RAFT: A Real-Time Framework for Root Cause Analysis in 5G and Beyond Vulnerability Detection

Yifeng Peng*, *Student Member, IEEE*, Xinyi Li¶, Jingda Yang‡, *Student Member, IEEE*,
Sudhanshu Arya†, *Member, IEEE*, and Ying Wang§, *Member, IEEE*
*†‡§School of Systems and Enterprises, Stevens Institute of Technology, Hoboken, USA
¶Pratt School of Engineering, Duke University, Durham, USA
*ypeng21@stevens.edu, ¶xinyi.li@duke.edu, ‡jyang76@stevens.edu, †sarya@stevens.edu, and §ywang6@stevens.edu

2024 IEEE 21st Consumer Communications & Networking Conference (CCNC) | 979-8-3503-0457-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/CCNC51664.2024.10454824

***Abstract*—The reliability of 5G systems and their applications in a complex, dynamic, and heterogeneous environment requires rigorous testing and real-time detection for system vulnerabilities and unintended emergent behaviors. In this paper, fuzz testing is performed on 5G systems by randomly injecting and permuting control commands into the system under test (SUT) of the 5G radio resource control (RRC) authentication and authorization process, emulating Man-In-The-Middle (MITM) attacks to trigger potential vulnerabilities and unintended behaviors. The fuzzed system behaviors contain information that could indicate the system's health status, and potential vulnerabilities, and, more importantly, it enables the causation analysis in the SUT to detect the location and type of attacks or abnormal inputs from the profiling of the impacted behaviors. We then propose a Real-time Framework for Root Cause Analyses (RAFT) in NextG Vulnerability Detection based on analyzing the random fragments of the log file generated during the communication process. By processing the random fragments of the logging profiles captured during fuzz testing with the continuous bag-of-words (CBOW) Model, we extract the information of states and states transitions and perform causal analysis to identify the root cause for vulnerability detection in the 5G system. The novelty of our framework lies in the creation and analysis of the information extraction that does not require capturing the entire log file instead only the log file fragments to achieve high accuracy. This approach enables real-time detection and deployment to real-life scenarios where access to the entire logging profile is difficult to obtain or unavailable. The presented framework RAFT directly adapts to various machine learning (ML) models, which allow the adaptation to hardware with various computation complexity from internet-of-things (IoT) to Radio Access Network (RAN) servers. The experimental results show a significant performance gain and are thoroughly evaluated by the accuracy and area under the curve (AUC) results. In particular, we show that the proposed framework can attain a high AUC value $(0.92 \leq \text{AUC} < 0.96)$ by accessing only a 70% fragment of the original log file while maintaining a higher accuracy. In addition, we find that RAFT reduces the time complexity by more than 5% as the fragment size reduces to 70% of the original log file. The causation analysis nature of RAFT summarizes vulnerability information into essential root causes that can be easily transmitted within the network in real-time and turned into guidance for back-end engineers. The unique advantages of RAFT, including accurate causation with information fragments, reliable performance without large training datasets and less computation complexity guarantee a wide range of use cases and deployment environment of RAFT.**

***Index Terms*—5G, CBOW, fuzz testing, root cause analysis, machine learning, vulnerability detection.**

This effort was sponsored by the Defense Advanced Research Project Agency (DARPA) under grant no. D22AP00144. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

I. Introduction

The fifth-generation mobile communication technology (5G) introduces numerous cutting-edge possibilities in various verticals providing more mobile bandwidth, massive machine-type communications, and ultra-reliable and low-latency communications. With recent advances in wireless technologies, there has been a focus on developing next-generation ultra-reliable and low-latency communications (URLLC) to facilitate high-speed data transmission with minimal delay. These advancements aim to meet the requirements of various use cases, such as device-to-device (D2D) and IoT applications [1]–[3]. In order to ensure the security of the data transmission between user equipment (UE) and gNodeB (gNB) in 5G communication systems, the detection of vulnerabilities has become paramount. Therefore, it is inevitable to develop a robust framework for vulnerability detection to enable secure data transmission while ensuring confidentiality, integrity, and availability. Recent research has demonstrated a growing interest in the development and analysis of vulnerability detection techniques for 5G communication protocols [4], [5]. However, as the network size expands, the applicability of case-by-case vulnerability detection becomes increasingly limited. Despite the existence of numerous proposed test cases, both functional and non-functional, as well as extensive research on 5G network threats, vulnerabilities, and intrusion detection approaches, the testing of 5G network components remains a challenging task [6]. One of the reasons behind this challenge is the scarcity of available labeled datasets that incorporate realistic user behavior and up-to-date attack scenarios [7].

With the advent of 5G virtualization, hypervisors and containers are becoming more prevalent. While these technologies allow multiple tenants and virtual network functions to reside on the same physical hardware, they also increase the systems' attack surface to threats such as data exfiltration, resource starvation, and side channel attacks [8]. Fuzz testing detects vulnerabilities in 5G systems by injecting and replacing random test data to trigger the System Under Test (SUT) into various states to generate a large amount of sample

data to characterize system behavior. As the size of the 5G network architecture increases, the system complexity metric increases (defined as the sum of the number of nodes in the architecture and the number of one-way interactions). There has been research in fuzz testing in addressing system complexity and discovering vulnerabilities that may arise in complex environments [9]–[11].

However, fuzz testing may not directly reflect the vulnerability, and post-processing may be needed in mining the profiling information generated from fuzz testing to determine the existence of the true vulnerability, the impact of the potential vulnerability, and the root cause of the vulnerability, which may be computation-intensive to process. Existing fuzz testing frameworks focus on identifying a system that crashes, generates memory leak, or shows anomalous behavior, however, more subtle vulnerabilities or vulnerabilities with chain effects are challenging to detect, especially the vulnerabilities that are more related to domain knowledge [12]. Moreover, existing solutions rarely investigate the rooting causing and causation analysis of the system behavior that is exposed via the test cases generated from fuzz testing [13]. The conducted experimental fuzz test cases and the ground truth information they provided can be considered as the crucial step in the vulnerability detection and root cause analysis process. Once the fuzz testing has identified a possible vulnerability, further debugging, and code review could be further required to determine the specific cause of the vulnerability. The causation analysis nature of the presented system RAFT summarizes vulnerability information into the essential root causes that can be easily transmitted within the network in real-time and turned into guidance for back-end engineers.

There has been extensive research in causal analysis for vulnerability detection in wireless communication systems, including Bayesian network model-based causal inference and formal method-based root causes. In [14], [15], the Bayesian network model was constructed to leverage the domain and physical relationship among performance metrics in the 5G network for real-time root cause analysis. A vulnerability detection approach based on the formal method with root cause analysis was documented to identify the vulnerabilities and anomaly behaviors over the 5G Non-Access Stratum (NAS) layer protocol [16]. However, most of the existing vulnerability detection methods rely mostly on the researcher's domain experience and are highly complex. The Bayesian network model's effectiveness relies on domain knowledge to construct an efficient model topology and large data samples to autoform the topology, thereby limiting its applications.

With the advancement of machine learning algorithms, many researchers have utilized machine learning for vulnerability detection in 5G networks, due to their outstanding computational power and ability to extract critical information [17]–[24] and the maturity of the experimental testbed in 5G and nextG [25]–[27]. Furthermore, for deeper analysis of the system-implementation prototyping of the 5G and nextG, a simulation-driven approach has been shifted to an experimental approach [28]. It leverages subsequent open-source softwarization of mobile network functions through various testbeds and significantly accelerates innovation in 5G and nextG vulnerability detection [25], [29].

In this paper, based on the logging profile information generated by fuzzed 5G protocols in the 5G fuzzing platform, RAFT performs causal analysis and the root cause of the fuzzed position in the log files and the layer transitions accurately by RAFT framework and a set of machine learning models. By processing the fuzzed log files, a communication engineer can usually optimize systems, analyze the causes of problems, and track system performance [30]–[32]. Inspired by the recent development of the word embedding models, we utilize it to translate abstract layer-to-layer and state-to-state connectivity relations into vector matrices.

A. Contributions

Our Contributions are summarized as below:

- We proposed a real-time framework RAFT for real-time root cause in 5G and Beyond vulnerability detection. The RAFT framework leverages the fuzz testing generated experimental results and the provided ground truth to examine and analyze the characteristics of the vulnerabilities and unintended emergent behaviors of the 5G authentication and authorization process.
- The novel of the RAFT lies in the creation and analysis of the information extraction that utilizes log files fragments instead of requiring entire log files to be captured in achieving high accuracy. This approach significantly expands the application scenarios of RAFT to scenarios when the entire logging profile is difficult to obtain or not recorded. It enables real-time identification of attack positions and facilitates researchers in conducting an efficacious root cause analysis, thereby streamlining the investigation of complex system behaviors.
- RAFT is a versatile solution that can be integrated with any pre-existing machine learning models. We show that the proposed framework can readily be adapted to most of the ML models, thereby, making it universally compatible with all current advancements in model architecture and feature engineering.
- Our proposed solution RAFT detects the root cause of the SUT behaviors in real-time and provides insight into system behaviors under various MITM attacks. It enables real-time defense for the RRC authentication and authorization of 5G and beyond. The summarised root cause can be easily transmitted in the network and turned into guidance for back-end engineers.

The remainder of this paper is organized as follows. Section II summarises the research background. Section III introduces the methodology of the proposed framework RAFT. Section IV describes the experimental setting for 5G vulnerability detection. Section V depicts the experiment results and analysis. Finally, the conclusions are drawn in Section VI.

II. Background

In this paper, we apply RAFT to log files generated on srsRAN, an open-source project focusing on software-defined radio access networks. Utilizing srsRAN, we perform fuzz testing on 5G RRC and tunneled NAS protocols by message reorder injection and invalid data replacement [33], [34].

A. 5G Protocol Stack

The fifth-generation mobile communication network relies on the 5G protocol stack, which consists of three core layers: the physical layer, the data link layer, and the network protocol layer.

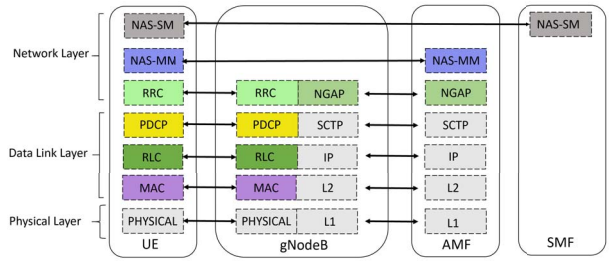


Fig. 1. 5G Protocol Stack.

As depicted in Fig. 1, these layers facilitate communication interactions among several important entities, including UE, gNB, access and mobility management function (AMF), and session management function (SMF). The layer transition and sequence contain critical information about the system behaviors. Based on the layer transition and sequence, in this paper, we further extract and convert it to an information chain that could be used to detect and root cause of the undesired system behaviors.

B. MITM-based Fuzzing Model

As an attack model, MITM achieves the effect of an attack by inserting itself into the communication process and by intercepting the messages sent by both parties. In Fig. 2, MITM Fuzzer performs fuzz on different communication layers in lieu of the normal communication process.

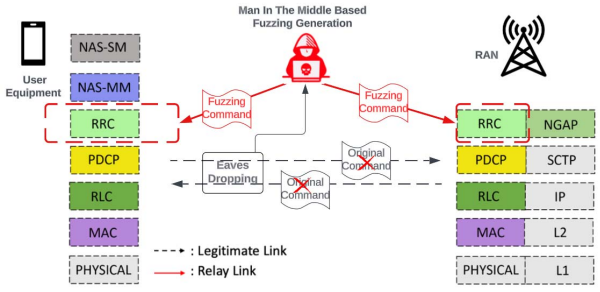


Fig. 2. MITM-based Fuzzing Model.

In this paper, RAFT predicts the information and state of the layer subject to fuzz by processing the communication log files generated during the fuzz testing process. RAFT can effectively help researchers avoid MITM attacks and improve the security of network communications.

III. Methodology

A. CBOW Model

CBOW is known as an algorithm in Word2Vec that generates word vectors, a method for mapping words into vectors that captures the semantic and syntactic relationships between words [35]. The CBOW model is constituted of three main components: the input layer, the hidden layer, and the output layer. Given a target word, neighboring/surrounding words are selected as input. These inputs, derived from one-hot encoded vectors, undergo a transformation via a specific weight matrix, mapping them onto the hidden layer. Following this, the hidden layer's output is formulated as the mean of the word vectors corresponding to the input words. This resultant output undergoes a subsequent transformation through a separate weight matrix, projecting it onto the output layer. Subsequently, the softmax function is employed to transmute each output layer value into a probabilistic value, signifying the likelihood of each word from the vocabulary being the target word.

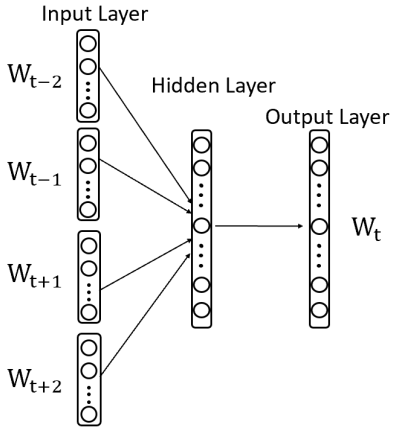


Fig. 3. CBOW Model [35].

The CBOW model [35] is utilized to extract layer connection information from individual log files. The extracted information pertaining to layer-to-layer connections is subsequently transformed into a vector representation.

B. RAFT

In this paper, we propose a real-time framework for root cause analyses, as shown in Fig. 4, where the raft represents our framework, with lotus leaves and rocks floating nearby as possible data fragments for future implementation.

Lotus leaves generally stand for log file fragments, whereas incomplete leaves mean an incomplete log file. These incomplete log files signify the nonnecessity of a whole or complete log file to generate root cause analysis in practical industrial applications, as only the random log file fragments are sufficient to perform the analysis. The rocks with different

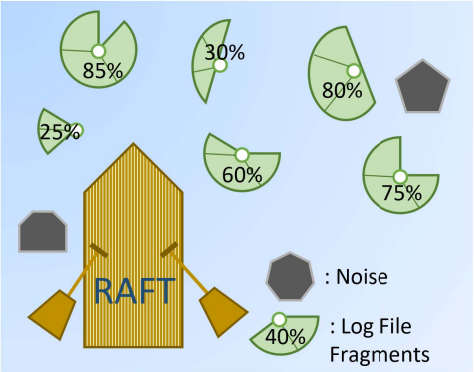


Fig. 4. RAFT Model.

shapes represent noises of distinct characteristics. One possible kind of noise would be some log files without fuzzing. In real industrial scenarios, RAFT encounters lotus leaves and rocks while traveling in the river, and then handles the lotus leaves in real time.

By scrutinizing an arbitrary log file, the number of occurrences and distribution pattern for each layer are randomly generated, illustrated in Fig. 5. The root cause analysis of vulnerability is hardly determined by the researchers directly from the occurrence frequency and pattern. Under this circumstance, ML models are introduced to analyze the root cause of vulnerability by implementing supervised learning, with only the need for log file fragments rather than complete whole log files, which greatly improves the efficiency of detection and is more in line with industrial production requirements.

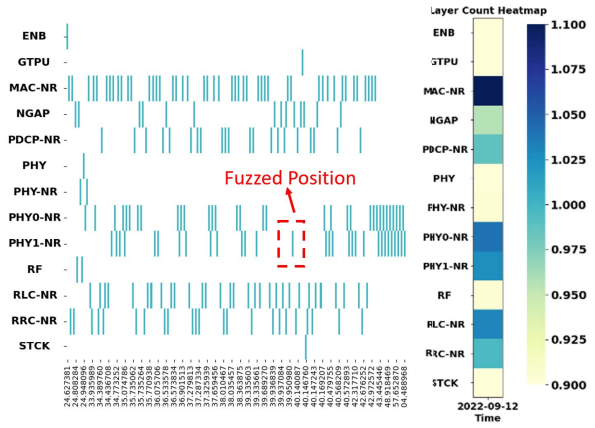
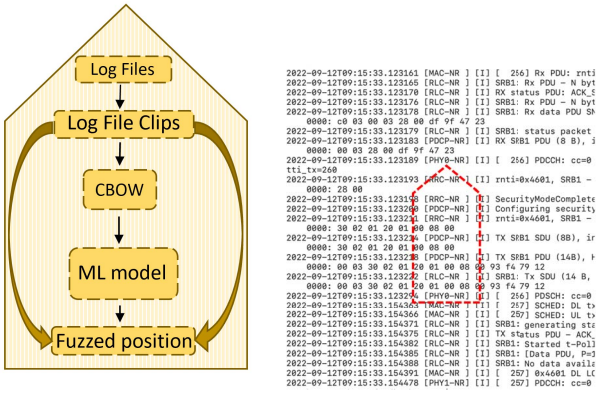


Fig. 5. Heatmap for one arbitrary log file.

In the original log file, as shown in Fig. 6(b), our framework glides through it in real-time and is able to select different random fragments for detection, illustrated as the red dashed framework. In Fig. 6(a), we randomly select fragments of different portions (lengths) through log files and then convert the fragments into corresponding embedding vectors using the CBOW model after simultaneously training the model. These vectors cover different layer information and connection relationships inside the fragments. At last, the ML models are utilized for performing the fuzzed position prediction to help the root cause analysis. More details for RAFT are shown in Algorithm 1.



(a) RAFT Structure (b) Part of one arbitrary log file

Fig. 6. RAFT in real-time log files detection.

IV. Experimental Settings

In Fig. 7, the experiment begins with the fuzz settings, where we define the fuzzing settings as message replacement and injecting large amounts of random, invalid, or abnormal data in the 5G communication system. Then we apply the fuzz testing to RRC and NAS protocols on the srsRAN network so that we can get the log files that record the communication process and statutes. Then, we conducted different experiments on the same log files data set. Each set of log files is distinguished by different colors, with $L_i, i \in \Xi = \{PHY, PDCP{-}NR, \cdots, NGAP\}$, represents the ith layer in the cardinality Ξ in any log file. To generate different lengths of log file fragments, the continuous segments are randomly extracted from the original log files as the fragments shown in the red box of Fig. 6(b). Random percentage in the second step means how much data is taken from each log file, which in other words represents the portion of segment information out of the whole log file information which is continuous. After generating a small piece of the data segment from each log file, the chosen sequence of layers is further transformed into a well-ordered sentence considering layer names and their interdependence. These smaller sentences generated from log file fragments are then combined together as a set, each line in the colored box represents a log file. Three colors are utilized in distinguishing log file fragments sets with different lengths, where the length is shown in percentage form compared with the original length of log files.

As all log files fragments are transferred into sentence representations, the word embedding model CBOW is ready to be applied as shown in the third step. After the word embedding model is trained, each sentence is converted into a word vector representation form through the word embedding model. These word embedding vectors are able to record the connection relations and states of different layers, providing a more detailed machine-friendly data set.

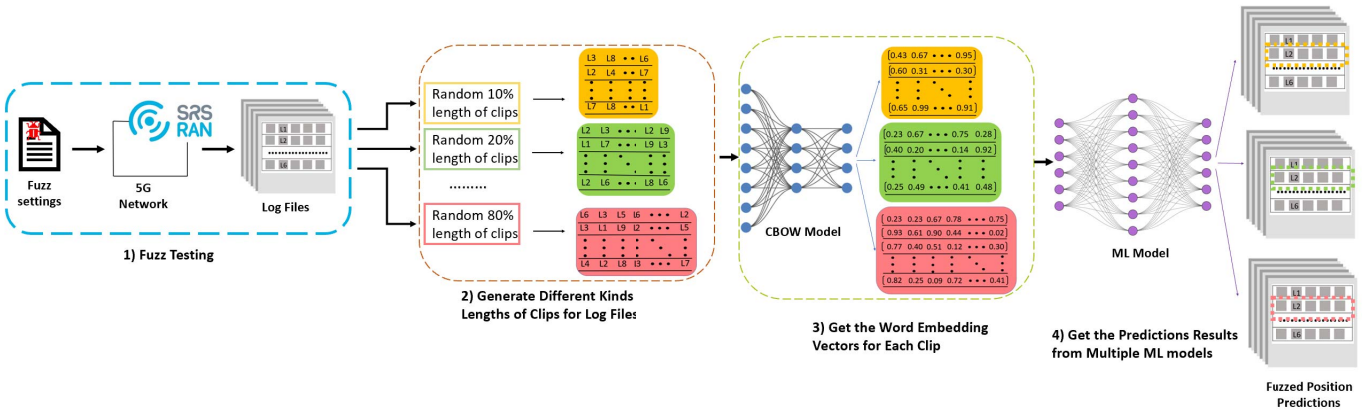


Fig. 7. The flowchart of the experimental process.

Prediction of fuzzed position fell into the last step. Each word embedding vector matrix is predicted for a position utilizing distinct ML models. These fuzzed positions are of great significance for root cause analysis.

In the training of the CBOW model, the dimension of the word embedding vector is set to 128, and the size of the context window to be considered in the training of the model is set to 50, which means that 50 words before and after the central word are considered. The minimum frequency of the words to be considered is specified as 0, resulting from the need to consider all words in sentences. The number of parallel threads for the training of the model is set to 4 and the number of iterations is set to 50.

Six networks are utilized in combination with the word2vec model as the way of training, namely long short-term memory (LSTM), recurrent neural network (RNN), convolutional neural network (CNN), gated recurrent unit (GRU), full connected network (FCN) and multilayer perceptron (MLP).

The LSTM model consists of a two-layer LSTM network followed by a fully connected layer and 500 hidden size (dimension of the hidden state) is contained in the LSTM network. RNN is designed to consist of a two-layer RNN network with 500 hidden sizes and a fully connected layer. CNN contains two one-dimensional convolutional layers. Each convolutional layer is followed by a ReLU activation function and a maximum pooling layer. The number of filters (i.e., the number of output channels) in the convolutional layer is 100. The first convolutional layer has 3 convolutional kernels, and the convolutional kernels move in steps of 1 and padding of 1. The second layer has the same convolutional kernels, steps, and padding as the first layer. GRU consists of a two-layer GRU network with 500 hidden sizes and a fully connected layer. FCN has 100 hidden layers and the optimizer is Adam. MLP has 500 neurons in the hidden layer. The number of training epochs for all the networks is set to 500.

V. Results and Analysis

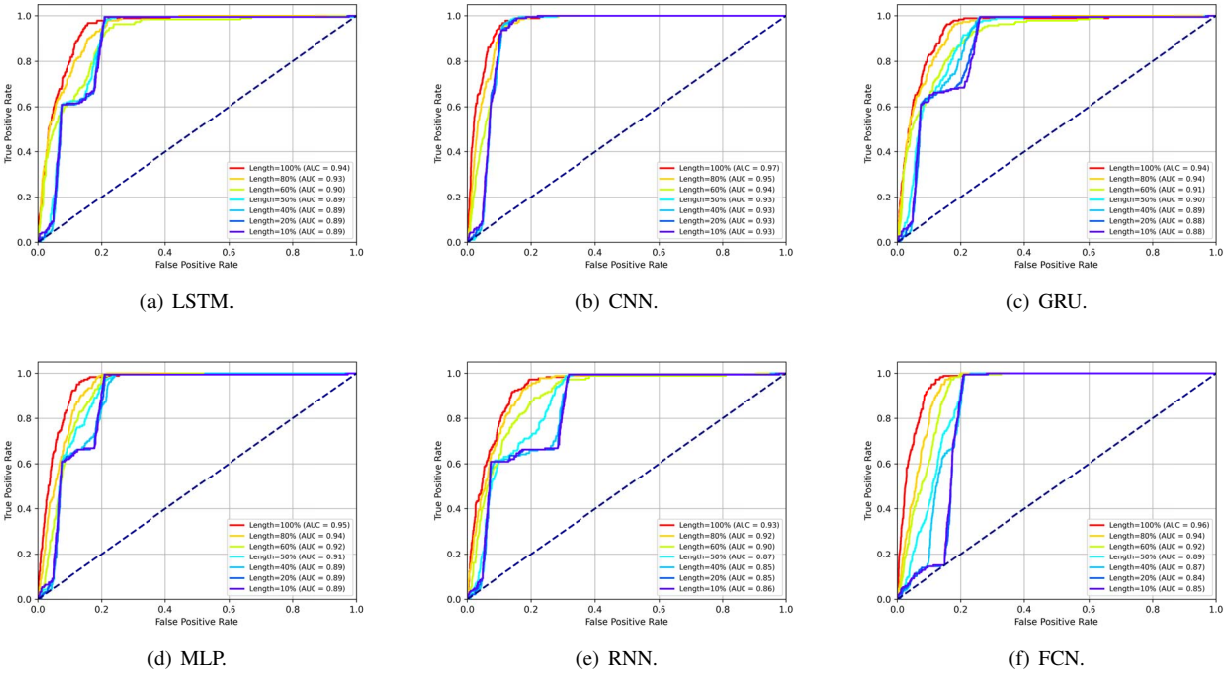
All the input for ML models is the same set of word embedding vectors from the same log file fragments. The word embedding vectors of different portions are fed into different ML models simultaneously. Demonstrated in Fig. 8 are six sets of receiver operating characteristic (ROC) curves, generated from five corresponding ML models. Each set of ROC curves is generated after applying the ML model to eight different lengths of log file fragments after word embedding from length 100% to length 10%. In general, all the ROC curves among all the ML models exhibit similar trends within the different lengths of portions. The segment portion with a larger length normally enjoys a higher AUC value, revealing a better performance. The dashed line illustrates random guessing. A shorter log file length, resulting in shorter sentences for word embedding, has been observed to lead to a notable decline in performance across most of the machine learning models. This phenomenon suggests that the characteristics of the input have a direct impact on the corresponding output.

TABLE I
AUC value over models.

Testing Portions	Testing Length	AUC Over Models					
		LSTM	MLP	RNN	CNN	GRU	FCN
100%	210	0.943	0.954	0.933	0.966	0.940	0.961
90%	189	0.924	0.940	0.930	0.956	0.926	0.949
80%	168	0.934	0.936	0.924	0.954	0.937	0.937
70%	147	0.928	0.934	0.919	0.955	0.930	0.936
60%	126	0.905	0.918	0.899	0.941	0.905	0.919
50%	105	0.894	0.907	0.874	0.927	0.897	0.889
40%	84	0.889	0.892	0.855	0.926	0.888	0.867
30%	63	0.890	0.890	0.853	0.926	0.884	0.849
20%	42	0.889	0.891	0.854	0.927	0.880	0.843
10%	21	0.890	0.893	0.855	0.928	0.879	0.845

Testing Length: Number of layer states in one log file

As depicted in Fig. 9, we first choose various portions of the log file contents. These selected sentences, which are made up of various layers (states), are then fed into the CBOW model to generate their corresponding embedding vectors. Following this, these vectors are input into the t-SNE model. This process allows us to visualize the spatial similarities between different segments using t-SNE. Interestingly, the embedding vectors corresponding to different proportions exhibit distinct spatial characteristics. As we observe in the figure, as the proportion decreases sequentially from 90%, 80%, 70%, to 60% and



(a) LSTM. (b) CNN. (c) GRU.

(d) MLP. (e) RNN. (f) FCN.

Fig. 8. ROC and AUC curves for different models.

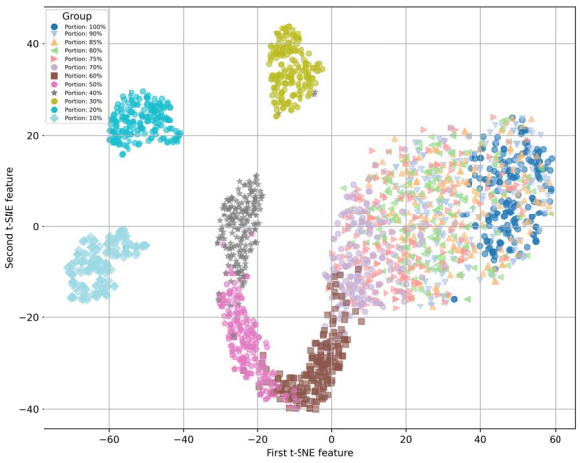


Fig. 9. t-sne for testing data.

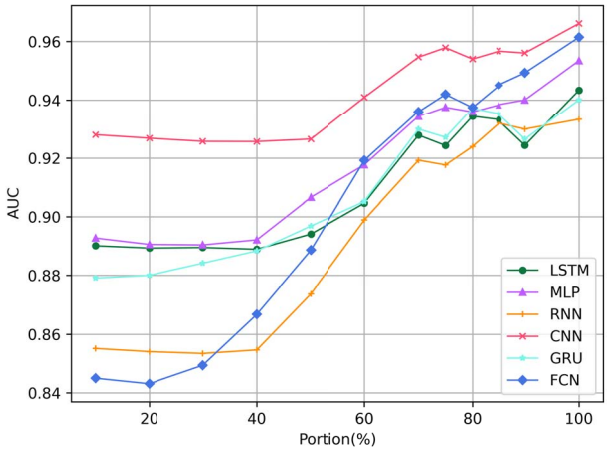


Fig. 10. AUC lines for all models.

finally 40%, there is a remarkable continuity between these vectors. They appear to be linked together, demonstrating a connected transition. However, post 40%, there's a sharp separation, akin to a cliff edge. At 30%, the vectors start to stand apart in a distinct zone, showing no connection with the others. Based on the observations depicted in Fig. 9, it becomes evident that a certain level of interconnection exists between the fragments spanning from 100% to 70%. This interconnection signifies that these fragment lengths convey comparable information, as the model's predictive capabilities are adequately supported by the available length. However, less than the 70% mark, a notable increase in separation between the fragments becomes apparent, albeit with a head-to-tail arrangement. This phenomenon indicates that as the segment length decreases, the contained information diminishes, and the embedding vector matrix captures distinct information patterns.

Further analysis of the figure reveals that starting from 20%, the fragments exhibit a complete disconnection and a discernible trend of separation. This implies that when the segment length is equal to or less than 20%, the state connection information represented by the embedding matrix deviates from the standard information pattern. Consequently,

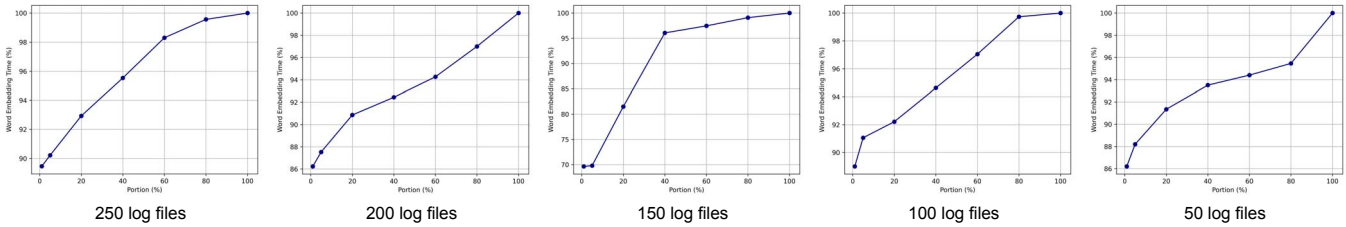


Fig. 11. Time complexity comparison.

Algorithm 1 RAFT

Input: Cardinality $\Xi = \{PHY, PDCP-NR, \cdots, NGAP\}$
Input: N_{th} log file $\chi \in \Xi$
Train CBOW Model:

Randomly generate different portion fragments for the N_{th} log file $\chi' \in \Xi$, vocabulary V for χ'
Vocabulary size $|V|$, the number of training rounds $epochs$, the context window size m, learning rate η, $y(w)$ denotes the one-hot vector representation of the target word w, the weight matrix $u(w)$ and $v(w)$ for word w
for $e = 1$ to epochs **do**
  for $i = 1$ to $|V|$ **do**
    for each word w in χ' **do**
      Calculate the one-hot code vector for each word in $|V|$, h
      Calculate the average of the contextual word vectors for word w
      $h(w) = \frac{1}{2m} \times [h_{-m} + ... + h_{-1} + h_1 + ... + h_m]$
      Calculate the output after input layer
      $s(w) = h(w) \cdot u(w)$
      Calculate the output after hidden layer
      $k(w) = s(w) \cdot v(w)$
      Activated by Softmax to get the Probability
      $\mathbb{P}(w) = \sigma(k(w))$
      Calculate the Loss Function
      $\mathcal{L} = -\sum_{w \in V} y(w) \log \mathbb{P}(w)$
      Gradient descent, back propagation
      $u(w) \leftarrow u(w) - \eta \frac{\partial \mathcal{L}}{\partial u(w)}$
      $v(w) \leftarrow v(w) - \eta \frac{\partial \mathcal{L}}{\partial v(w)}$
    end for
  end for
end for
Get the word vector matrix for fragments of N_{th} log file $\hat{\chi} \in \mathbb{R}^{1 \times 128}$

Train ML Model:

Initialize the parameters for the ML model, the number of training rounds $epochs$.
for $e = 1$ to epochs **do**
  Train ML model LSTM, CNN, GRU, RNN, MLP, FCN
end for
Predict the Fuzzed layer location

Output: Predicted fuzzed layer location for the N_{th} log file

when the length of the segments is excessively short, the word embedding vector matrixes become inadequate in expressing comprehensive information, thereby impeding the model's ability to make accurate predictions due to insufficient information availability.

In summary, Fig. 9 provides insights into the relationship between segment length and the information content represented by the embedding matrix. It demonstrates that longer clip lengths allow for substantial information expression and effective predictions, while shorter lengths limit the amount of information conveyed and hinder the model's predictive capabilities. As the length of the selected portions starts to get shorter, we can clearly see that the points in the graph are also gradually separating, indicating the decay of the information, which is of great significance for visualizing how much information is available.

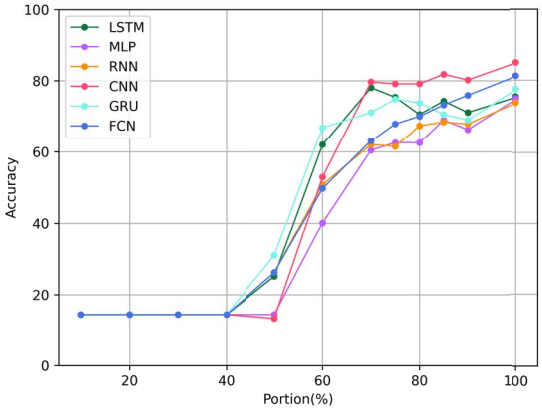


Fig. 12. Accuracy over portions for different models.

From Fig. 10, it's evident that the model's decision-making capacity diminishes significantly as the fragment size reduces, a consequence of decreasing information. This leads to the model making incorrect predictions. There's a significant decrease at around 60%, and the performance deteriorates severely by the time it hits 40%. The x-axis represents the percentage length of the selected fragments of the log files, while the y-axis portrays the AUC values. This figure effectively illustrates the fluctuation of AUC values with respect to

the length of the testing portions. In Table I, similar trends can be summarized.

All of the machine learning models show a similar pattern of increasing performance, with a region spanning from 30% to 60% of the log file length. However, in general, we will find that the three curves of CNN, FCN, and MLP are generally higher than the other three models based on LSTM, GRU, and RNN. This is because the prediction target is not directly related to the order of the input sequence, and the performance of all these networks that can handle time series is therefore somewhat misleading. In the higher percentage regions, which corresponds to more comprehensive information extracted from the log files, the AUC value remains high.

A similar trend is illustrated in Fig. 12 for the accuracy of different models. In obtaining these results, the tolerance is set to 1, which implies that if the fuzzed position is f_p, then $f_p \pm 1$ will be considered to be correct.

Next, we show an important performance metric, that is, the computation time for CBOW. The corresponding time measurements, in seconds, across all file numbers and various log file lengths, are provided in Table II.

TABLE II
TIME COMPLEXITY OF CBOW.

	Tesing portion (%)						
File Number	**100**	**80**	**60**	**40**	**20**	**5**	**1**
250	4.72s	4.70s	4.64s	4.51s	4.39s	4.26s	4.22s
200	3.89s	3.78s	3.67s	3.60s	3.54s	3.41s	3.36s
150	2.84s	2.82s	2.77s	2.73s	2.32s	1.98s	1.98s
100	1.94s	1.94s	1.88s	1.84s	1.79s	1.77s	1.73s
50	0.96s	0.92s	0.91s	0.90s	0.88s	0.85s	0.83s

s: seconds
File Number: Number of Testing Log Files

Based on the time complexity analysis presented in Table II, Fig. 11 presents a percentage comparison of the time complexity for various test portions. A 100% signifies the time required to process an entire log file given the specific number of files. A consistent reduction in time complexity is observed as the portion of the log file decreases. On average, we notice a 20% decrease in the total time consumed when shifting from processing entire log files to merely 1% of log file length. Moreover, we would like to point out that, since the testing log files varies in length significantly due to the random nature of the communication process, therefore, as illustrated in Fig. 11, there is no direct relation between the computational time and the number of log files.

VI. CONCLUSION

In this paper, we have proposed a real-time framework for root cause analyses in 5G and beyond vulnerability detection based on analyzing the random fragments of the log file generated during the communication process. The conducted experiments successfully showed the effectiveness of the proposed framework. RAFT is a versatile solution that can be integrated with any pre-existing machine learning models and can achieve real-time detection with high AUC value($0.92 \leq \text{AUC} < 0.96$). By the innovative approach of the creation and analysis of the information extraction, RAFT allows utilizing log files fragments instead of requiring entire log files to be captured in achieving high accuracy. Without scarifying accuracy, the fragment-based real-time detection in RAFT also reduces computation complexity with 5%. In the future, we aim to further improve our framework. Specifically, we plan to direct the search process of our fuzz testing as an optimization problem, that is, maximization of 5G protocol state transitions to significantly improve the coverage of the 5G protocol state.

REFERENCES

[1] S. Arya, J. Yang, and Y. Wang, "Towards the designing of low-latency sagin: Ground-to-uav communications over interference channel," *Drones*, vol. 7, no. 7, 2023. [Online]. Available: https://www.mdpi.com/2504-446X/7/7/479

[2] S. Arya and Y. H. Chung, "Fault-tolerant cooperative signal detection for petahertz short-range communication with continuous waveform wideband detectors," *IEEE Transactions on Wireless Communications*, vol. 22, no. 1, pp. 88–106, 2023.

[3] B. Mak, S. Arya, Y. Wang, and J. Ashdown, "Characterization of low-latency next-generation evtol communications: From channel modeling to performance evaluation," *Electronics*, vol. 12, no. 13, p. 2838, 2023.

[4] J. Xiong, E. C.-H. Ngai, Y. Zhou, and M. R. Lyu, "Realproct: Reliable protocol conformance testing with real nodes for wireless sensor networks," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, 2011, pp. 572–581.

[5] Y. Qi, Q. Pei, Y. Zeng, C. Yang, and S.-p. Bu, "A security testing approach for wsn protocols based on object-oriented attack model," in *2011 Seventh International Conference on Computational Intelligence and Security*, 2011, pp. 517–520.

[6] Z. Salazar, H. N. Nguyen, W. Mallouli, A. R. Cavalli, and E. M. Montes De Oca, "5Greplay: A 5G Network Traffic Fuzzer - Application to Attack Injection," in *ACM International Conference Proceeding Series*. Association for Computing Machinery, 8 2021.

[7] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Computers & Security*, vol. 82, pp. 156–172, 2019.

[8] A. Dutta and E. Hammad, "5g security challenges and opportunities: A system approach," in *2020 IEEE 3rd 5G world forum (5GWF)*. IEEE, 2020, pp. 109–114.

[9] Y. Cao, Y. Chen, and W. Zhou, "Detection of pfcp protocol based on fuzz method," in *2022 2nd International Conference on Computer, Control and Robotics (ICCCR)*, 2022, pp. 207–211.

[10] S. K. Cha, M. Woo, and D. Brumley, "Program-adaptive mutational fuzzing," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 725–741.

[11] H. Yoo and T. Shon, "Grammar-based adaptive fuzzing: Evaluation on scada modbus protocol," in *2016 IEEE International conference on smart grid communications (SmartGridComm)*. IEEE, 2016, pp. 557–563.

[12] M. E. Garbelini, Z. Shang, S. Chattopadhyay, S. Sun, and E. Kurniawan, "Towards automated fuzzing of 4g/5g protocol implementations over the air," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 86–92.

[13] H. Chen, B. Yuan, D. Zou, and H. Jin, "A fuzzing-based method for testing rules in intrusion detection systems in 6g networks," *IEEE Network*, vol. 36, no. 4, pp. 150–158, 2022.

[14] Y. Wang, S. Jere, S. Banerjee, L. Liu, V. Modeling, and S. Dayekh, "Anonymous Jamming Detection in 5G with Bayesian Network Model Based Inference Analysis Sachin Shetty," in *IEEE International Conference on High Performance Switching and Routing 6–8 June 2022 // Virtual Conference*, 2022.

[15] S. Jere, Y. Wang, I. Aryendu, S. Dayekh, and L. Liu, "Machine learning-assisted bayesian inference for jamming detection in 5g nr," *arXiv preprint arXiv:2304.13660*, 2023.

[16] X. Hu, C. Liu, S. Liu, W. You, Y. Li, and Y. Zhao, "A systematic analysis method for 5g non-access stratum signalling security," *IEEE Access*, vol. 7, pp. 125 424–125 441, 2019.

[17] H. Fang, X. Wang, and S. Tomasin, "Machine learning for intelligent authentication in 5g and beyond wireless networks," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 55–61, 2019.

[18] M. E. Morocho-Cayamcela, H. Lee, and W. Lim, "Machine learning for 5g/b5g mobile and wireless communications: Potential, limitations, and future directions," *IEEE Access*, vol. 7, pp. 137 184–137 206, 2019.

[19] Y. Wang, A. Gorski, and L. A. DaSilva, "Ai-powered real-time channel awareness and 5g nr radio access network scheduling optimization," in *2021 17th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2021, pp. 1–7.

[20] S. Arya and Y. H. Chung, "Novel iterative machine learning for accurate photon-counting poisson channel estimation," *IEEE Communications Letters*, vol. 26, no. 9, pp. 2081–2085, 2022.

[21] Y. Peng, X. Li, S. Arya, and Y. Wang, "Deft: A novel deep framework for fuzz testing performance evaluation in nextg vulnerability detection," *IEEE Access*, vol. 11, pp. 116 046–116 064, 2023.

[22] Y. Peng, J. Yang, S. Arya, and Y. Wang, "Smile net: A supervised graph embedding-based machine learning approach for nextg vulnerability detection," in *MILCOM IEEE Military Communications Conference, 1-7*, 2023.

[23] S. Arya and Y. H. Chung, "Ann-assisted real-time blind signal detection over time-varying doubly-stochastic poisson channel for indoor optical wireless communications," *IEEE Photonics Journal*, vol. 12, no. 5, pp. 1–18, 2020.

[24] S. Yuan, J. Yang, S. Arya, C. Lipizzi, and Y. Wang, "From ambiguity to explicitness: Nlp-assisted 5g specification abstraction for formal analysis," *arXiv preprint arXiv:2308.03277*, 2023.

[25] Y. Wang, A. Gorski, and A. P. da Silva, "Development of a data-driven mobile 5g testbed: Platform for experimental research," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*. IEEE, 2021, pp. 324–329.

[26] J. Yang and Y. Wang, "A nextg hybrid testing platform for multi-step deep fuzzing and performance assessment from virtualization to over-the-air," in *IEEE Cloudnet*, 2023.

[27] J. Yang, S. Arya, and Y. Wang, "Formal-guided fuzz testing: Targeting security assurance from specification to implementation for 5g and beyond," *arXiv preprint arXiv:2307.11247*, 2023.

[28] X. Foukas, M. K. Marina, F. Sardis, M. A. Lema, F. Foster, and M. Dohler, "Experience building a prototype 5G testbed," in *EM-5G 2018 - Proceedings of the 2018 Workshop on Experimentation and Measurements in 5G, Part of CoNEXT 2018*. Association for Computing Machinery, Inc, 2018, pp. 13–18.

[29] Y. Wang, P. Tran, and J. Wojtusiak, "From wearable device to open-emr: 5g edge centered telemedicine and decision support system." in *HEALTHINF*, 2022, pp. 491–498.

[30] P. Kiran, M. G. Jibukumar, and C. V. Premkumar, "Resource allocation optimization in lte-a/5g networks using big data analytics," in *2016 International Conference on Information Networking (ICOIN)*, 2016, pp. 254–259.

[31] S. Yi, X. Hu, and H. Wu, "An automatic reassembly model and algorithm of log file fragments based on graph theory," in *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2015, pp. 686–689.

[32] Z. Wang and Y. Wang, "Nlp-based cross-layer 5g vulnerabilities detection via fuzzing generated run-time profiling," *arXiv preprint arXiv:2305.08226*, 2023.

[33] J. Yang, Y. Wang, T. X. Tran, and Y. Pan, "5g rrc protocol and stack vulnerabilities detection via listen-and-learn," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, 2023, pp. 236–241.

[34] D. Dauphinais, M. Zylka, H. Spahic, F. k. Shai, J. Yang, I. Cruz, J. Gibson, and Y. Wang, "Automated Vulnerability Testing and Detection Digital Twin Framework for 5G Systems," in *9th IEEE International Conference on Network Softwarization*, Madrid, 6 2023.

[35] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.