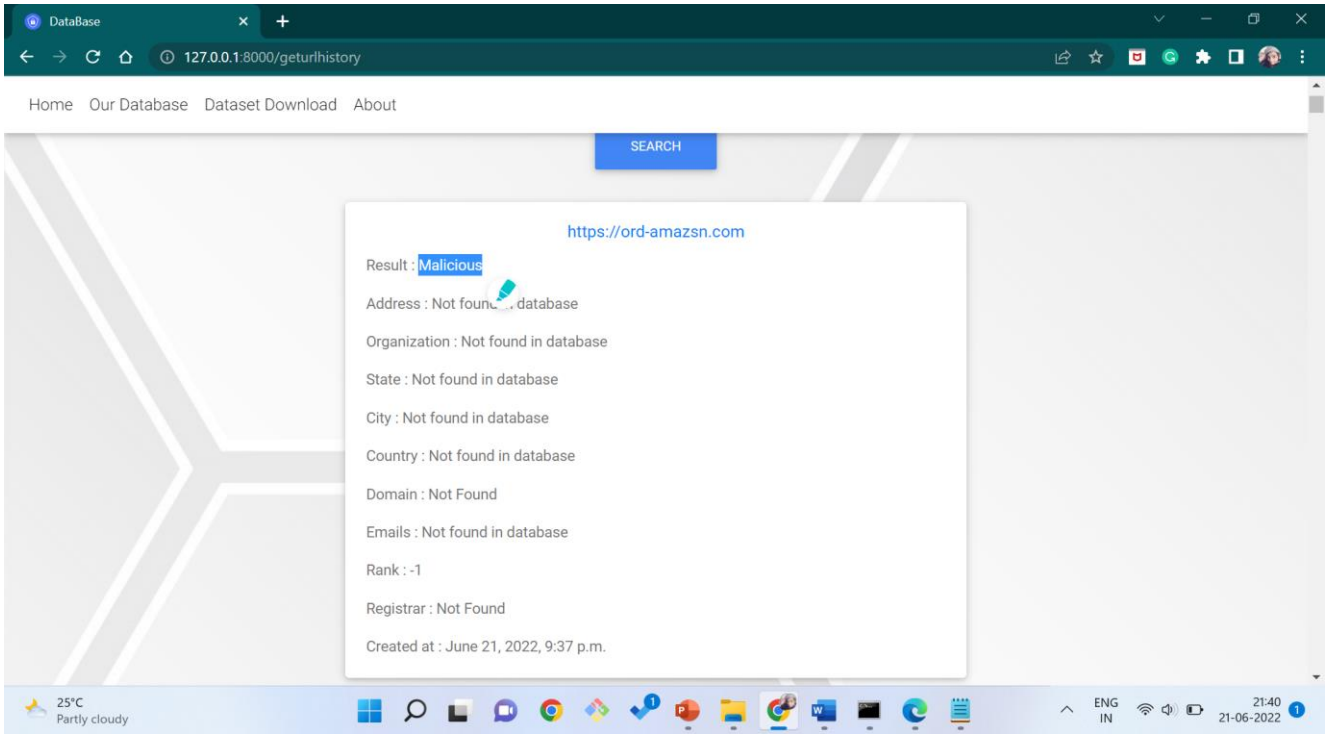
Screenshot 8.19: Attributes of the Malicious Url

8.20: Attributes of the Malicious Url:



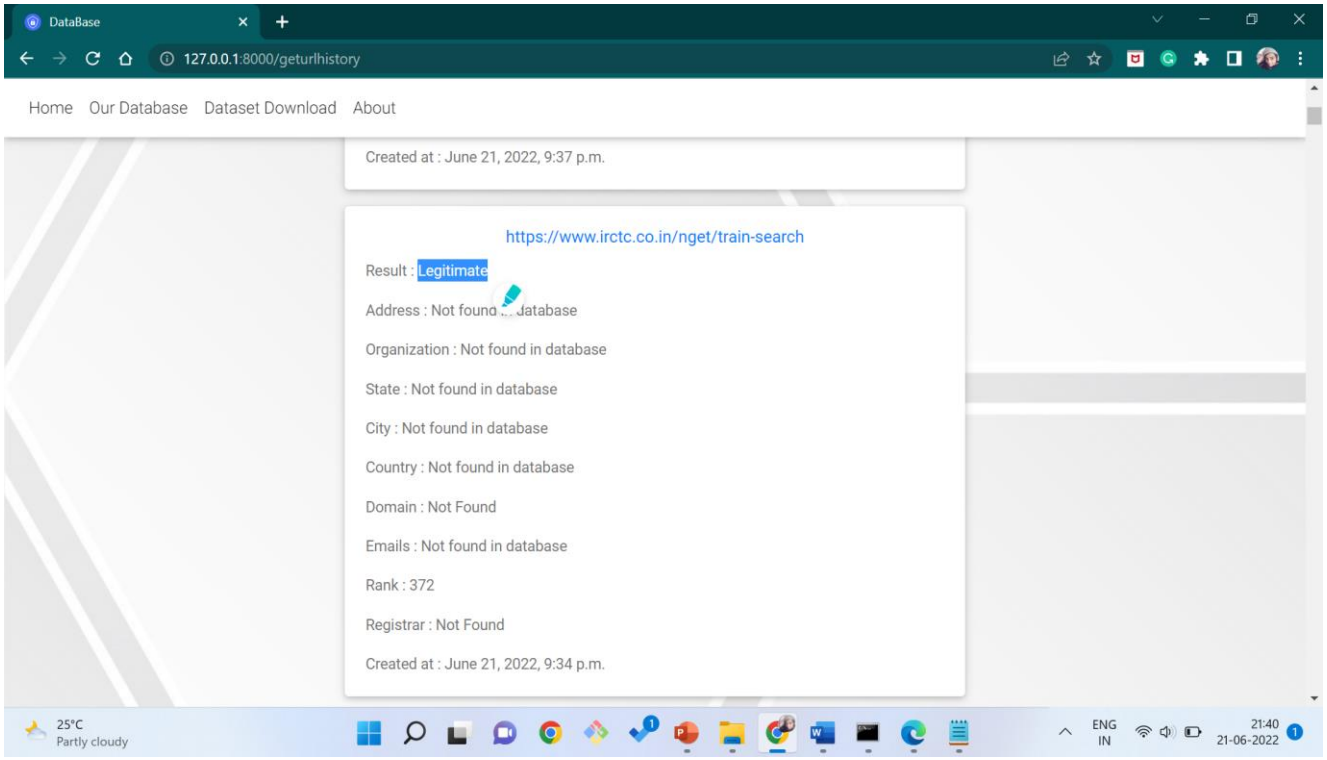
Screenshot 8.20: Attributes of the Malicious Url

8.21: Search History of Previous URLs:



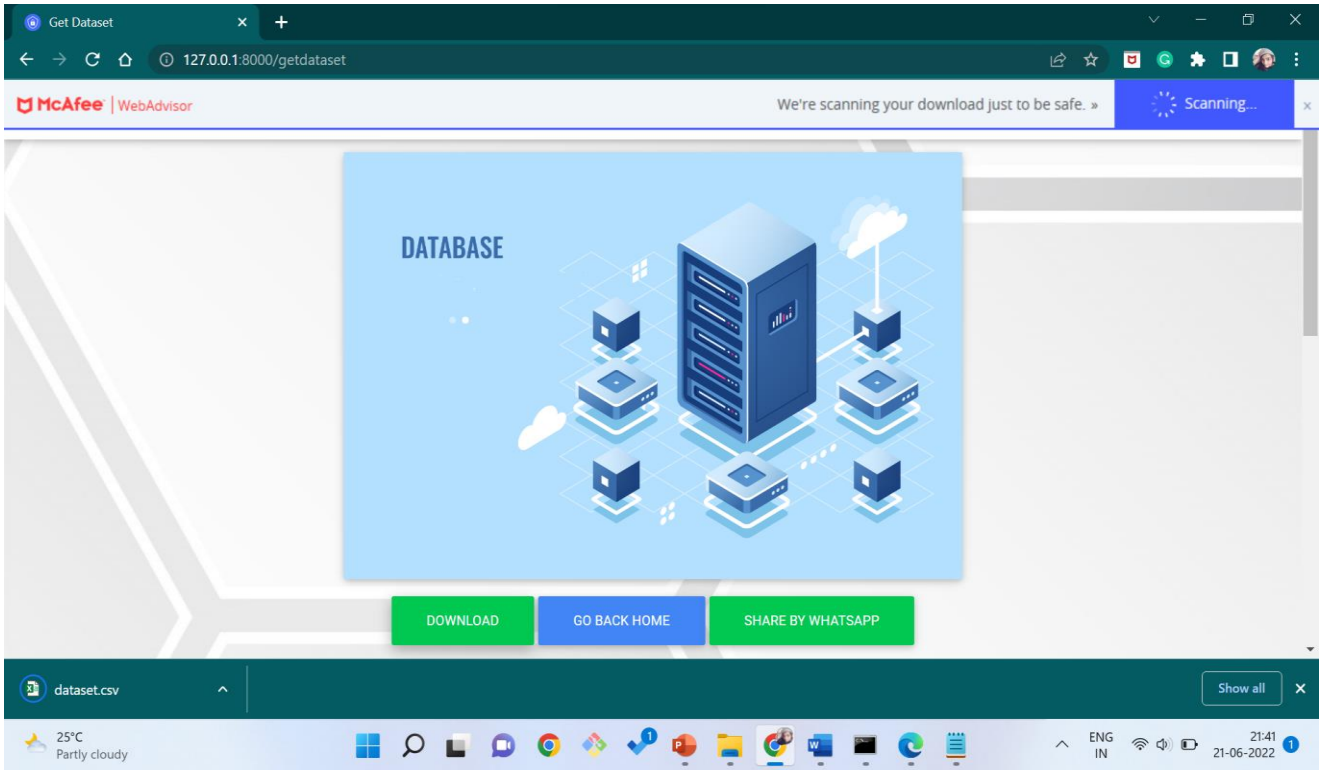
Screenshot 8.21: Search History of Previous URLs

8.22: Search History of Previous URLs:



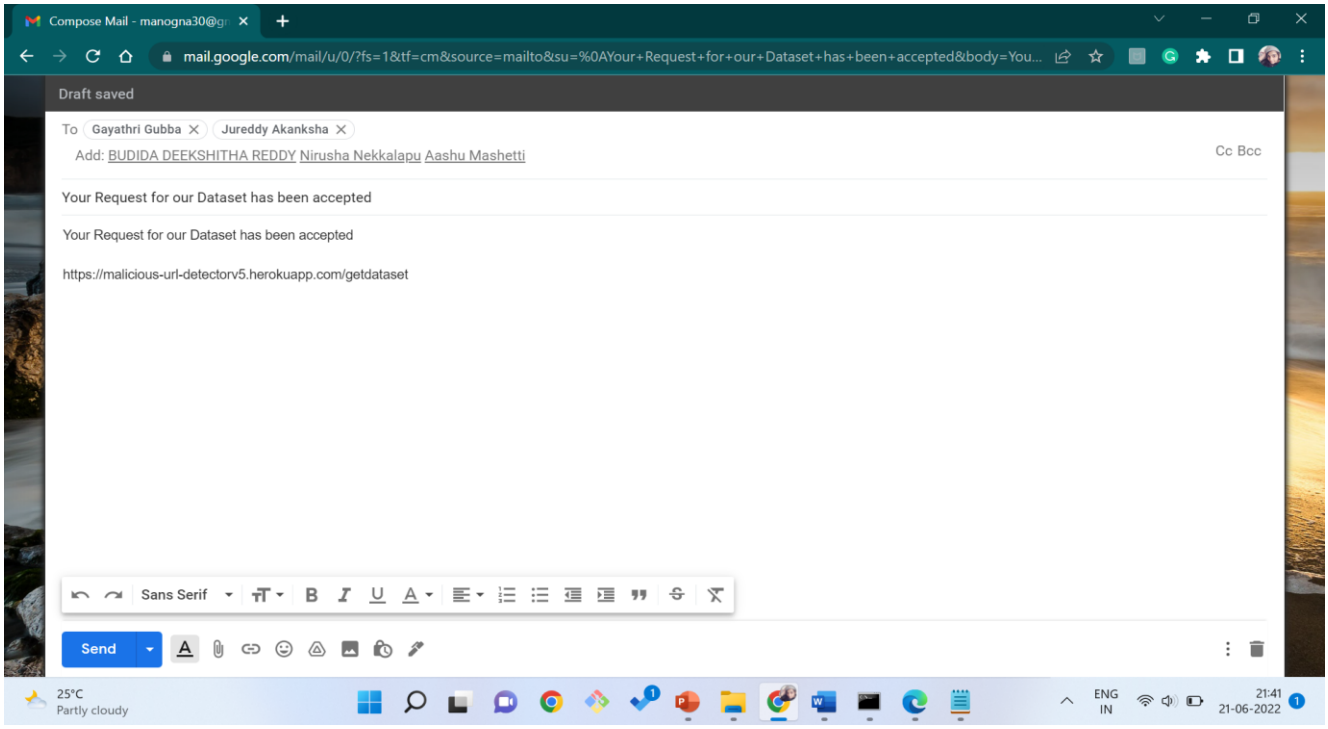
Screenshot 8.22: Search History of Previous URLs

8.23: Download the Dataset from the Website:



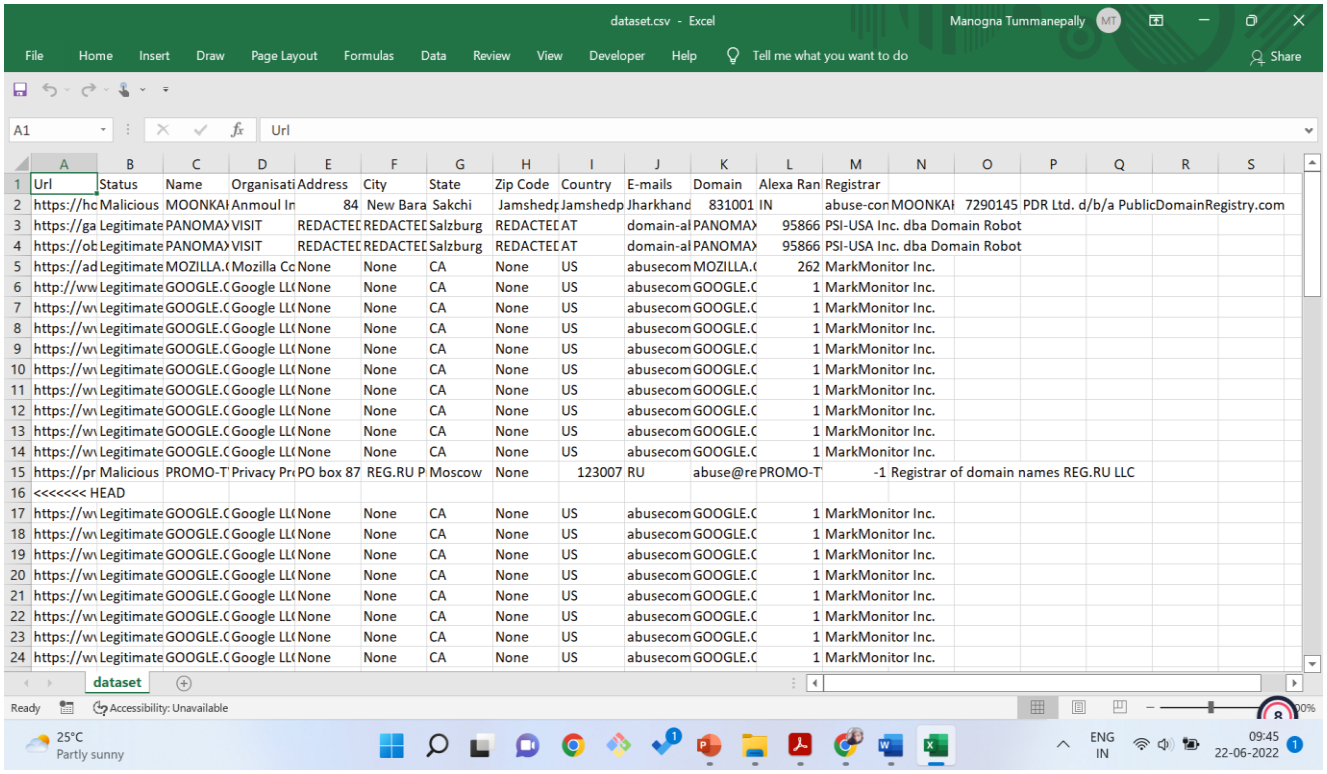
Screenshot 8.23: Download the Dataset from the Website

8.24: Sharing of Dataset through Email option:



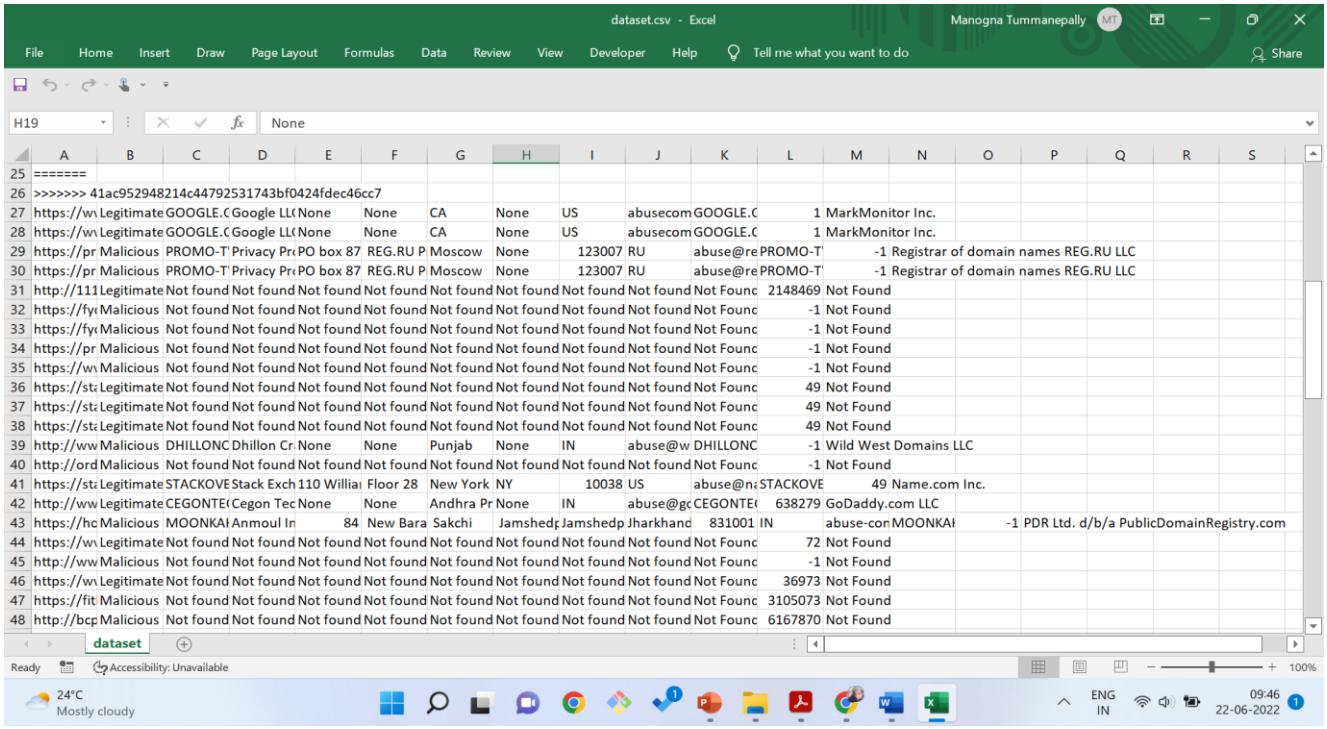
Screenshot 8.24: Sharing of Dataset through Email option

8.25: NSL-KDD Dataset:



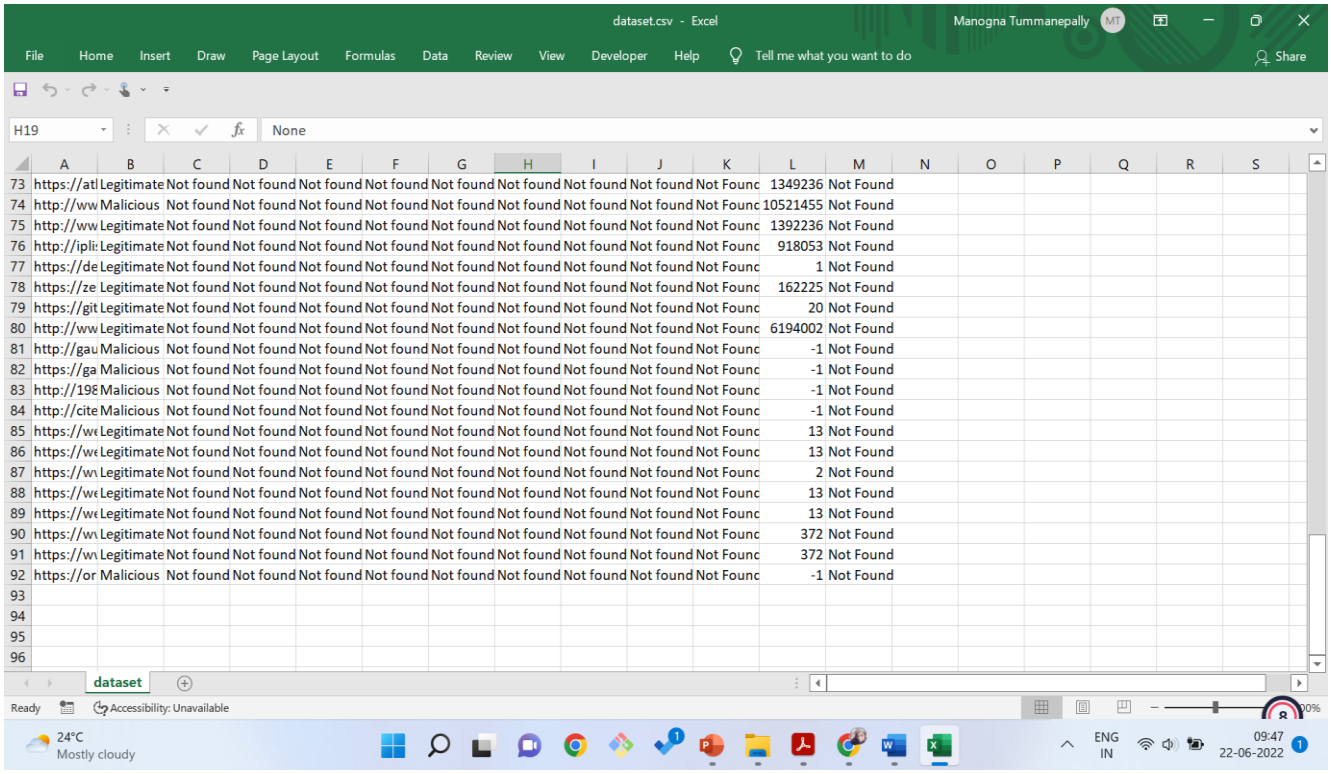
Screenshot 8.25: NSL-KDD Dataset

8.26: NSL-KDD Dataset:



Screenshot 8.26: NSL-KDD Dataset

8.27: NSL-KDD Dataset:



	A	B	C	D	E	F	G	H	I	J	K	L	M
73	https://atl	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	1349236	Not Found
74	http://ww	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	10521455	Not Found
75	http://ww	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	1392236	Not Found
76	http://ipli	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	918053	Not Found
77	https://de	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	1	Not Found
78	https://ze	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	162225	Not Found
79	https://git	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	20	Not Found
80	http://ww	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	6194002	Not Found
81	http://gau	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	-1	Not Found
82	https://ga	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	-1	Not Found
83	http://198	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	-1	Not Found
84	http://cite	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	-1	Not Found
85	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	13	Not Found
86	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	13	Not Found
87	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	2	Not Found
88	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	13	Not Found
89	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	13	Not Found
90	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	372	Not Found
91	https://w	Legitimate	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	372	Not Found
92	https://or	Malicious	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	Not found	-1	Not Found
93													
94													
95													
96													

Screenshot 8.27: NSL-KDD Dataset

9. Testing

9. Testing

Testing is the process of executing a program with the aim of finding errors. To make our software perform well it should be error-free. If testing is done successfully, it will remove all the errors from the software.

9.1: Types of Testing:

- White Box Testing
- Black Box Testing
- Unit testing
- Integration Testing
- Alpha Testing
- Beta Testing
- Performance Testing and so on

9.1.1: White Box Testing

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, and conditions. It is performed by software developers.

9.1.2: Black Box Testing

A method of software testing that verifies the functionality of an application without having specific knowledge of the application's code/internal structure. Tests are based on requirements and functionality.

9.1.3: Unit Testing

Software verification and validation method in which a programmer tests if individual units of source code are fit for use. It is usually conducted by the development team.

9.1.4: Integration Testing

The phase-in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

9.1.5: Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually, it is performed by the end-users.

9.1.6: Beta Testing

Final testing before releasing the application for commercial purposes. It is typically done by end-users or others.

9.1.7: Performance Testing

Functional testing is conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

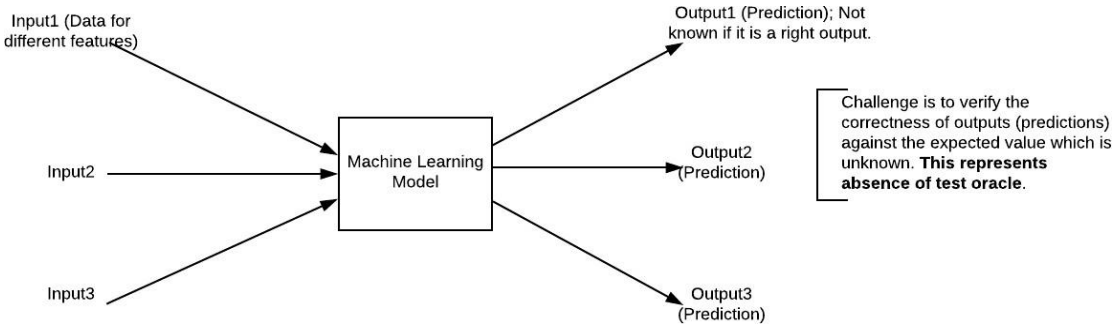
9.1.8: Black Box Testing

Blackbox testing is testing the functionality of an application without knowing the details of its implementation including internal program structure, data structures, etc. Test cases for black-box testing is created based on the requirement specifications. Therefore, it is also called specification-based testing. Fig. represents the black box testing:

Fig.9.1.8: Black Box Testing

When applied to machine learning models, black-box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model, etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.



The above Fig.9.1.9 represents the black box testing procedure for machine learning algorithms.

Fig.9.1.9: Black Box Testing for Machine Learning algorithms

9.2: Test Cases:

S. No	Test Case	Input	Expected Output	Actual Output	Result
1	Real Time Analysis	URL starting with https or http	Short window displaying that url is safe or Phishing detected	Short window displaying that url is safe or Phishing detected	Pass
2	Real Time Analysis	URL not starting with https or http	Displays 404 Error message	Displays 404 Error message	Pass

Table 9.2: Testcases

9.3: Validations:

S. No	Field Name	Validation	Message
1	URL	Enter the URL starting with either http or https followed by :// and must contain www. And then followed by subdomain of length(2,256) & last part containing top level domain like .com, .org etc	Please fill out the field

Table 9.3: Validation

10. Conclusion

10. Conclusion

We propose an end-to-end deep learning model BAT-MC that is composed of BLSTM and attention mechanisms. BAT-MC can well solve the problem of intrusion detection and provide a new research method for intrusion detection. Also introduced is the attention mechanism into the BLSTM model to highlight the key input. The attention mechanism conducts feature learning on sequential data composed of data package vectors. The obtained feature information is reasonable and accurate. We compare the performance of BAT-MC with traditional deep learning methods, the BAT-MC model can extract information from each packet. By making full use of the structure information of network traffic, the BAT-MC model can capture features more comprehensively. We evaluate our proposed network with a real NSL-KDD dataset. The output shows whether the given URL is legitimate or malicious.

11. Future Scope

11. Future Scope

This project only checks whether the given URL is malicious or not. It can be modified by adding a learn more option which redirects it to the requested website when clicked on it to know the information in it. We can also extend this project by adding an alert window that gives a warning with two options "Proceed to Continue" or "Back to Safety" when phishing is detected**.**

12. References

12. References

[1] D. Giusto, A. Iera, G. Morabito, L. Atzori (Eds.), The Internet of Things, 1661 Springer, 2010. ISBN: 978-1-4419-1673-0.

[2] National Intelligence Council, Disruptive Civil Technologies – Six 1663 Technologies with Potential Impacts on US Interests Out to 2025 – 1664 Conference Report CR 2008-07, April 2008, .

[3] INFSO D.4 Networked Enterprise & RFID INFSO G.2 Micro & Nanosystems, in: Co-operation with the Working Group RFID of the ETP EPOSS, Internet of Things in 2020, Roadmap for the Future, Version 1.1, 27 May 2008

[4] Auto-Id Labs, http://www.autoidlabs.org/

[5] The EPCglobal Architecture Framework, EPCglobal Final Version 1.3, 1673 Approved 19 March 2009,www.epcglobalinc.org.

[6] K. Sakamura, Challenges in the age of ubiquitous computing: a case 1675 study of T-engine – an open development platform for embedded 1676 systems, in: Proceedings of ICSE'06, Shanghai, China, May 2006.

[7] L. Srivastava, Pervasive, ambient, ubiquitous: the magic of radio, in: 1715 European Commission Conference ''From RFID to the Internet of 1716 Things", Bruxelles, Belgium, March 2006.

[8] K. Finkenzeller, RFID Handbook, Wiley, 2003.

[9] B. B. Zarpelo, R. S Miani, C. T. Kawakani, and S. C. de Alvarenga, ''A survey of intrusion detection in Internet of Things,'' J. Netw. Comput. Appl., vol. 84, pp. 25–37, Apr. 2017.

[10] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, ''Network intrusion detection,'' IEEE Netw., vol. 8, no. 3, pp. 26–41, May 1994.

[11] S. Kishorwagh, V. K. Pachghare, and S. R. Kolhe, ''Survey on intrusion detection system using machine learning techniques,'' Int. J. Control Automat., vol. 78, no. 16, pp. 30–37, Sep. 2013.

[12] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, ''Survey on SDN based network intrusion detection system using machine learning approaches,'' Peer-to-Peer Netw. Appl., vol. 12, no. 2, pp. 493–501, Mar. 2019.

[13] M. Panda, A. Abraham, S. Das, and M. R. Patra, ''Network intrusion detection system: A machine learning approach,'' Intell. Decis. Technol., vol. 5, no. 4, pp. 347–356, 2011.

[14] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, ''A new intrusion detection system based on KNN classification algorithm in wireless sensor network,'' J. Electr. Comput. Eng., vol. 2014, pp. 1–8, Jun. 2014.

[15] S. Garg and S. Batra, ''A novel ensembled technique for anomaly detection,'' Int. J. Commun. Syst., vol. 30, no. 11, p. e3248, Jul. 2017.