Reverse Analysis Method of Unknown Protocol Syntax in Mobile Multimedia Communications

Yichuan Wang, Binbin Bai, Zhigang Liu, Xinhong Hei(✉), and Han Yu

Laboratory for Network Computing and Security Technology,
Xi'an University of Technology, Shannxi, China
heixinhong@xaut.edu.cn

Abstract. With the development of mobile multimedia communication, the frequency of a few protocols and special communication protocols increases rapidly, and the probability of network attack events is frightening. This has caused people financial losses and psychological panic. At present, the commonly used protocol recognition tools use a single and targeted method, which has a lot of limitations. Based on the existing algorithms, this paper presents a new protocol feature extraction method, which greatly improves the efficiency of feature extraction. In this paper, the idea of Apriori algorithm is improved, and the feature string is searched under the idea of composite features of CFI algorithm. Combining the advantages of previous algorithms, a more efficient OFS (Optimal Feature Strings) algorithm is proposed, which can perform better when facing the feature extraction problem of bit stream protocol. Then it compares with the existing algorithms, tests the superiority of the new algorithm in running time, and further illustrates the accuracy and correctness of the algorithm.

Keywords: Protocol recognition · Feature extraction · Composite feature · Communication security

1 Introduction

With the rapid development of China's social economy, China's science and technology level has been improved as a whole. As the product of information society, mobile multimedia communication technology is widely used in all walks of life. By using the advantages and values of mobile multimedia communication technology, it has made great contributions to the rapid development of social information in China [1]. As multimedia communication technology involves many fields, such as service, military, finance and administration, it plays an important role in modern social production activities. Therefore, this paper analyzes the specific application and development trend of multimedia communication technology [2,3], so as to improve the role of multimedia communication technology in promoting the development of modernization and make it more in line with the needs of social development goals.

J. Xiong et al. (Eds.): MobiMedia 2021, LNICST 394, pp. 47–62, 2021.
https://doi.org/10.1007/978-3-030-89814-4_4

As the most widely used modern technology in the field of science and technology, multimedia communication technology occupies an important development position [4]. Due to the rapid development of multimedia communication technology, it directly affects the development and openness of modern society. Therefore, strengthening the application of multimedia communication technology and making clear the development trend of multimedia communication technology are conducive to promoting the rapid development of science and technology society in China. Multimedia communication technology is mainly through the media to process the information and data, and display the data in the form of files, audio or pictures. It is an effective method to realize the rapid dissemination of information and interactive processing [5].

In recent years, a variety of network security incidents have frequently appeared in the public's vision [6,7]. The endless malicious network attacks have brought a lot of economic losses and psychological panic to the people. Network security has a great impact on the people, enterprises and countries. At this time, reasonable and effective network security supervision is of great significance. In order to better regulate network security, it is necessary to identify and analyze the unknown protocols in the network [9].

The three basic elements of the protocol are semantics, syntax and timing [10]. The inference of protocol message format and the determination of its field content belong to the content of protocol syntax analysis. The analysis and extraction of protocol syntax is the basis of protocol analysis and identification. It needs to analyze the control statement of protocol message, and extract the semantics of protocol based on data mining and sequence ratio method. The purpose of protocol syntax rule inference is to build a logical model of protocol syntax, focusing on the inherent logical relationship between protocol messages [11]. How the protocol interacts must follow certain syntax rules.

By analyzing the role of network protocol specification in the field of network supervision, we can obtain the network traffic information in the target network [12]. By classifying the traffic generated by these protocols, the network usage can be identified, the network expansion plan can be formulated, and the bandwidth of specific protocols can be controlled. Protocol analysis can help analyze network vulnerabilities, or provide useful information for firewalls and intrusion detection and defense systems, so as to discover and prevent previously unknown attacks.

With the rapid development of computer network and communication technology, the rise of various network services based on data transmission has enriched people's life. At the same time, the network security issues related to privacy and sensitive data have been paid more and more attention. Usually, security protocols run in a very complex development environment [13]. In order to reveal each vulnerability of communication protocol accurately and effectively through formal analysis and reasoning of security protocol model, it is necessary to cover every detail in modeling [14]. However, this is difficult to achieve, and when the security vulnerability is not considered in the model, it is difficult to improve the protocol.

Bitstream protocol format analyzer works at the bottom of network environment. By analyzing and processing the bitstream protocol data captured in the vehicle, the content of these data can be analyzed in real time, and then the protocol format can be analyzed [15]. Further ensure the safety of vehicles and the privacy of passengers. The current network protocol analysis method is to analyze a large number of protocol frames, and the data frame itself is relatively complex, and the algorithm will run for a long time [16]. How to optimize the algorithm is a research direction that needs to be studied continuously.

The rest of this paper is arranged as follows. The first part introduces the development of unknown protocols in network security [17,18]. The second part introduces the related work we have done in unknown protocol parsing. The third part is about the Basic knowledge reserve. In the fourth part, we propose a new protocol format analysis algorithm. In the fifth part, we analyze the performance of the new algorithm from several aspects and compare it with other algorithms. Finally, let's summarize our work.

2 Related Work

In the aspect of protocol feature extraction, pattern recognition and data mining have some applications. For example, Wang Xufang based on the existing BM algorithm and analyzed the common pattern matching algorithm [19]. Data mining, such as unsupervised, supervised and semi supervised learning methods to study. The algorithm used in this paper is optimized by the main idea of association rule mining algorithm in data mining.

Protocol informatics project published by Beddoe, referred to as PI project [20], is used for protocol reverse analysis, but its idea is also applicable to protocol identification. In bioinformatics, it is necessary to search for specific genes that produce proteins from DNA [21]. Similar to this, many network applications will identify certain protocols through some specific domains. Therefore, PI project introduces sequence alignment algorithm to match these similar fields. The purpose of sequence alignment algorithm is to find a way of permutation, combination or insertion to make the two sequences most similar after splitting and reorganizing the sequence [22]. That is, the two string sequences are arranged up and down, cut off or insert spaces in some positions, and then compare their string matching on each block in turn, so as to find out an arrangement method that makes the two sequences most similar. Similarly, for many protocols, there are some similar signature codes or feature fields [23]. Therefore, many researchers use this feature to match these feature fields in the way similar to pattern matching, so as to judge and identify a certain protocol. Good results have been achieved in some specific protocols [30]. This method belongs to the customized feature extraction method, but there may be some missing reports, because there may be a special case, that is, some packets using the protocol do not have the feature field, and if the protocol has update iteration, it is necessary to re study the protocol extraction features.

In addition to the above methods for extracting special fields, there are also methods for feature extraction using flow statistics. For example, Andrew

Moore [24] extracted 248 features for network flows, including packet level and flow-i Even, TCP protocol acknowledgement number and sequence number and some flag bits, but in order to calculate all these characteristics, a lot of computing resources are needed, and there will be some invalid features, which are not helpful for classification. Therefore, Nguyen [25] screened these features to a certain extent, and selected 20 distinguishing features, which greatly improved the computational efficiency.

For example, Chen Liang [26] and others have proposed a feature extraction method based on Chi square statistics, which optimizes feature extraction by introducing chi square statistics in statistical theory. By analyzing the difference between the target protocol traffic and the total traffic or other traffic, the validity of the feature is defined, and the optimal feature is selected, and then the eDonkey protocol is used to optimize the feature extraction. Experiments are carried out on the protocol, and good results are achieved in the recognition of the protocol, but this is a customized feature extraction method.

Chu Huilin [27] and others used the ReliefF algorithm to filter the irrelevant features in the traffic characteristics to obtain the optimal feature subset, and then combined with genetic algorithm and support vector machine to optimize the model parameters of support vector machine, so as to achieve better classification effect. In addition, Ma Yongli et al. [28] used correlation feature selection and genetic search methods to filter out some useless features from all traffic attribute features, so as to obtain a new feature subset. All features of the subset are relatively representative traffic attribute features, so better classification effect can be obtained. However, these two methods are not universal enough, and each protocol needs to be analyzed separately.

Yang Feihu [29] optimized ReliefF algorithm and combined ReliefF algorithm with mutual information method. Firstly, the ReliefF algorithm is used to remove some classification independent features, and then the mutual information method is used to remove redundant features. According to the classification results, the threshold is adjusted and iterated. Finally, through a number of comparative experiments, it is proved that the optimized algorithm achieves better results in classification accuracy and feature dimension reduction effect. However, because of the idea of wrapper, the time complexity of the algorithm becomes higher.

In 2016, Guan Lei et al. [31] introduced big data analysis technology into the field of network security situation awareness, and proposed a multi-functional security situation assessment platform. The platform provides many functions that need to be used in the situation assessment process, which has a good reference significance for the design and implementation of the subsequent situation assessment platform.

In 2016, Zhang Shuwen [32] and others proposed a hierarchical network security situation awareness model based on information fusion. The model uses D-S evidence theory for data fusion, and evaluates situation parameters through neural network technology. Finally, experiments show that the model can solve the problem of strong subjectivity in parameter setting in the field of situation awareness.

In 2018, zjfan et al. [33] focused on network security situation prediction, and proposed a prediction method based on spatio-temporal analysis. The method predicted the network security situation from two dimensions of time and space. Finally, experiments show that this method can more accurately predict the change trend of future network security compared with other state aware prediction algorithms.

3 Protocol Feature Extraction Algorithm

3.1 Algorithm Flow

Algorithm Related Definitions. In order to better illustrate the algorithm, some concepts are introduced here.

Definition 1 minimum support: A user-defined reasonable threshold used to measure the size of support, which represents the minimum statistical significance of the data, and can be expressed as Min_Sup .

Definition 2 frequent substring: Suppose there are M data frame messages with the length of $L1$ bit sequence. If there is a substring β with length of $L2(L1 \geqslant L2)$, if β appears in K data frames, the probability of β occurrence is $p(K/M)$. If the probability of a string occurrence is greater than or equal to Min_Sup, then it is called frequent substring [3].

$$Seq = \{\beta | P(\beta) \geqslant Min\_Sup\} \tag{1}$$

Define 3 minimum frequent substring length: A user-defined value. The length of a frequent substring must meet the minimum frequent substring length. Otherwise, it will be filtered. The value is expressed as Min_Len.

Definition 4 protocol features: If a certain frequent substring β appears frequently at a certain or multiple positions in the protocol data frame, it is considered that the frequent substring may be the protocol feature of the protocol, expressed as Fi [3].

Algorithm Data Initialization. The algorithm data is initialized in five steps:

(1) Input support threshold Min_Sup, traverse the data set, find out the data with the longest length in the data set, and record the length as Max_len;
(2) Use a one-dimensional vector, $localVector$, and use Max_len initializes it, and all elements in it are initialized to 0 by default;
(3) Go through all the data frames of the data set, and record whether the element of each position of each data is '0'. If it is '0', then add one to the location of the $localVector$. For example, if the data[i] in the i-th position is equal to '0', then the $localVector[i]$ plus one;
(4) Traverse the vector $localVector$ once and calculate the support of each position. If the support of a position $sup \geqslant Min\_Sup$ or $sup \leqslant 1 - Min\_Sup$ (assuming$Min_Sup > 0.5$), it means that the position may exist in a feature string, otherwise, the position cannot exist in a feature string.

After calculating the support of each location, two important definitions need to be introduced:

Define 5 bad characters: If a location is not supported within the range described in step (4), the location is considered a bad character and is represented as Ci.

Define 6 ideal strings: A substring that appears between two adjacent bad characters in the $localVector$ and is represented as Pi.

For Definition 6, it is clear that a data frame collection has only one bad character $C1$, then the substring from 0 to $C1$ of the $localVector$ (containing 0 characters but not $C1$) is considered an ideal string. Similarly, a substring from $C1$ to the end of the $localVector$ (containing the end character but not the $C1$ character) is also an ideal string. You can pass the minimum frequent substring length Min_Len partially filters the ideal string.

(5) At this point, through the location of the $localVector$ and bad characters, it is easy to get all the ideal strings, and record all the ideal strings in a set $prunSet$.

Data Reprocessing. After data preprocessing in the previous step, a $prunSet$ has been obtained that contains all possible locations where the signature string may appear. However, since the data in the $prunSet$ is calculated from the frequency at each location, such data must contain a large amount of redundant data. That is, the calculation of each location can get a range, but the range is too large, which is not friendly for subsequent specific operations to find the feature string. The reason for this is not difficult to find. Obviously, because the frequency statistics for each location ignore the continuous attributes of the string, the result is a wide range. It is not difficult to imagine that reprocessing using the continuous properties of a string is a good way. See the following steps for specific operation.

(1) Traverses through each data Str in the $prunSet$, creating a one-dimensional vector $localVector$ with the length of Str and setting it to 0.
(2) Traverse the dataset $dataSet$ to intercept strings of data of the same length and location as Str. Depending on the minimum frequent substring length Min_Len of Definition 3 make a slice and determine. If they are equal, the $localVector[i]$ plus one for the slicing position i. If not, do not operate.

The updated $prunSet$ can then be obtained by referring to the third, fourth, and fifth steps of the preprocessing operation. At this point, the data processing operation has been completed.

3.2 Algorithm Flow

The algorithm is described in Table 1. The flow chart of the algorithm is shown in Fig. 1.

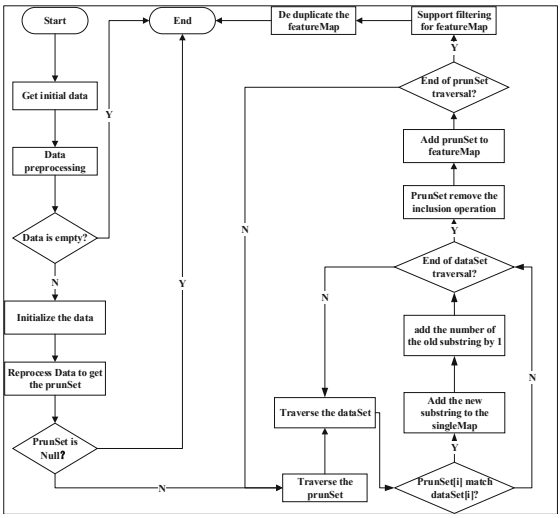

Fig. 1. Flow chart of OFS algorithm.

Table 1. Description of OFS algorithm.

Input	The ideal string set of data preprocessing is $prunSet$; the data frame set is $dataSet$; the minimum support Min_Sup;
Output	Frequent itemset $featureMap$
1	Judge whether $prunSet$ is empty;
2	If it is empty, the algorithm ends; otherwise, the $prunSet$ is reprocessed;
3	Judge whether the $prunSet$ is empty. If it is, the algorithm ends. Otherwise, traverse the $prunSet$
4	Traverse each piece of data in the $prunSet$ in the data frame set $dataSet$;
5	From each data frame in the $dataSet$, the strings with the same position and length as the data are intercepted and compared. If any substring matches successfully, it will be added to $singleMap$;
6	Whether the $dataSet$ traversal is finished. If not, return to step 4. Otherwise, the data in $singleMap$ will be removed and included;
7	Add the data in $singleMap$ to $featureMap$;
8	Judge whether the traversal of $prunSet$ is finished. If not, return to step 3. Otherwise, filter the support of $featureMap$;
9	De duplicate $featureMap$;
10	At the end, the $featureMap$ is output as the final frequent itemset;

From Table 1, it can be seen that after data processing, OFS algorithm is simpler, but its efficiency is not low. This is because, after data preprocessing, the initial ideal string is obtained, and the minimum frequent substring length is used when data is processed, which is often accompanied by a continuous nature, so this operation can further narrow the ideal string from preprocessing, and it often excludes a large amount of useless information. This makes the subsequent operation very efficient. In contrast, data preprocessing only uses fixed locations, and the ideal string computed does not utilize the continuity property, so the ideal string obtained has a large range and contains a lot of useless information. However, both data preprocessing and data reprocessing are indispensable, and it is these two key operations that greatly improve the efficiency of the algorithm.

Get Itemset Procedure. As you can see from the process, *prunSet* store all the ideal set of strings, so the process of getting frequent substrings is naturally also from the ideal set. Assuming that an ideal string is "0100001000100101 01#20" and the string intercepted from a data set in the data frame collection *dataSet* is "01000010101#20", you can see that the two strings are only different at 29 positions. Then, from this data, you can get two substrings of "010000100#20" and "10010101#30". At this time, you put them into *singleMap*, and then intercept and compare all data frames of *dataSet*, the ideal set of substrings can be obtained. A *singleMap* of all ideal strings can be obtained by acquiring all ideal strings.

Remove Inclusion Operation. After getting the itemset of the ideal string, you get a *singleMap* belonging to the ideal string. The *singleMap* needs to be stripped of inclusions at this point, considering three scenarios.

After Inclusion: In an ideal string, the following two substrings appear: "0001001011#302", "01001011#304". Obviously, the substring at position 304 is a true suffix of the substring at position 302, which is called After inclusion.

Pre Inclusion: In an ideal string, there are two substrings as follows: "1111 111110100#361", "11111111101#361". Obviously, the latter is a true prefix of the former, then this case is called pre inclusion.

Mutual Inclusion: In an ideal string, the following two substrings appear: "0100010010#156", "0001001010#158". Obviously, if a true prefix of the latter is a true suffix of the former, the case is called mutual inclusion.

After inclusion will lead to the number of substring statistics error, resulting in frequent substring missing. Because they are counted individually in the *singleMap*. In an extreme case, "0001001011" appears in the first 50% of the data frame set dataset, while "01001011" appears in the last 50% of the data frame set. If Min_Sup is 0.7, then both substrings cannot be used as frequent substrings. However, the string "01001011#304" is obviously a characteristic string, because it actually appears in 100% of the data. Therefore, when dealing with such cases, it is necessary to add the number of times of the string "0001001011"

in the *singleMap* to the string "01001011#304", so that the statistics are complete. Similarly, for post inclusion, you need to add the number of times a longer substring has in the *singleMap* to another substring. For mutual inclusion, we need to add location information to the mutual inclusion part of two strings to form a new substring, and add the times of both in *singleMap* to the new substring. Before processing these three situations, the *singleMap* is copied to a tmp *SingleMap*. Whether the number of times is increased or the new string is added, the *singleMap* needs to be updated after processing.

Get Frequent Substring. After all the ideal strings in the *prunSet* get item set and remove inclusion operations, each substring and corresponding times in the *singleMap* of each ideal string will be added to the $featureMap$.

After that, the support of each substring in the $featureMap$ is calculated, and all the substrings with support less than Min_sup are deleted.

At this time, it is possible that the substrings in the $featureMap$ are duplicated due to the inclusion conditions mentioned above. For example, in an extreme case, if the supports of strings "1111111110100" and "11111111101" are greater than the minimum support, then both of them will not be removed. But obviously, the same position of the string, only need to leave a longer. So at this time, delete the short string. It can be seen that when it is not necessary to remove the inclusion as mentioned above, the judgment is divided into several situations. The processing method is relatively simple. You only need to judge whether there is a substring included in another substring. If so, you just need to delete the shorter string, such as "0001001011#302" , and "01001011#304" , and just delete "01001011#304".

At this point, the final frequent itemset of the data frame set dataset has been obtained.

However, the generation of association rules still follows the analysis method of association rules of Apriori algorithm.

3.3 Algorithm Evaluation

Evaluating an algorithm needs to be judged from many perspectives. The most common means are to calculate the time and space complexity of the algorithm.

Time Complexity. Suppose the data frame collection *dataSet* has n data frames and the average length of the data frame is m. Then first iterate through the *dataSet* to initialize the vector $localVector$ with $O(mn)$ time complexity. The ideal set of strings, *prunSet*, is obtained through the $localVector$ with $O(m)$ time complexity. Then all the ideal strings in the *prunSet* add up to no more than m, and each ideal string in the *prunSet* is compared with the dataset to get a substring, which has an $O(mn)$ time complexity. Overall, the final time complexity of the algorithm is $O(mn)$. This also demonstrates the superiority of the new algorithm.

Spatial Complexity. All operations are $localVectors$ based on the initial data preprocessing, so all subsequent operations will no longer exceed the $localVector$, so the spatial complexity of the algorithm is $O(m)$.

4 Experimental Results and Analysis

The details of seven sets of data frame sets are introduced. Seven sets of data frames are different protocol files. Among them, DNS protocol file size is 9384 kb, HTTP protocol file size is 43642 kb, HTTP2 protocol file size is 46337 kb, ICMP Protocol file size is 750 kb, ICMP2 protocol file size is 6071 kb, OICQ protocol file size is 20119 kb, SSDP protocol file size is 6839 kb. The detailed data set is shown in Table 2. The specific size and running time of the two algorithms are detailed in Table 3 (the data in the table are arranged in ascending order of file size). The seven groups of data are protocol data intercepted from Wireshark. The running time of the two algorithms comes from the execution time of the console program.

Table 2. Protocol data set.

Protocol type	Total number of data frames (Pieces)	Total data frame size (KB)
ARP-like protocol	30000	2639
DNS-like protocol	113000	9384
HTTP-like protocol	230000	43642
HTTP2-like protocol	238000	46337
ICMP-like protocol	10000	750
ICMP2-like protocol	31000	6071
OICQ-like protocol	267000	20119
SSDP-like protocol	108000	6839
TCP-like protocol	30000	2595
UDP-like protocol	30000	2565
Train Set	240000	20834

Table 3. Running time comparison of CFI algorithm and OFS algorithm

File size (KB)	750	6071	6839	9384	20119	43642	46337
CFI time (s)	9.2	160.2	2198.4	77.0	391.7	508.9	583.5
OFS time(s)	0.4	1.3	0.5	1.7	3.8	16.6	18.2

In Table 3, due to space constraints, only one decimal place is reserved according to the rounding principle. We can see the superiority of OFS algorithm. In

the SSDP protocol file, the CFI algorithm duration is 2198.4 S. Considering the overall situation, the running time data of SSDP is regarded as a bad data. Then we can draw a broken line diagram of the two algorithms, and we can see the difference between them more clearly, as shown in Fig. 2.

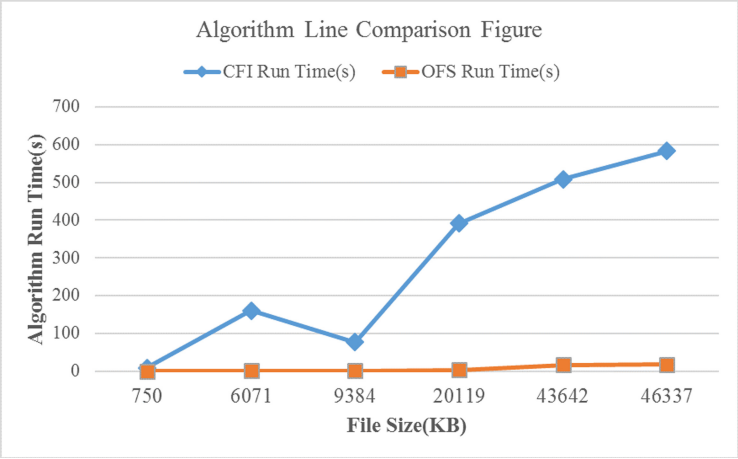

Fig. 2. Broken line comparison between CFI algorithm and OFS algorithm.

In the broken line comparison diagram in Figs. 4 and 5, it can be clearly seen that the running time of OFS algorithm is only 34.9 s even if the http2 protocol frame set with the largest dataset has 46337 kb data, which fully illustrates the advantages of OFS algorithm.

After feature extraction of the input data, the extracted feature string will be saved. The result of the feature string is shown in Fig. 3.

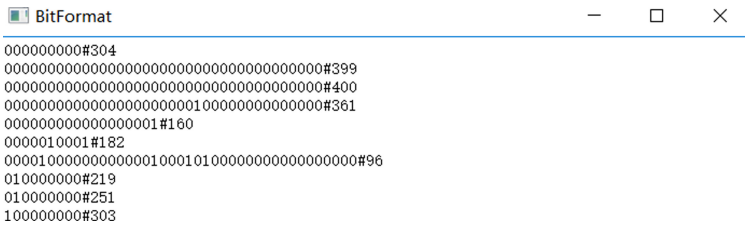

Fig. 3. Display of feature extraction results.

From the Fig. 3, we can clearly see the result of protocol features. The left side of the protocol features is the feature string, and the right side is the feature string position. The picture shows that the method can achieve good results.

The OFS algorithm is embedded into the unknown protocol syntax reverse analysis system, and different similarity is output according to different support degrees. The results are shown in Fig. 4. When six protocols are selected, ICMP is the result when the support degree is 0.60, ICMP2 is the result when the support degree is 0.65, ICMP3 is the result when the support degree is 0.70, and ICMP4 is the result when the support degree is 0.75. It can be seen from the figure that with the increase of support, the similarity increases gradually.

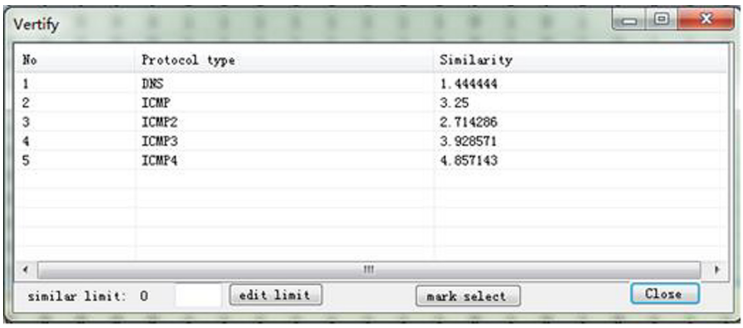

Fig. 4. Comparison of the results of different supports.

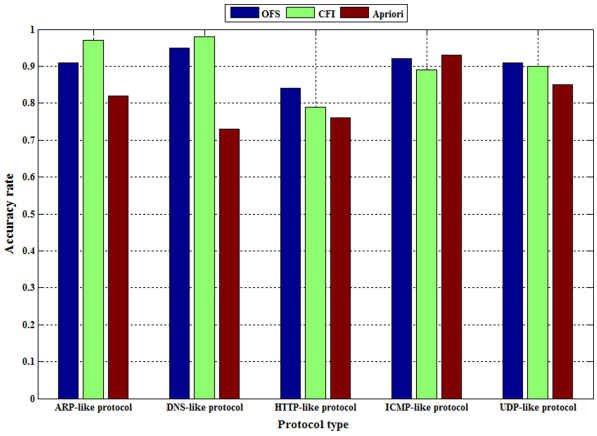

Fig. 5. Protocol accuracy comparison figure.

The Fig. 5 above shows the comparison of protocol accuracy. From the figure, we can see that although the running time of ofs is greatly reduced, its accuracy rate is basically not affected, which is equivalent to that of CFI algorithm. Therefore, it has certain advantages to apply the algorithm to the identification of unknown protocols.

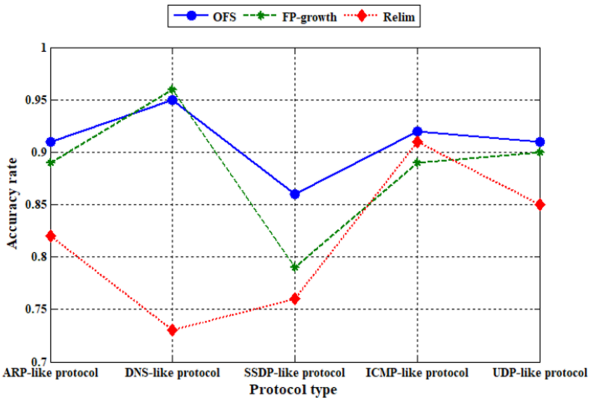

Fig. 6. Protocol accuracy comparison.

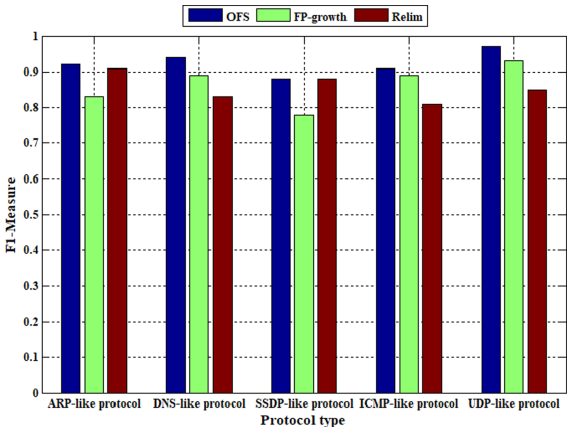

Fig. 7. Protocol F1-measure comparison.

Figures 6 and 7 are the experimental charts of accuracy and F1 value comparison of OFS algorithm with FP-growth algorithm and Relim algorithm. From the accuracy comparison chart of Fig. 6, it can be seen that the accuracy of OFS algorithm is more stable than other algorithms. From the F1 value comparison

chart of Fig. 7, it can be seen that the performance evaluation of OFS algorithm is slightly higher than FP-growth algorithm and relim algorithm, about 5% higher than FP-growth algorithm.

5 Conclusion

In this paper, the bitstream data frame set is taken as the research object. The main work is to extract the feature string from the data frame set, and put the feature string into the database. If there is a new data frame, the data frame will be compared with the features in the library, and finally the protocol of the data frame can be identified. In this paper, the Apriori algorithm is improved, and the CFI algorithm is used to find the feature string. Combining the advantages of the previous algorithms, a more efficient OFS algorithm is proposed, which has better performance in the face of feature extraction of bitstream protocol. In this paper, based on OFS algorithm, the frequent itemsets of the data frame set are extracted more efficiently, and the strong association rules of the items in the frequent substring itemsets are analyzed by the Apriori algorithm, and the frequent substrings with low recognition degree are removed, and the more representative frequent subsequence sets are stored in the agreement feature database. Experimental results show that OFS has a good effect on protocol reverse analysis, and greatly speeds up the efficiency of the algorithm based on the original CFI algorithm.

Acknowledgment. This research work is supposed by the National Joint Funds of China (U20B2050), National Key R&D Program of China(2018YFB1201500), National Natural Science Founds of China (62072368, 61773313, 61702411), National Natural Science Founds of Shaanxi (2017JQ6020, 2016JQ6041), Key Research and Development Program of Shaanxi Province (2020GY-039, 2017ZDXMGY-098, 2019TD-014).

References

1. Zhou, Z., Zhao, L.: Cloud computing model for big data processing and performance optimization of multimedia communication. Comput. Commun. **160**, 326–332 (2020)
2. Jiang, Y.: Wireless resource management mechanism with green communication for multimedia streaming. Multimedia Tools and Applications **78**(7), 8699–8710 (2018). https://doi.org/10.1007/s11042-018-6149-4
3. Hei, X., Bai, B., Wang, Y., et al.: Feature extraction optimization for bitstream communication protocol format reverse analysis. In: 2019 18th IEEE International Conference on Trust, Security and Privacy Iin Computing and Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE), pp. 662–669 IEEE (2019)
4. Dai Y.F.: Research and application of apriori algorithm in analysis of students' achievement. In: Information Engineering Research Institute, USA. Proceedings of 2012 Third International Conference on Theoretical and Mathematical Foundations of Computer Science (ICTMF 2012), vol. 38, pp. 894–899. Information Engineering Research Institute (2012)

5. Yang, H., Li, P., Zhu, Q., et al.: The application layer protocol identification method based on semisupervised learning. IEEE **6**, 115–120 (2011)
6. Yuan, Z., Xue, Y., Dong, Y.: Havesting unique characteristics in packet sequences for effective application classification. In: Proceedings of the 1st IEEE Conference on Communications and Network Security, pp. 341–349. IEEE Communication Society, Los Alamitos (2013)
7. Mohsin, A.H., Bakar, K.A., Zainal, A.: Optimal control overhead based multi-metric routing for MANET. Wirel. Netw. (2), 1–17 (2017)
8. Naseem, M., Kumar, C.: Queue-based multiple path load balancing routing protocol for MANETs. Int. J. Commun. Syst. **30**(6), e3141 (2017)
9. Pradittasnee, L., Camtepe, S., Tian, Y.C.: Efficient route update and maintenance for reliable routing in large-scale sensor networks. IEEE Trans. Ind. Inf. **13**, 144–156 (2016)
10. Zheng, J.: Research on a cluster system for binary data frames of wireless sensor network. Cluster Comput. **19**(2), 783–791 (2016)
11. Tao, L.: Packet signature mining for application identification using an improved apriori algorithm. Nanjing University of Science and Technology, Shanghai University of Finance and Economics. Proceedings of 2015 IEEE International Conference on Progress in Informatics and Computing (PIC 2015 V1). Nanjing University of Science and Technology, Shanghai University of Finance and Economics, IEEE Beijing Section, pp. 664–668 (2015)
12. Lin, Y., Nie, Z., Ma, H.: Structural damage detection with automatic feature-extraction through deep learning. Comput. Aided Civ. Infrastruct. Eng. **32**(12), 1025–1046 (2017)
13. Song, Z., Wu, B.: Anomaly detection based on feature extraction of unknown protocol payload format. In: 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC), pp. 704–709. IEEE (2020)
14. Sengupta, S., Chowdhary, A., Sabur, A., et al.: A survey of moving target defenses for network security. IEEE Commun. Surv. Tutorials **22**, 1909–1941 (2020)
15. Jeong, C., Ahn, M., Lee, H., Jung, Y.: Automatic classification of transformed protocols using deep learning. In: Park, J.H., Shen, H., Sung, Y., Tian, H. (eds.) PDCAT 2018. CCIS, vol. 931, pp. 153–158. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-5907-1_16
16. Tang, H., Xiao, B., Li, W., Wang, G.: Pixel convolutional neural network for multi-focus image fusion. Inf. Sci. **433**, 125–141 (2018)
17. Xiao, B., Wang, K., Bi, X., Li, W., Han, J.: 2D-LBP: an enhanced local binary feature for texture image classification. IEEE Trans. Circuits Syst. Video Technol. **29**(9), 2796–2808 (2019)
18. Shen, Z., Lee, P.P., Shu, J., et al.: Encoding-aware data placement for efficient degraded reads in XOR-coded storage systems: algorithms and evaluation. IEEE Trans. Parallel Distrib. Syst. **29**(12), 2757–2770 (2018)
19. Cheng, Y., Wang, F., Jiang, H., et al.: A communication-reduced and computation-balanced framework for fast graph computation. Front. Comput. Sci. Chin. **12**(5), 887–907 (2018)
20. Lin, B., Guo, W., Xiong, N., et al.: A pretreatment workflow scheduling approach for big data applications in multicloud environments. IEEE Trans. Netw. Serv. Manage. **13**(3), 581–594 (2016)
21. Wang, J., Zhang, X., Lin, Y., et al.: Event-triggered dissipative control for networked stochastic systems under non-uniform sampling. Inf. Sci. **447**, 216–228 (2018)

22. Zou, J., Dong, L., Wu, W.: New algorithms for the unbalanced generalized birthday problem. IET Inf. Sec. **12**, 527–533 (2018). https://doi.org/10.1049/iet-ifs.2017.0495
23. Rawat, D.B., Garuba, M., Chen, L., et al.: On the security of information dissemination in the internet-of-vehicles. Tsinghua Sci. Technol. **22**(4), 437–445 (2017)
24. Radhakrishna, V., Aljawarneh, S., Kumar, P.V., et al.: A novel fuzzy gaussian-based dissimilarity measure for discovering similarity temporal association patterns. Soft Comput. **22**(6), 1903–1919 (2018)
25. Wang, C., Zheng, X.: Application of improved time series apriori algorithm by frequent itemsets in association rule data mining based on temporal constraint. Evol. Intell. **13**(1), 39–49 (2020)
26. Cui, W., Kannan, J., Wang, H.J.: Discoverer: automatic protocol reverse engineering from network traces. In: Proceedings of the 16th USENIX Security Symposium, pp. 199–212. IEEE (2007)
27. Trifilo, A., Burschka, S., Biersack, E:. Traffic to protocol reverse engineering. In: Proceedings of the Second IEEE International Conference on Computational Intelligence for Security and Defense Applications, pp. 1–8. IEEE (2009)
28. Sabokrou, M., Fayyaz, M., Fathy, M., et al.: Deep-anomaly: fully convolutional neural network for fast anomaly detection in crowded scenes. Comput. Vis. Image Understand. **172**, 88–97 (2018)
29. Giuseppe, S., Massimiliano, G., Antonio, M., et al.: A CNN-based fusion method for feature extraction from sentinel data. Remote Sens. **10**(2), 236 (2018)
30. Kumar, B., Anand, D.K., Anjanappa, M., et al.: Feature extraction and validation within a flexible manufacturing protocol. Knowl. Based Syst. **6**(3), 130–140 (1993)
31. Yi, L., Hua, P., Zhen-Hua, Z.: Protocol recognition feature extraction algorithm of high frequency communication signals based on wavelet de-noising. J. Inf. Eng. Univ. **13**, 438–442 (2012)
32. Ding-Ding, F., Rong-Feng, Z., An-Min, Z.: Protocol feature extraction and anomaly detection based on flow characteristics of industrial control system. Mod. Comput. (2019)
33. Lin., C.C., Wang, C.N., et al.: Combined image enhancement, feature extraction, and classification protocol to improve detection and diagnosis of rotator-cuff tears on MR imaging. Magn. Reson. Med. Sci. Mrms Official J. J. Soc. Magn. Reson. Med. **13**, 155–166 (2014)
34. Xiong, J., Bi, R., Zhao, M., Guo, J., Yang, Q.: Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles. IEEE Wirel. Commun. **27**(3), 24–30 (2020)
35. Tian, Y., Wang, Z., Xiong, J., Ma, J.: A blockchain-based secure key management scheme with trustworthiness in DWSNs. IEEE Trans. Ind. Inf. **16**(9), 6193–6202 (2020)
36. Xiong, J., et al.: A personalized privacy protection framework for mobile crowdsensing in IoT. IEEE Trans. Ind. Inf. **16**(6), 4231–4241 (2020)
37. Xiong, J., Zhao, M., Bhuiyan, M.Z.A., Chen, L., Tian, Y.: An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT. IEEE Trans. Ind. Inf. **17**(2), 922–933 (2021)
38. Xiong, J., et al.: Enhancing privacy and availability for data clustering in intelligent electrical service of IoT. IEEE Internet Things J. **6**(2), 1530–1540 (2019)