Research Article

From Unknown to Similar: Unknown Protocol Syntax Analysis for Network Flows in IoT

Yichuan Wang,1,2 Han Yu,1,2 Xinhong Hei,1,2 Binbin Bai,1,2 and Wenjiang Ji1,2

1Xi’an University of Technology, School of Computer Science and Engineering, Xi’an, China
2Shaanxi Key Laboratory for Network Computing and Security Technology, Xi’an, China

Correspondence should be addressed to Xinhong Hei; heixinhong@xaut.edu.cn

Received 7 May 2021; Revised 20 June 2021; Accepted 17 July 2021; Published 2 August 2021

Copyright © 2021 Yichuan Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Internet of Things (IoT) is the development and extension of computer, Internet, and mobile communication network and other related technologies, and in the new era of development, it increasingly shows its important role. To play the role of the Internet of Things, it is especially important to strengthen the network communication information security system construction, which is an important foundation for the Internet of Things business relying on Internet technology. Therefore, the communication protocol between IoT devices is a point that cannot be ignored, especially in recent years; the emergence of a large number of botnet and malicious communication has seriously threatened the communication security between connected devices. Therefore, it is necessary to identify these unknown protocols by reverse analysis. Although the development of protocol analysis technology has been quite mature, it is impossible to identify and analyze the unknown protocols of pure bitstreams with zero a priori knowledge using existing protocol analysis tools. In this paper, we make improvements to the existing protocol analysis algorithm, summarize and learn from the experience and knowledge of our predecessors, improve the algorithm ideas based on the Apriori algorithm idea, and perform feature string finding under the idea of composite features of CFI (Combined Frequent Items) algorithm. The advantages of existing algorithm ideas are combined together to finally propose a more efficient OFS (Optimal Feature Strings) algorithm with better performance in the face of bitstream protocol feature extraction problems.

1. Introduction

As the global economy continues to develop, the impact of scientific and technological advances on the daily lives of people around the world is gradually increasing. The Internet of Things (IoT) technology, which is derived from the advancement of science and technology, has also been developed significantly and has been applied in various industries around the world. The Internet of Things (IoT) has emerged in the context of information technology development, and its degree of development has been influenced by the processing power of information technology in the information age [1–4]. The popularization and development of IoT technology mark the comprehensive and integrated development of network information technology for the whole human race, which has laid a solid hardware foundation for the complete interconnection of all countries around the world.

IoT technology is characterized by a major shift in the mode of accessing and applying information among people, as well as a gradual change and subversion of people’s behavioral patterns such as clothing, food, housing, and transportation [5–8]. Especially, in the fields of smart home, autonomous driving, and health care, IoT is already quietly changing people’s lifestyles and will even have a further impact on all details of human life in the future [9–12]. It is worth everyone’s attention that despite the good prospects and speed of development of IoT technology in developing and developed countries around the world in recent years, cybersecurity issues between IoT devices are frequent, which can have a certain impact on the global economic and technological development [13, 14].

As economic globalization continues, network communication technology has turned the whole world into a global village. Currently, the number of Internet users
worldwide has exceeded 5 billion, which shows that Internet devices are essential in everyone’s daily life, and communication protocols play a very important role as a bridge for data communication between networked devices, so the classification and identification of these communication protocols has been a popular topic. Moreover, various network security incidents have appeared in people’s view recently, and the endless malicious network attacks have brought a lot of economic losses and psychological panic to people [15, 16]. Whether the communication security of the network can be ensured is related to the fundamental interests of individuals, enterprises, and the country. It is significant to carry out reasonable and effective network maintenance and network regulation at this time [17-19]. Therefore, it is necessary to analyze and identify the unknown protocols in the network in order to better regulate the network security.

The three basic elements of network protocols are semantics, syntax, and timing. The inference of the protocol message format and the determination of its field contents belong to the protocol syntax analysis [20]. The analysis and extraction of protocol semantics is the basis of protocol analysis and identification. It requires analysis of the control statements of protocol messages and extraction of the protocol semantics based on data mining and sequence ratio methods. The purpose of protocol syntax rule inference is to build a logical model of the protocol syntax, focusing on the intrinsic logical relationships between protocol messages. How protocols interact must follow certain syntax rules.

Analyzing the role of network protocol specifications in the field of network regulation can help us to obtain information about the network traffic in the target network [21, 22]. By classifying the traffic generated by these protocols, network usage can be identified, network expansion plans can be developed, and bandwidth for specific protocols can be controlled. Protocol analysis can provide useful information to firewalls and intrusion detection prevention systems to help analyze network vulnerabilities and thus prevent and detect unknown attacks.

The bitstream protocol format analyzer works at the bottom of the network environment, analyzing the content of the acquired bitstream protocol data in real time by parsing the data and analyzing the protocol format. The current network protocol analysis method, with the huge number of analyzed protocol frames and the complexity of the data frames themselves, can take a long time for the algorithm to run, and how to optimize the algorithm is a research direction that needs to be continuously studied.

In general, the rapid development of IoT has brought us some opportunities and challenges, and the security of communication between IoT devices is now the biggest challenge; specifically, there are still a lot of defects about the unknown protocol analysis. When processing a large number of protocols, the complexity of data processing is large and the system response speed is still slow and needs to be optimized in order to play a role in the actual scenario. It needs to be optimized to be useful in practical scenarios. When capturing data, there may be more than one protocol type, the length may be inconsistent, and many protocol identification methods will be greatly affected. Therefore, to propose an algorithm to solve the problem of intelligent inverse analysis of unknown protocols in connection with practical difficulties is the main research of this paper.

The rest of this paper is organized as follows: the first part introduces the current state of development and security issues of IoT technology worldwide, and the second part presents our work related to the unknown protocol parsing. The third part proposes a new protocol format analysis algorithm. The fourth part analyzes the performance of the new algorithm from several aspects and compares it with other algorithms. The fifth part concludes the work.

### 2. Related Work

Most of the existing studies on protocol identification have been based on content, port, and behavior-based protocol identification techniques [23]. The earliest protocol identification method used is the port number-based protocol identification technique. Port number-based protocol identification is well guaranteed in terms of correctness and efficiency in identifying traditional TCP/IP protocols. The algorithm principle of this technique is to use the service port number of TCP/IP protocol to identify the underlying protocol, and then compare the identified port number with the port number issued by IANA (Internet Assigned Numbers Authority) [24], and find the correspondence between the port and the protocol by cross-referencing to know the identified protocol’s type. However, this technique also has certain defects because the port numbers managed by IANA are not all static, and some port numbers are dynamic, and dynamic port numbers can be easily controlled by Trojan horse programs to carry out network attacks and endanger the security of the Internet. With the continuous development of the Internet, new protocols are born that tend to use dynamic port numbers and no longer use IANA to register port numbers, at which time port-based protocol identification methods are no longer efficient and accurate. The reasons for the failure of this technique have been analyzed in the related literature because the lack of necessary semantic information and corresponding protocol specifications for unknown bitstream protocols, not to mention the unavailability of any information about the protocol ports, makes the port number-based protocol analysis technique nowadays not applicable to the field of reverse identification of unknown protocols for bitstreams.

Multiple sequence comparison techniques in genetics [25-28] can extract similar segments in DNA, and in the field of bitstream protocol, reverse identification also requires the extraction of format-specific message segments from messages; therefore, multiple sequence comparison techniques can also be applied to protocol format inference, where researchers infer and analyze message format information by extracting variable and immutable fields in identified messages. Pi, ScriptGen, Discoverer, and Netzob use bioinformatics-based sequence matching techniques to determine message similarities and cluster them. They, then, separate messages by identifying common parts between messages in the same group. The amount of data has a
significant impact on the quality of the protocol specification, but multiple sequence matching has exponential complexity because sequence matching algorithms use only two messages at a time as input [29].

Zhang et al. [30–32] studied and proposed a feature extraction method combining multipattern matching and association rules by investigating the bitstream protocol feature extraction technique to divide the bitstream protocol multiprotocol data frames into single-protocol data frames. The work is done for offline data, which cannot meet the real-time nature of bitstream data analysis and identification. Moreover, although the accuracy of the method is high, it still takes a long time to run for a large amount of data centrally and consumes more resources.

Youxiang et al. proposed a semiautomatic protocol inverse analysis method based on artificial knowledge [33], which suggests that sociological engineering and artificial guessing can be used to obtain a priori knowledge such as “field semantics,” length, and boundaries in the process of protocol format identification. The method first separates the segments of the message sequence associated with the a priori knowledge and then uses them as the basis for subsequent format inference, deriving the semantic inference of the fields using the a priori knowledge and verifying the results. After experimental validation, it is concluded that the method can verify the semantics and format of the obtained data and improve the accuracy of the initial clustering, thus greatly improving the accuracy of format inference for unknown protocols.

Xiao-Li et al. conducted intensive research on existing protocol recognition techniques [34–37] to enumerate the algorithms and principles related to pattern matching and data mining by analyzing their strengths and weaknesses. A detailed presentation is also provided for the analysis of a considerable amount of bitstream data using the relevant theories of data mining to analyze the meaning relationships in order to find all possible candidate strings. Then, pattern matching algorithms are used for further analysis.

In addition to sequence comparison techniques, data mining techniques can also be used to perform inverse analysis of protocols. Karimov et al. used the Apriori algorithm to extract the keywords and message format of the protocol. The Aho-Corasick algorithm was first used for keyword extraction of protocols [38], followed by the frequent pattern FP-Growth algorithm to extract the format of messages [39]. Unlike the sequence comparison technique, the data mining technique takes all messages as input at once, which directly leads to a large computational cost for candidate selection. In addition, it is crucial to learn how to optimize the results to make them intuitive and clear.

3. Protocol Feature Extraction Algorithm

In order to avoid and improve the shortcomings of the algorithm described in the previous section, OFS, a protocol feature extraction algorithm based on the idea of the Apriori algorithm and the idea of composite features of CFI, is proposed, which is different from the idea of CFI algorithm. The previous algorithm tends to iterate to find feature strings from nothing, while the OFS algorithm tends to find the range of possible feature strings at one time and then go to search feature strings within the range. This chapter will introduce the idea and steps of the OFS algorithm.

3.1. Algorithm Ideas

3.1.1. Algorithm-Related Definitions. To better illustrate the algorithm, some definitions are introduced and presented here.

Definition 1. Minimum support: A reasonable threshold defined by the user to measure the magnitude of support, which in a statistical sense represents the minimum standard of the importance of the data; here, we use Min_sup to represent it.

Definition 2. Frequent substring: If there are N data frame messages, these are sequences of bits of length L1, there exists a substring α of length L2 (L1 > L2), and if substring α has occurrences in M of the N data frames, the probability of occurrence of α is P(M/N). If the probability of a substring occurring is greater than or equal to Min_Sup, then the string is called a frequent substring:

\[ \text{Seq} = \{ \alpha | P(\alpha) \geq \text{Min}_\text{sup} \} \]  \hspace{1cm} (1)

Definition 3. Minimum frequent substring length: A user-defined value where the length of a frequent substring is filtered out if it is less than the length of the least frequent substring, denoted as Min_sup.

Definition 4. Protocol feature: If frequent substring α appears frequently at one or more specific locations in the protocol data frame, it is considered likely that the frequent substring is a protocol feature of the protocol.

3.1.2. Algorithm Data Initialization. Algorithm data initialization is a five-step process:

1. Enter the support threshold Min_sup, traverse the data set, and record the length of the longest data in the data set as Min_sup (the length of the longest data in the data set).
2. Define a one-dimensional vector Vector and initialize all elements of it to 0 with Max_len.
3. Traverse all the data frames in the data set and record whether the elements of each position of each data are 0. If the value of the data at a position is 0, let the value of the one-dimensional vector at the corresponding position of that data be added by 1. For example, if the i-th position data [i] of a data is equal to 0, then the value of the one-dimensional vector at its corresponding position is added by 1, that is, Vector [i] is added by 1.
4. The support of each position is calculated by traversing the vector Vector once. If the position support sup ≥ Min_Sup or sup ≤ 1 − Min_Sup
(assuming Min_Sup > 0.5), it means that the position may exist in a feature string; it is impossible for the position to exist in a feature string.

After calculating the support for each position, two more important definitions need to be stated to complement the data initialization process of the algorithm, Definition 5 and 6.

**Definition 5.** Bad characters: If the support of a position is not within the range specified in step (4), the character at that position is considered to be a bad character and is denoted as $C_i$.

**Definition 6.** Ideal string: The substring that appears between two adjacent bad characters in a one-dimensional vector Vector is called the ideal string. If there is only one bad character $B_1$ in a certain data frame, the substring from the beginning of the vector Vector0 all the way to $B_1$ (containing the character at 0 but not at $B_1$) is considered as the ideal string and similarly from $B_1$ all the way to the end of the vector Vector is also considered as an ideal string. The minimum frequent substring Min_len can be used to filter part of the ideal string.

(5) After the processing of the above steps, all ideal strings will be obtained by Vector and the position of bad characters, and then these ideal string records are put into a set prun Set (the set of ideal strings).

3.1.3. Data Reprocessing. After the initialization of the algorithm data, we get the set prun Set after preprocessing; the data set contains all the possible locations of the feature string, but the range of the occurrence of each feature string is too large, which is not convenient enough for the specific search of the feature string afterwards. Because this operation of frequency statistics for each position ignores the continuity of the string, the range obtained is relatively large, so we use the continuous property of the string to perform in processing in a good way; the specific steps are as follows:

(1) Iterate over each data Str (string in the ideal string set) of the data set prun Set to get the Str length of each data and use this length to build a one-dimensional vector Vector, so that its value is 0.

(2) Reiterate the data set date set (the original data set), intercept the string date (the string in the original data set) with the same length and the same position as Str in the data set, cut Str and date using the length of the least frequent substring Min_len, and judge whether they are equal. If they are equal, then the one-dimensional vector Vector[i] corresponding to the cut position is added by one; if not equal, there is no operation.

(3) Then, we refer to steps 3, 4, and 5 of the algorithm data initialization to obtain the updated prun Set, and the data processing operation of the algorithm is completed.

3.2. Algorithm Flow. The entire flow of the algorithm is described in Table 1, and Figure 1 shows the algorithm flowchart.

From Table 1, we can see that the OFS algorithm divides the data processing into two stages: preprocessing and reprocessing. Although the data is preprocessed to obtain the approximate range of the ideal string, only the fixed position of the ideal string is obtained without using the continuity of the ideal string, so the range of the ideal string is very large, which contains a large amount of useless information. Therefore, the role of data reprocessing at this time is to use the definition of the minimum frequent substring length to further reduce the range of the ideal string obtained by preprocessing, the operation will eliminate a large number of useless information, making the subsequent data operations much more efficient.

3.2.1. The Process of Obtaining the Set of Items. From the above operation, we can conclude that prun Set is a collection of all ideal strings, so naturally the frequent substrings must also be obtained from the ideal string. Suppose a certain ideal string is “001000100010001001001#47”; the string intercepted from the corresponding position of the data frame set date set is 001000100010001001001#47. By comparison, we can see that the two strings differ only in the characters of position 56. The comparison shows that the two strings differ only in the characters at position 56. Since this is the case, we can separate the two substrings “0010001000#47” and “0010001000#57” from this data. Put them into a new set of single Map (ideal string of substring collection); all the data frames of the data frame collection date Set to intercept and compare the separation can get the ideal string of the substring collection and all the ideal string to obtain the operation can get all the ideal string of single Map.

3.2.2. Removing the Include Operation. After getting the item set of the ideal string, single Map belonging to the ideal string is obtained, and single Map needs to be removed from the containment operation. In an ideal string, there are two substrings: “00010001001#223” and “00010001001#226.” Obviously, the substring at position 226 is a true suffix of the substring 223, and the case becomes a postinclusion. Similarly, if a substring is a true prefix of another substring, it is a preinclusion. If the true prefix of a substring is the true suffix of another substring, the case is called mutual inclusion. Postinclusion can cause the number of substrings to be counted incorrectly, which can lead to missing frequent substrings. This is because the counts of them are counted separately in single Map. Consider an extreme case where “00100010010#402” appears in the first 50% of the data frames of the data set and “00101010#406” appears in the last 50% of the data frames of the data set. If Min_sup is 0.7 at this time, then both substrings cannot be used as frequent substrings. However, the string “00101010#406” is obviously a feature string, because it actually appears in 100% of the data. Therefore, when dealing with such cases, we need to add the number of times the string “00100010010#402” is in the
Table 1: Description of OFS algorithm.

| Input                                                                 | Output                                      |
|-----------------------------------------------------------------------|---------------------------------------------|
| The ideal string set of data preprocessing is prunSet; the data frame set is dataSet; the minimum support is Min_Sup | Frequent item set feature Map                |
| (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) Deduplicate featureMap                                              |                                              |
| (10) In the end, the featureMap is output as the final frequent item set |                                              |

Figure 1: Alerts filter and identification model.
single Map to the string “0010110#406”, so that the statistics are complete. Similarly, for postinclusion, the number of times the longer substring is in the single Map should be added to the other substring. For mutual inclusion, we need to intercept the mutual inclusion part of two strings plus position information to form a new substring and add the number of times both are in the single Map to the new substring.

Before handling these three cases, the single Map is copied to tmp Single Map (a collection of substrings of the temporary ideal string), and either adding times or adding new strings is done in tmp Single Map, so the single Map needs to be updated after processing.

3.2.3. Get Frequent Substrings. After all ideal strings in the prun Set are subjected to item set acquisition and inclusion removal operations, each substring and corresponding count in the respective single Map of each ideal string is added to the feature Map (feature set).

Then, the support is calculated for each substring in single Map, and all substrings with support less than Min_Sup are deleted.

For example, consider a case where both the string “000110100110#153” and the string “01110010110#153” have support greater than the minimum support. Neither of them will be removed, but it is obvious that for strings in the same position, only the longer ones should be left.

At this point, the final set of frequent items of data Set, a collection of data frames, has been obtained.

The association rule generation, however, still follows the association rule analysis method of the Apriori algorithm.

3.3. Algorithm Evaluation. Evaluating the merits of an algorithm requires several perspectives. The most common means is to calculate the time complexity and space complexity of the algorithm.

3.3.1. Time Complexity. Suppose the data Set has \( n \) data frames and the average length of the data frames is \( m \). Then, first iterate through the data Set to initialize the vector Vector with \( O(nm) \) time complexity. The time complexity is \( O(m) \) to obtain the ideal string set prun Set by Vector. Then, the length of all the ideal strings in prun Set does not exceed \( m \). For each ideal string of prun Set to compare with dataSet and get substrings, the time complexity of this operation is \( O(mn) \). Overall, the final time complexity of the algorithm is \( O(mn) \). This also shows the superiority of the new algorithm.

3.3.2. Spatial Complexity. Assuming that the average length of data frame is \( m \), the one-dimensional vector Vector is initialized with Max_len, and the length of Max_len is taken as the average length \( m \), so the one-dimensional vector Vector has \( m \) elements, all operations are based on the Vector obtained from the initial data preprocessing work, and the subsequent data reprocessing operations are all for the ideal string cross-matching work. Therefore, the space of all operations after data preprocessing does not exceed the Vector, so the space complexity of the algorithm is \( O(m) \).

4. Analysis of Experimental Results

The content focuses on testing the OFS algorithm to ensure the correctness of the algorithm. And the OFS algorithm is compared with the CFI algorithm to derive the correctness and superiority of the optimization direction of the OFS algorithm.

4.1. Support and Coverage Testing. This step focuses on testing the algorithm by two means. The first one is the extraction of frequent substrings from the OFS algorithm using the set of data frames, and then the extraction results are taken out for separate check counts, thus testing the correctness of the OFS algorithm for extracting frequent substrings in terms of support counts. In the second test, the OFS algorithm is compared with the CFI algorithm implemented to extract frequent substrings from the same set of data frames, and the results of the two algorithms are compared to see if the frequent item sets of the two algorithms are the same in number and correspond to each other. This further tests the correctness of the OFS algorithm in terms of the range and support of the extracted frequent substrings.

As shown in Table 2, the data shows the comparison of the frequent item set extraction results for DNS protocols using the two matching methods, and it is obvious from the corresponding entries that the algorithm results are consistent with the test results of the brute force method.

As shown in Table 3, the data shows the comparison of the frequent item set extraction results for the HTTP protocol using the two methods, and it is obvious from the corresponding entries that the algorithm results are consistent with the test results of the brute force method.

The test results from the comparison of the two sets of tables show that the OFS algorithm has the same results as after the brute force search. This indicates that the OFS algorithm possesses some correctness in counting the support of frequent substrings.

As shown in Table 4, the data shows the comparison of frequent item set extraction results for both protocols using the CFI algorithm; it can be seen that using the same data for CFI algorithm testing, the OFS algorithm and CFI algorithm extract the exact same frequent item set under the same condition of the data frame set, which shows that the coverage of OFS algorithm in terms of frequent substring acquisition is comprehensive and once again correct in terms of support counting.

4.2. Algorithm Time Comparison Analysis. We try to demonstrate whether the OFS algorithm has an advantage over the CFI algorithm in terms of recognition speed by comparing the time used for feature extraction using both OFS and CFI algorithms for seven different data frames. These seven data frames are the data sets of seven common communication protocols. The size of the ICMP protocol file
is 800 kB, QICQ protocol file is 20015 kB, DNS protocol file is 9264 kB, and SSDP protocol file is 6889 kB. The specific size of each protocol and the running time of both algorithms are detailed in Tables 5 and 6. All seven protocol data sets are intercepted by Wireshark and both algorithms are performed in CodeBlocks, and the runtimes are derived from the execution times of the console programs.

As shown in Table 6, we can see that the OFS algorithm has a significant advantage in speed compared to the CFI algorithm, but the CFI algorithm in the SSDP protocol file has a recognition time of 2095.6 s and combined with the overall data in Table 6 to compare, it is clear that the CFI algorithm for the SSDP protocol has too long a recognition time, so this time is defined as bad data. The running time of the two groups of algorithms is compared as a line graph, and the difference between the two can be seen more clearly. This is shown in Figure 2.

### 4.3. Algorithm Accuracy Comparison Analysis

Figure 3 shows the comparison results of the three algorithms for different protocols after the accuracy test, respectively. We know from Section 4.2 that the running time of the OFS algorithm is greatly shortened compared to the CFI algorithm, but it can be seen from Figure 3 that the accuracy of the OFS algorithm is still close to the CFI algorithm, so it can be seen that the OFS algorithm has a considerable advantage when performing unknown protocol analysis.

Figures 4 and 5 show the experimental plots comparing the $F_1$ values and accuracy of the OFS algorithm with the Relim algorithm and the FP growth algorithm.
Table 4: DNS protocol and HTTP protocol frequent item set extraction result table.

| Matching program | Frequent item set                                                                 | Quantity (pieces) |
|------------------|----------------------------------------------------------------------------------|-------------------|
| DNS protocol     | 000000000000000000000000#160                                                    | 24335             |
|                  | 000000000000000000000000#398                                                   | 17248             |
|                  | 000000000000000000000000#416                                                   | 19914             |
|                  | 00000000000000000000000000000000#399                                          | 18847             |
|                  | 000000000000000000000000000000000000000000000000000#361                         | 19084             |
|                  | 000100100100#183                                                               | 19580             |
|                  | 001000100000#184                                                               | 17182             |
|                  | 0100000000#302                                                                 | 17565             |
|                  | 01000100000000001001000000000000000000000#361                                 | 19603             |
|                  | 1000000000000000#303                                                           | 18269             |
| HTTP protocol    | 00000000000000000000000000000000000000000000000000000#160                       | 24335             |
|                  | 0000000000000000000000000000000000000000000000000000000000000000000000000000000#381 | 11511             |
|                  | 0000000000000000000000000000000000000000000000000000000000000000000000000000000#416 | 10701             |
|                  | 0000000000000000000000000000000000000000000000000000000000000000000000000000000#162 | 9725              |
|                  | 0000011000#184                                                                 | 10135             |
|                  | 00001010100110000000000010111190010000000000000#240                           | 10177             |
|                  | 010000000000000000000000#368                                                 | 10239             |
|                  | 0101000000#368                                                                 | 13688             |

Table 5: Protocol data set details.

| Protocol type   | Total number of data frames (pieces) | Total data frame size (kB) |
|-----------------|--------------------------------------|---------------------------|
| SSDP protocol   | 11687                                | 6889                      |
| QICQ protocol   | 47812                                | 20015                     |
| SSDP protocol   | 1705                                 | 800                       |
| ICMP2 protocol  | 9056                                 | 40642                     |
| HTTP protocol   | 9056                                 | 43337                     |
| HTTP2 protocol  | 9651                                 | 4337                      |
| DNS protocol    | 24340                                | 9364                      |
| Training set    | 125117                               | 126978                    |

Table 6: OFS algorithm and CFI algorithm running time comparison.

| File size (kB) | 6889 | 20015 | 800 | 6031 | 40632 | 43337 | 9264 |
|----------------|------|-------|-----|------|-------|-------|------|
| CFI algorithm time (s) | 2095.6 | 391.7 | 9.2 | 160.2 | 502.6 | 5730.4 | 77.0 |
| OFS algorithm time (s)   | 0.5   | 3.7   | 0.4 | 1.2  | 15.9  | 17.8  | 1.6  |

Figure 2: Algorithm running timeline comparison chart.
As can be seen from the accuracy comparison plot in Figure 4, the OFS algorithm is more stable than the other algorithms. From the $F_1$ value comparison plot in Figure 5, it can be seen that the OFS algorithm has a slightly higher performance evaluation than the FP growth algorithm and the REIM algorithm, which is about 4% higher than the FP growth algorithm.

### 4.4. Support and Similarity Tests

The OFS algorithm is embedded into an unknown protocol syntax inverse analysis system to detect the effect of the size of support on the number and length of feature strings in a protocol by performing feature extraction tests with different support degrees on a set of protocol data sets by the OFS algorithm.

#### Table 7: Comparison of feature extraction of the OFS algorithm with different support degrees.

| Support level | Feature string                                      |
|---------------|-----------------------------------------------------|
| 0.6           | 11111111*10.**6814                                  |
|               | 0100001010011001100#222.**10802                     |
|               | 00000000#258.**6814                                 |
|               | 0100000101001100100#302.**8627                      |
| 0.7           | 0100000101001100100#222.**10802                     |
|               | 0100000101001100100#302.**8627                      |
| 0.8           | 0100000101001100100#222.**10802                     |
|               | 0000101001100100100#304.**10802                     |

#### Table 8: Similarity comparison of 10 ARP protocol data frames with different support degrees.

| Protocol type | Similarity |
|---------------|------------|
| ARP (0.6)     | 4.4        |
| ARP (0.7)     | 6.67       |
| ARP (0.8)     | 7          |

#### Table 9: Similarity comparison of 100 ARP protocol data frames with different support degrees.

| Protocol type | Similarity |
|---------------|------------|
| ARP (0.6)     | 32.8       |
| ARP (0.7)     | 45.33      |
| ARP (0.8)     | 50         |

#### Table 10: Similarity comparison of 1000 ARP protocol data frames with different support degrees.

| Protocol type | Similarity |
|---------------|------------|
| ARP (0.6)     | 412.8      |
| ARP (0.7)     | 556        |
| ARP (0.8)     | 622        |

Table 7 shows the details of the feature strings obtained after feature extraction by the OFS algorithm for the same ARP protocol data set with support degrees of 0.6, 0.7, and
From Tables 7, it can be seen that when the support degree is 0.6, the protocol syntax inverse analysis system extracts 4 feature strings after feature extraction, and when the support degree is 0.7 and 0.8, the system can only extract 2 feature strings, and the feature strings at the support degree of 0.8 are shorter than those at the support degree of 0.7. Therefore, it can be concluded that the number of feature strings will gradually decrease as the support degree increases, and the length of the feature strings will also become shorter. Therefore, it can be concluded that the number of feature strings decreases as the support increases and the length of the feature strings becomes shorter, which proves that the core idea of the OFS algorithm is correct and can yield the expected results.

Tables 8–10 show that the corresponding feature strings are extracted by the OFS algorithm with different support degrees and then used to identify and match the protocols to obtain the matching similarity. In Tables 8–10, 10, 100, and 1000 data frames are randomly selected from 10802 data frames of the ARP protocol data set for similarity testing, and the results are shown in Tables 8–10. This also proves that the core idea of the OFS algorithm is correct and can achieve the expected results.

5. Experimental Development Configuration

The operation of the OFS algorithm relies on the Unknown Bitstream Protocol Intelligent Reverse Analysis System to implement the system, which has integrated features including piecewise import and export, protocol analysis, known protocol libraries, and some other essential modules. Data import and export include opening MAT format files, opening binary TXT files, opening Wireshark files, saving MAT format files, and exporting PNG format files.
5.1. Development Environment. The prototype system development and implementation environment for intelligent reverse analysis of BitTorrent protocol syntax is configured as follows:

- Operating system: Windows 10 Inter(R) Core(TM) CPU 64-bit operating system.
- Memory: 16.0 GB
- Debugging environment: Microsoft Visual Studio 2019.
- Development language: C# language.
- Protocol analysis tool: Wireshark.

5.2. Data Source. The experimental data set source of this system is divided into two main parts. The first part is the real-time data frames captured using the Wireshark tool, and the data frames are classified and saved in pcap format and TXT text format.

5.3. System Experiment Interface. The main interface of the system is shown in Figure 6, and its foreground display is an Excel-like display control.

The feature mining function of the system relies on the core idea of the OFS algorithm. The system first converts the hexadecimal strings of the protocol data set into binary strings and then displays them in the main interface, and then the feature mining module calls the OFS algorithm embedded in it to read the feature strings of the protocol data set and introduces the classical association rule mining algorithm to analyze the relationship between items in the frequent item analysis results. The frequent substrings with the lowest recognition rate are removed. Finally, the more discriminative results are stored in the protocol feature library.

The protocol identification module mainly includes two parts: protocol type determination and marking protocol features. Protocol type determination mainly relies on the protocol features generated by feature mining. When matching, the system compares the selected protocol data set with the features of each protocol in the feature library, calculates the similarity based on the number of features that can be matched to each protocol, and outputs the protocol types that satisfy the threshold value.

In the protocol type determination result, select the protocol type you want to view, and you can color the selected data set with the protocol feature marker so that you can view different protocol feature distributions according to different types. For example, after marking all “1111” strings in the file and exporting to PNG, the result is shown in Figure 7.

6. Conclusion

In this paper, we analyze the security risks of network communication in today’s high-speed development of IoT technology and propose a reverse analysis method for protocol feature extraction and identification, which first extracts the feature strings from the protocol data frames and deposits them in the protocol library, mainly for the purpose of matching with the new unknown data. The main purpose of this method is to compare and match with the new unknown data frames to identify the true identity of the unknown protocol and to achieve the purpose of maintaining the security of communication between IoT devices. The method is named the OFS algorithm, which is born by improving the existing Apriori algorithm and combining it with the idea of finding feature strings in the CFI algorithm. Combining the advantages of previous algorithms, the OFS algorithm can extract the frequent items set in the protocol data set more efficiently. Experimental results show that the OFS algorithm has a good improvement in the accuracy and speed of protocol identification and greatly improves the efficiency of the algorithm based on the original CFI algorithm, which has a good effect in the field of reverse identification of protocols.

Data Availability

All the data and methods have been presented in the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research work was supported by the National Joint Funds of China (U20B2050), National Key R&D Program of China (2018YFB1201500), National Natural Science Funds of China (62072368, 61773313, and 61702411), and Key Research and Development Program of Shaanxi Province (2020GY-039, 2021ZDLGY05-09, 2017ZDXMGY-098, and 2019TD-014).

References

[1] V. G. Semin, E. R. Khakimullin, A. S. Kabanov, and A. B. Los, “Problems of information security technology the "internet of things"” in Proceedings of the 2017 International Conference “Quality Management, Transport and Information Security Information Technologies”, pp. 110–113, Saint Petersburg, Russia, September 2017.
[2] W. Iqbal, H. Abbas, M. Daneshmand, B. Rauf, and Y. A. Bangash, “An in-depth analysis of IoT security requirements, challenges, and their countermeasures via software-defined security,” IEEE Internet of Things Journal, vol. 7, no. 10, pp. 10250–10276, 2020.
[3] Y. Yang, W. Zhang, F. Dang, L. Yan, and H. Liang, “Research on computer network information security and protection strategy based on internet of things,” in Proceedings of the 2020 IEEE 3rd International Conference of Safe Production and Informatization (ICSPI), pp. 688–691, Chongqing, China, November 2020.
[4] A. M. Mohamed and Y. A. M. Hamad, “IoT security: review and future directions for protection models,” in Proceedings of the 2020 International Conference on Computing and Information Technology (ICCIT-1441), pp. 1–4, Tabuk, Saudi Arabia, September 2020.
[5] F. Ni, J. Wei, and J. Shen, “An internet of things (IoTs) based intelligent life monitoring system for vehicles,” in Proceedings of the 2018 IEEE 3rd Advanced Information Technology,
Electronic and Automation Control Conference (IAEAC), pp. 532–535, Chongqing, China, October 2018.

[6] H. Chen, “Application of internet of things technology in ship’s personal life,” in Proceedings of the 2017 International Conference on Computer Technology, Electronics and Communication (KCCTEC), pp. 1318–1321, Sanya, China, June 2017.

[7] S. Yury and E. Samoylova, “The internet of things as socio-technological institution of civil society in post-informational era,” in Proceedings of the 2017 2nd International Conference on Computer and Communication Systems (ICCCS), pp. 142–145, Krakow, Poland, July 2017.

[8] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, “Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles,” IEEE Wireless Communications, vol. 27, no. 3, pp. 24–30, 2020.

[9] O. Al-Mahmud, K. Khan, R. Roy, and F. Mashuque Alamgir, “Application of internet of things (IoT) based smart health care medical box for elderly people,” in Proceedings of the 2020 International Conference for Emerging Technology (INCET), pp. 1–6, Begaum, India, June 2020.

[10] S. S. Mishra and A. Rasool, “IoT health care monitoring and tracking: a survey,” in Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1052–1057, Tirunelveli, India, April 2019.

[11] M. A. Mahmud, K. Bates, T. Wood, A. Abdelgawad, and K. Yelamarthi, “A complete internet of things (IoT) platform for structural health monitoring (SHM),” in Proceedings of the IEEE 4th World Forum on Internet of Things (WF-IoT), pp. 275–279, Singapore, February 2018.

[12] J. Xiong, R. Ma, L. Chen et al., “A personalized privacy protection framework for mobile crowdsensing in IoT,” IEEE Transactions on Industrial Informatics, vol. 16, no. 6, pp. 4231–4241, 2020.

[13] Y. Xu, J. Liu, Y. Shen, J. Liu, X. Jiang, and T. Taleb, “Incentive jamming-based secure routing in decentralized internet of things,” IEEE Internet of Things Journal, vol. 8, no. 4, pp. 3000–3013, 2021.

[14] Y. Xu, J. Liu, Y. Shen, X. Jiang, Y. Ji, and N. Shiratori, “QoS-aware secure routing design for wireless networks with selfish jammers,” IEEE Transactions on Wireless Communications, p. 99, 2021.

[15] W. Wang, X. Zhang, L. Dong, Y. Fan, X. Diao, and T. Xu, “Network attack detection based on domain attack behavior analysis,” in Proceedings of the 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 962–965, Chengdu, China, October 2020.

[16] J. Xiong, M. Zhao, M. Z. A. Bhuiyan, L. Chen, and Y. Tian, “An AI-enabled three-party game framework for guaranteed data privacy in mobile edge crowdsensing of IoT,” IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 922–933, 2021.

[17] Mulyadi and D. Rahayu, “Indonesia national cybersecurity review: before and after establishment national cyber and crypto agency (BSNN),” in Proceedings of the 2018 6th International Conference on Cyber and IT Service Management (CITSM), pp. 1–6, Parapat, Indonesia, August 2018.

[18] M. R. Egas, G. Ninahuapalpa, D. Molina, M. Ron, G. Ninahuapalpa, and J. Diaz, “National cybersecurity strategy for developing countries: case study: Ecuador proposal,” in Proceedings of the 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), pp. 1–7, Seville, Spain, June 2020.

[19] J. Xiong, J. Ren, L. Chen et al., “Enhancing privacy and availability for data clustering in intelligent electric service of IoT,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1530–1540, 2019.

[20] Y. Hu, L. Pang, Q. Pei, and X. A. Wang, “Analyze network protocol’s hidden behavior,” in Proceedings of the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), pp. 403–406, Krakow, Poland, November 2015.

[21] L. Gergo, “Message format and field semantics inference for binary protocols using recorded network traffic,” in Proceedings of the 2018 26th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–6, Split, Croatia, September 2018.

[22] B. D. Sija, Y. Goo, K. Shim, S. Kim, M. Choi, and M. Kim, “Survey on network protocol reverse engineering approaches, methods and tools,” in Proceedings of the 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 271–274, Seoul, South Korea, September 2017.

[23] T. Gu, A. Abhishek, H. Fu, H. Zhang, D. Basu, and P. Mohapatra, “Towards learning-automation IoT attack detection through reinforcement learning,” in Proceedings of the 2020 IEEE 21st International Symposium on “A World of Wireless, Mobile and Multimedia Networks” (WoWMoM), pp. 88–97, Cork, Ireland, September 2020.

[24] Iana: http://www.iana.org/assignments/port-numbers.?

[25] H. Seo and D. Cho, “A new alignment free genome comparison algorithm based on statistically estimated feature frequency profile,” in Proceedings of the 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 4265–4268, Jeju Island, Republic of Korea, July 2017.

[26] R. Singh, D. Rai, R. Prasad, and R. Singh, “Similarity detection in biological sequences using parameterized matching and Q-gram,” in Proceedings of the 2018 Recent Advances on Engineering, Technology and Computational Sciences (RAETCS), pp. 1–6, Allahabad, India, February 2018.

[27] M. H. Neamatollahi and M. Naghibzadeh, “Simple and efficient pattern matching algorithms for biological sequences,” IEEE Access, vol. 8, pp. 23388–23346, 2020.

[28] Y. Tian, Z. Wang, J. Xiong, and J. Ma, “A blockchain-based secure key management scheme with trustworthiness in DWSNs,” IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp. 6193–6202, 2020.

[29] A. Wichmann and S. Schupp, “Matching machine-code functions in executables within one product line via bioinformatic sequence alignment,” in Proceedings of the IEEE 5th Workshop on Mining Unstructurated Data (MUD), pp. 1–5, Bremen, Germany, May 2015.

[30] L. Zhang, Research on Feature Extraction and Identification Method of Bitstream Protocol, Xi’an University of Technology, Xi’an, China, 2019.

[31] X. Hei, B. Bai, Y. Wang, L. Zhang, L. Zhu, and W. Ji, “Feature extraction optimization for bitstream communication protocol format reverse analysis,” in Proceedings of the 2019 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/13th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), pp. 662–669,Rotorua, New Zealand, August 2019.

[32] W. Wang, B. Bai, Y. Wang, X. Hei, and L. Zhang, “Bitstream protocol classification mechanism based on feature extraction,” in Proceedings of the 2019 International Conference on
[33] D. U. Youxiang, W. Li-fa, H. Zheng, and P. Fan, "A semi-automatic protocol reverse method based on message sequence analysis," *Computer Engineering*, vol. 38, no. 19, pp. 277–280, 2012.

[34] M. Xiao-Li and Z. Xiao-Lei, "The application of data mining technology in computer network security," in *Proceedings of the 2015 7th International Conference on Measuring Technology and Mechatronics Automation*, pp. 1126–1129, Nanjing, China, June 2015.

[35] Y.-H. Goo, K.-S. Shim, U.-J. Baek, J.-T. Park, M.-G. Shin, and M.-S. Kim, "An automatic protocol reverse engineering approach from the viewpoint of the TCP/IP reference model," in *Proceedings of the 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 43–48, Daegu, Republic of Korea, September 2020.

[36] L. Cai, R. Shi, and D. Xu, "Communication protocol identification based on data mining and automatic reasoning," in *Proceedings of the IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 211–216, Beijing, China, March 2017.

[37] Z. Jie and L. Jianping, "Feature identification of unknown protocol," in *Proceedings of the 2016 13th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pp. 147–149, Chengdu, China, December 2016.

[38] M. Karimov, K. Tashev, and S. Rustamova, "Application of the Aho-Corasick algorithm to create a network intrusion detection system," in *Proceedings of the 2020 International Conference on Information Science and Communications Technologies (ICISCT)*, pp. 1–5, Karachi, Pakistan, February 2020.

[39] J. Sivapriya, R. Roy, M. Biswas, and S. Mandal, "Comparative study of APRIORI and FP algorithm for decision making," in *Proceedings of the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1058–1061, Tirunelveli, India, April 2019.