International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

Polymorphic Malware Detection by Image Conversion Technique

Arpan Chakraborty, Krishna Kriti, Yateendra, M.S. Bennet Praba

Abstract: ***This model implements ways to detect polymorphic malware. This model uses a dynamic approach to detect the polymorphic malware. The objective is to increase the accuracy and efficiency of the detection as this malware can morph themselves, making it difficult to trace through anti-malware systems. As the tracing is going to be difficult the detection and classification system needs to be flexible that can able to detect the malware in every possible environment. This objective can be achieved by giving the system a superintelligence, this can be done by using the Convolutional Neural Networks (CNNs) in our system. This method records the pattern or the traces made by the polymorphic malware. The pattern is in the form of the image which is formed by converting the binary format of the hash codes. The generated images are then sent to the training module, based on this training module the Convolutional Neural Networks gives the result for any testing data.***

Keyword: Convolutional Neural Network, Image Processing, Metamorphic viruses

I. INTRODUCTION

Malicious files often known as malware are the kind of software that can cause great damage to our data [21]. It was all started back in 1971, the creator Bob Thomas who was working at the BBN Technologies (Bolt Beranek and Newman Inc.) created the first computer virus known as the 'Creeper'[1][2]. The program was not actively a malicious software as it causes no damage to the data, it was just a self-replicating program. To stop this self-replication a new program was created called the 'Reaper'. It was created by Ray Tomlinson [1]. This was the first anti-virus program, the first step taken in the cybersecurity field. Now we have the most extended version of the virus called the malware. Malware is of many types namely trojan horses, ransomware, computer virus, worms, spyware, adware, and scareware [21][20]. As we are leading to the future many developments are happening in these fields. Malware is getting stronger and difficult to trace and at present, there are many methods to rectify them, but as the modifications are going on these methods fails at some points.

Revised Manuscript Received on February 18, 2020.

* Correspondence Author

Arpan Chakraborty, Pursuing B.Tech, Computer Science and Engineering, SRM Institute of Science andTechnology, Chennai, Tamil Nadu, India.

Krishna Kriti, Pursuing B.Tech, Computer Science and Engineering, SRM Institute of Science andTechnology, Chennai, Tamil Nadu, India.

Yateendra, Pursuing B.Tech, Computer Science and Engineering, SRM Institute of Science and Technology, Chennai, Tamil Nadu, India.

M.S. Bennet Praba, Assistant Professor in SRM Institute of Science and Technology in Ramapuram, Chennai, Tamil Nadu, India.

The changing properties of the malware are making it difficult to figure out. These properties are polymorphism and metamorphism. Morph generally means to convert into something. As this malware has morphing abilities, we need a trained system for detection. This section presents the detection of malware files based on the concept of machine learning. To do that we have to recognize the working pattern of malware and then we have to trace it. The idea comes from a mathematical concept known as convolution. In this method we predict the behavior or the modification of the third function based on the behavior of the other two functions. For example: If 'f' and 'g' are two functions then "f*g(t)" is the convolution of the two functions. This is the basic idea used in this section to detect the polymorphic malware.

II. EXISTING DETECTION TECHNIQUES

Many detection techniques are currently being used by the anti-malware systems for tracing and detection purposes which is the first step. These techniques can be mentioned as:

1.) Signature Based Method:

This is the classical form of the detection technique. As the name suggests it uses the signatures of the malware and then it compares it with the scanned files to detect the malware based on its signature [8]. Although it has a very less error rate it has a major disadvantage. The recent malware or new malware does not have any signature, so

this method fails in this case. Considering the case if we did find the signatures of the new malware then for every signature, we have to create a large database for the detection purposes which demands larger storage and its time-consuming. (The flow diagram can be seen as in Figure 2)

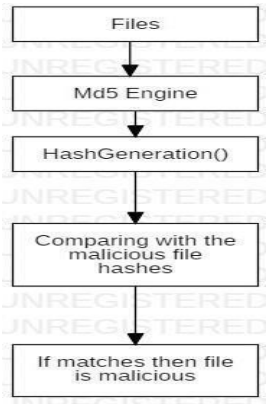

Figure 1

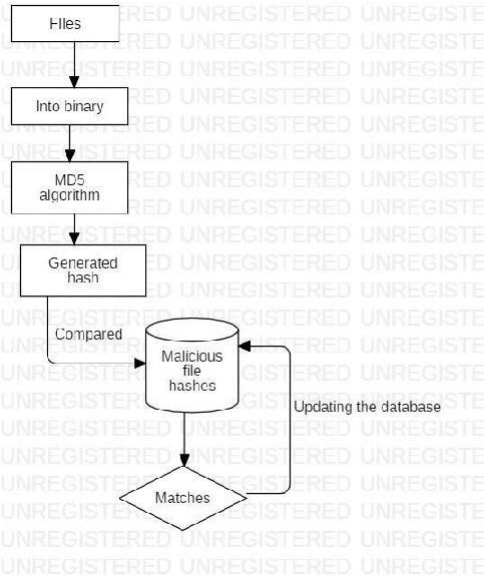

Figure 2

2.) Behavior-Based Method:

This an execution-based detection technique, which detects the malware, based on its execution pattern [10][11]. It's like detecting its suspicious behavior which are the actions performed by the malware that are unauthorized [10]. This approach detects both the static and dynamic behavior of malicious code. Malware carryout malicious behavior

by putting into an effort of system calls. System calls give useful information for finding malicious behavior.

3.) Heuristic Based Method:

Heuristic-based model is used to find the possible types of malware. It's an alternative to the signature-based method [11][12]. This method also uses Artificial intelligence as a way for the detection of malware files. Artificial intelligence gives the ability to self-analyze the code intelligently by performing a deep inspection of the code instruction sequence. Here is the list of the heuristic methods:

a.**) API/System:**

API or application programming interface is used to examine the behavior of a code that sends the request to the operating system [12]. This method analyses the code based on the request made to the operating system. The aim is to analyze what piece of code is sending request to the OS, that idea was proposed by Hofmeyr. The analysis of the code is based on its behavior which is done by using the sequence of the request made by the code to the system. The method which is used for this is known as the Hamming distance. The hamming distance was used for matching the sequences. In the year 2007 Y. Ye[12] proposed the Intelligent malware detection system(IMDS) using the Object-Oriented Association(OOA) mining based classification. Data mining proves a very efficient way of detecting or analyzing the files based on history. Methods like FP (frequent pattern)-Growth algorithms are used for this purpose.

b.) Op-Codes:

These are operational codes which are the subdivision of the machine language that is used for the identification of the operation that is to be executed [11][12]. Generating a graph of operational codes (opcodes) from an executable file, converting this graph to an image and then using the "GIST" method to extract features from each image. In the final step machine learning methods [18] such as Support Vector Machine (SVM), K-Nearest Neighbour (KNN), the ensemble is used for classification.

c.) N-Grams:

N-Grams are the substrings of the larger strings that can of any size. The string can be divided into many parts and every part is equal in length [11], it should not be underestimated as the proper division of the string. E.g. We have a string say APPLE so its N-gram strings of size 3 can be seen as APP, PPL, PLE like this [12]. After the generation of the strings these are converted into their binary form for detection purposes.

[17]The commercial antivirus systems that were used in recent days detect malware when it has already caused some damage to a system or software. The signature-based technique shows a result with an accuracy of 100% in the training data and 98% in 3-fold cross-validation.

d.) Control-flow-graph:

This is a directed graph that is used to analyze the functionality of a program. In the graph, each node tends to the statement of the program and each edge tends to control flow between the statements [11][12]. In the control flow graph searching of the analyzed program for an induced sub-graph which corresponds to the control flow graphs of a malicious program. The resulting detector is built over a strong theoretical framework [19]. To evaluate the proposed detection strategy certain experiments are carried out.

Analyzing Techniques:

1. **Static- analysis:**

Working framework and programming are practically the same as the customary Windows-based framework or UNIX based framework, conventional malware and endeavor can take a shot at these little brilliant sensors and gadgets, the utilization of

which is exponentially expanded. Subsequently, it ends up basic to oversee regularly advancing malware and related security hazards in the time of pervasive sensors and savvy gadgets.

Different methodologies have been proposed for malware locations. Recognition methods proposed before depended on static examination. Static examination looks at the parallel code, breaks down all conceivable execution ways, and distinguishes vindictive code without execution. In any case, dissecting parallel code ends up being troublesome these days. As jumbling procedures become increasingly modern, the static examination can be circumvented by different muddling systems,

for example, polymorphism, encryption, or pressing. Moreover, as static investigation depends on a prebuilt signature database, it can only with significant effort to distinguish new obscure malware until the mark is refreshed. Furthermore, some execution ways can be just investigated after execution. To beat these restrictions of static examination and supplement it, the dynamic investigation has been proposed and is broadly used to accomplish progressively powerful malware discovery. Systems dependent on powerful examination execute malware and follow its practices. Two noteworthy methodologies in unique examination are control stream investigation and API call investigation. The two methodologies recognize malware dependent on examination of closeness between the conduct of the new and the known ones. In any case, malware creators attempt to dodge those strategies through embeddings negligible. Malignant code is an inexorably significant issue that compromises the security of PC frameworks. The conventional line of barrier against malware is made out of malware finders, for example, infection and spyware scanners. Sadly, the two analysts and malware creators have shown that these scanners, which use example coordinating to recognize malware, can be effectively avoided by basic code changes. To address this deficiency, all the more dominant malware locators have been proposed. These instruments depend on semantic marks and utilize static investigation strategies, for example, model checking and hypothesis demonstrating to perform discovery. While it has been demonstrated that these frameworks are exceptionally compelling in distinguishing current malware, it is less clear how fruitful they would be against enemies that consider the novel discovery components. The objective of this paper is to investigate the points of confinement of static examination for the location of malevolent code. To this end, we present a paired jumbling plan that depends on the possibility of murky constants, which are natives that enable us to stack a steady into a register with the end goal that an examination apparatus can't decide its worth. Because of murky constants, we construct confusion changes that dark program control stream, mask access to nearby and worldwide factors, and interfere with the following of qualities held in processor registers. Utilizing our proposed jumbling approach, we had the option to demonstrate that best in class semantics-based malware indicators can be sidestepped. Additionally, our dark consistent crude can be applied in a manner with the end goal that is probably difficult to break down for any static code analyzer. This exhibits static examination procedures alone may never again be adequate to recognize malware. Ongoing work has shown that strategies, for example, polymorphism and transformative nature are effective in sidestepping business infection scanners. The explanation is that syntactic marks are unmindful of the semantics of guidelines. To address this issue, a novel class of semantics-mindful malware indicators was proposed. These finders work with conceptual models, or layouts, that portray the conduct of noxious code. Since the syntactic properties of code are (to a great extent) overlooked, these methods are (for the most part) flexible against the avoidance endeavors talked about above. The reason of semantics-mindful malware indicators is that semantic properties are progressively hard to transform in a robotized style than syntactic properties.

2. Dynamic analysis:

This examination manages dynamic malware investigation, which underlines: how the malware will act after execution, what changes to the working framework, vault also, arrange correspondence occur. Dynamic examination opens up the entryways for programmed age of irregularity and dynamic marks dependent on the new malware's conduct. The examination incorporates a plan of honeypot to catch new malware and a total unique examination lab setting. We propose a standard examination system by setting up the examination devices, at that point running the pernicious examples in a controlled situation to explore their conduct. To test possible new malware, we present two malware tests and their far-reaching dynamic investigation. Malware falls into various classes dependent on its conduct, purpose and disease vector. There is a likelihood for malware to share various qualities of various kinds. Coming up next are the most widely recognized malware types. Infection is a type of malware that attaches itself to an executable record furthermore, it is propelled when the injured individual executes the contaminated program. It is unequipped for duplicating itself or spreading to different machines without client intercession.

crushes all of code jumbling strategies by running the example and watching its conduct in a controlled situation. Along these lines, we can catch the noxious action continuously, investigate the framework changes and system correspondences. The objective is to comprehend the reason for the malware, what does it need to accomplish, where will it interface with, what sort of information it will download and execute, and what data it will exfiltrate.

III. PROPOSED SYSTEM

The proposed system uses a Convolutional Neural Network to detect polymorphic malware. Convolutional Neural Networks (CNNs) is a deep learning concept to implement the image categorization technique as the available computer programs has many difficulties to identify objects for many reasons, including lighting, viewpoint, deformation, and segmentation [25]. This technique is the same as how the human eye works. To detect an object the human eye first recognizes the structure then it's color, it can be visualized as a matrix of size n*m and our brain interprets on every block of it and then based on many hidden layers the object is recognized by humans [24]. CNN is arranged in 3-D structures with width, height, and depth as characteristics. In the case of images, the height and width are the image width and image width, and the depth is RGB(Red, Green, and blue) channels. How it is used in detecting the malware: - The main point here is converting malware into an image. Yes, it is possible to convert malware into the image as one of the proposed solutions has come from a research study called Malware Images: *Visualization and Automatic Classification* by Lakshmanan Nataraj from the Vision Research Lab, University of California, Santa Barbara.

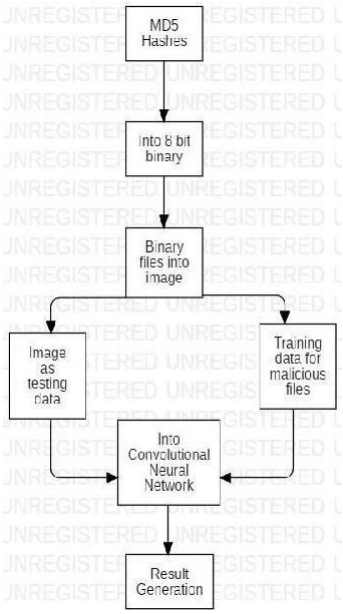

Figure 3

The concept is to see the pattern in which the malware morphs itself. While generating the image files through binary which are converted from the hashes of the malware creates the pattern in the binary form of the image file this pattern can traced by using this convolutional neural network which works like a human eye (Flowchart in figure 3).

This technique can be applied to metamorphic malware which also changes their form after each execution. Metamorphic malware is more advanced than polymorphic malware. Its abilities are far more complex than the polymorphic malware, it can morph itself at every execution. Tracing this kind of can be complicated, thus using the CNN to detect this malware can prove to be useful. To detect

whether the malware is polymorphic or not this project is using a single convolutional neural network that operates on the image file and shows the result. But in order to detect whether the malware is metamorphic or not we have to use two convolutional neural networks, first one will deal with the image file and work on the result and the second one will deal with the binary which is generated from the same image file, the trick is every image in its binary form shows a pattern so taking the property of the metamorphic malware, as it changes with every execution, the pattern in the binary of the image will change accordingly and this is the step where the second neural network comes in work. Now the network will trace the malware by shortlisting the pattern generated by the malware after each execution in its binary form. The implementation of the above statements can be seen as:

1. Extracting the files:

First step is to assemble the files and arrange them to put them into the scanner these can be arranged based on their size and also their extensions the probability of finding malware in more in the extensions like '.dll' and '.rar'. After arranging the files in the folder scanning is done and all the files are converted into their hashes as shown in Figure 4.

2. Converting Binary to image files:

After the hashes are generated these hashes are then converted into its binary format and then this binary format converted into its respective image and grayscale files. Grayscale image files are generated to maintain data integrity means there should be no loss of the data during the testing time (flowchart in figure 3).

3. Training the data:

The images which are generated after the conversion of the binary are now ready for training. The training involves types of images negative and positive. Negatives are the ignored material and positives are accepted materials (as shown in figure 4). Then data is arranged based on the batch size, epoch, and path. Now to prevent the loss some functions should be included like reduced mean and for optimization function Adam Optimizer is used.

4. Testing the data:

The input of the image files is then taken one by one into the CNN. Each image is converted into a pixel matrix and every pixel represents a unique identity of the image. Then from the pixel matrix, a resultant matrix is created which is used for the feature extraction. In the resultant matrix, every value is unique for every image. Using these values with randomization and optimization functions the Convolutional Neural Network generates the result.

Architecture Diagram:

IV. RESULT

As the pattern is recognized by CNN it distinguishes whether it is a polymorphic malware or not. The possibilities of getting the result are staggeringly high. As the current system only distinguishes whether it is polymorphic or not based on the condition given but the conditions can be random as taking the case of malware. So every time a different approach should apply to detect them.

V. CONCLUSION

As the polymorphic malware is detected this opens a broader of detecting many types of morphic malware. The thing is to arrange the convolutional neural network according to the situation as the traces are usually seen in their binary forms when it morphs. The best thing is to trace them at that time.

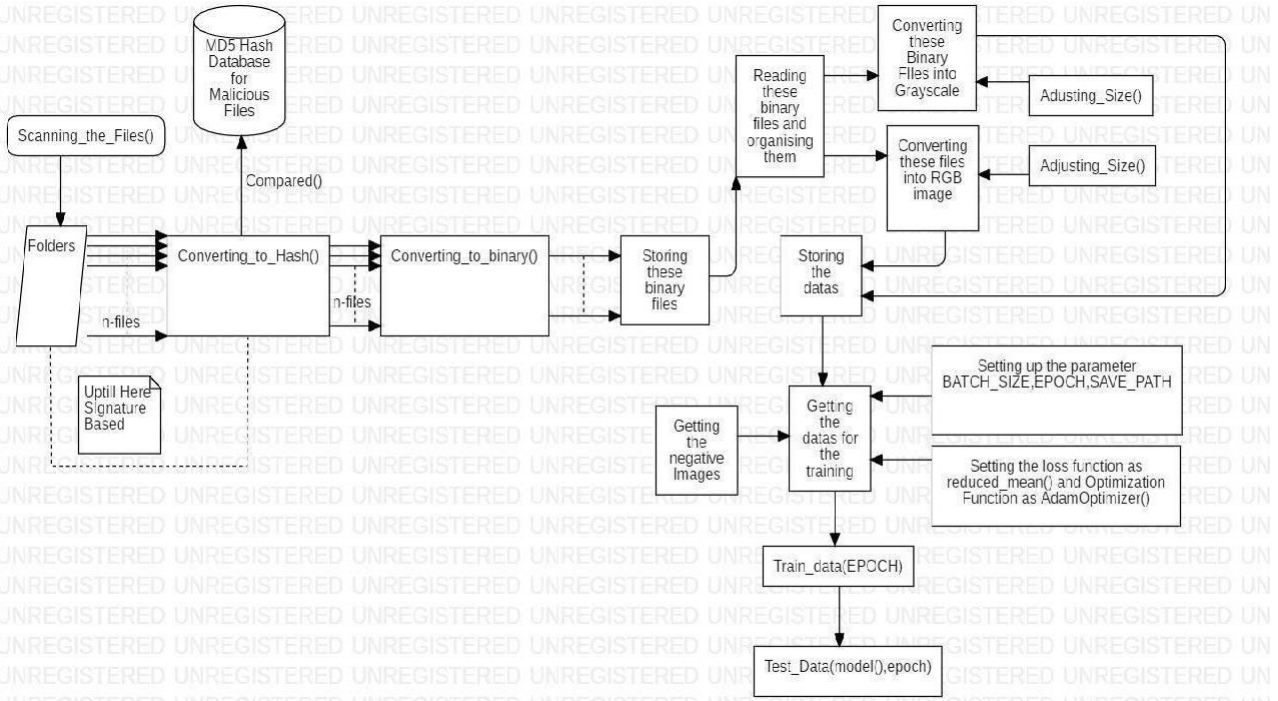

Figure 4

REFERENCES:

1. Wikipedia url:https://en.wikipedia.org/wiki/Computer_vi rus
2. Wikipedia url:https://en.wikipedia.org/wiki/Antivirus_so ftware
3. On the selection of decision trees and random forest (by Simon Bernard, Laurent Heutte and Sebastien Adam)
4. A User-Centric Machine Learning Framework for Cyber Security operation Center (by Charles Feng, Shunning Wu and Ningwei Liu)
5. Malware detection using Machine Learning (by Dragos Gavrilut, Mihai Cimpoesu, D.Anton, and Liviu Ciortuz)
6. Static Malware Detection and Subterfuge: Quantifying the Robustness of Machine Learning and Current Anti-Virus (by William Fleshman, Edward Raff, Richard Zak, Mark McLean, and Charles Nicholas)
7. Malaise-An Effective and Efficient Classification System for Packed and Polymorphic Malware(Silvio Cesare, Yang Xiang, Wanlei Zhou)
8. Analysis of Signature-Based and Behavior-Based Anti-Malware Approaches (by Ashwini Mujumdar, Gayatri Masiwal, Dr. B.B. Meshram) url:http://ijarcet.org/wp-content/uploads/VOLUME-2-ISSUE-6-2037-2039.pdf
9. A state-of-the-art survey of malware detection approaches using data mining techniques (by Alireza Souri & Rahil Hosseini)URL:https://hcisjournal.springerope n.com/articles/10.1186/s13673-018-0125-x
10. https://www.infosecurity-magazine.com/opinions/malware-detection-signatures/
11. Malware Detection and Evasion with Machine Learning Techniques: A Survey (by Jhonattan J., Barriga A., and Sang Gunn Yoo)
12. A Survey on Heuristic Malware Detection Techniques (by Zhara Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, Ali Hamzeh) url:https://www.researchgate.net/publication/ 260729684_A_survey_on_heuristic_malware _detection_techniques
13. Similarity-based Android Malware Detection Using Hamming Distance of Static Binary Features (by Rahim Tahira, Meysam Ghahramani, Reza Javidana, Mohammad Shojafarb, Zahra Pooranianc, Mauro-Conti) url:https://www.researchgate.net/publication/ 335233117_Similaritybased_Android_Malwa re_Detection_Using_Hamming_Distance_of_ Static_Binary_Features
14. A Novel Approach to Detect Malware Based on API Call Sequence Analysis (by Youngjoon K, Eunjin Kim, Huy Kang Kim) url:https://journals.sagepub.com/doi/pdf/10.1 155/2015/659101
15. Limits of Static Analysis for Malware Detection (by Andreas Moser, Christopher Kruegel, and Engin Kirda) url:https://sites.cs.ucsb.edu/~chris/research/do c/acsac07_limits.pdf
16. Analysis and Evaluation of Dynamic Feature-Based Malware Detection Methods (by - Arzu Gorgulu Kakisim, Mert Nar, Necmettin Carkaci, Ibrahim Sogukpinar) url:https://www.researchgate.net/publication/ 330894851_Analysis_and_Evaluation_of_Dy namic_Feature-Based_Malware_Detection_Methods_11th_In ternational_Conference_SecITC_2018_Bucha rest_Romania_November_8-9_2018_Revised_Selected_Papers
17. N-gram-based detection of new malicious code (by T. Abou-Assaleh, N. Cercone, V. Keselj, R. Sweidan) url:https://ieeexplore.ieee.org/document/1342 667 URL:
18. A new method for malware detection using opcode visualization (by Farnoush Manavi; Ali Hamzeh) url:https://ieeexplore.ieee.org/document/8324117
19. Control Flow to Detect Malware(by Guillaume Bonfante, Matthieu Kaczmarek and Jean-Yves Marion) url:https://pdfs.semanticscholar.org/b5fd/917f d9b76e386cf7d12b53d3ee871644f4f4.pdf
20. Review of Viruses and Antivirus Patterns(by Michelle Yusuf Wanjala & Neyole Misiko Jacob) url:https://computerresearch.org/index.php/co mputer/article/view/1608/1592
21. THE NEW AGE OF COMPUTER VIRUS AND THEIR DETECTION (by Nitesh Kumar Dixit, Lokesh Mishra, Mahendra Singh Charan and Bhabesh Kumar Dey) url:http://airccse.org/journal/nsa/0512nsa05.p df
22. Malware Images: Visualization and Automatic Classification(by L. Nataraj, S. Karthikeyan, G. Jacob, B. S. Manjunath) url:https://vision.ece.ucsb.edu/sites/default/fil es/publications/nataraj_vizsec_2011_paper.pd f
23. Dynamic malware analysis of phishing emails(by Mohammad Abu Qbeitah, Monther Aldwairi) url:https://ieeexplore.ieee.org/document/8355435
24. Object detection through search with a foveated visual system by(Emre Akbas, Miguel P. Eckstein) url:https://journals.plos.org/ploscompbiol/arti cle?id=10.1371/journal.pcbi.1005743

25. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review(by Waseem Rawat and Zenghui Wang)url:https://www.mitpressjournals.org/doi/10.1 162/neco_a_00990

AUTHORS PROFILE

Arpan Chakraborty, is currently pursuing B.Tech degree in Computer Science and Engineering from SRM Institute of Science and Technology, Chennai and will be graduated in 2021. He completed his 12th from M.G.M Higher Sec. School, Bokaro, Jharkhand in 2017. Fields of interest include Machine Learning, Graphic Designing and Software Development.

Krishna Kriti, is currently pursuing B. Tech Computer Science Engineering student at SRM institute of science and technology, Chennai and will be graduated in 2021. He completed his 12th from D.A.V Sector-4, Bokaro, Jharkhand in 2017. Fields of interest include Machine Learning, Artificial Intelligence, Computer Security.

Yateendra, is currently pursuing B. Tech in the field of Computer Science and engineering in SRM Institute Of Science and Technology, Chennai and will be graduated in 2021. He completed his class 12th from MDS Sr. Sec. School, Udaipur, Rajasthan in 2017. Fields of interest include Computer Security, Machine Learning, Quantum Computing and Graphic Designing.

M.S. Bennet Praba, is currently working as Assistant Professor in SRM Institute of Science and Technology in Ramapuram, Chennai, Tamil Nadu, India. Fields of interest are Networking and Database Management Systems.