A Customized Protocol Cluster Analysis Method Based on Reinforcement Learning

Peiying Wu, Xiaohui Li and Junfeng Wang

August 30, 2022

A Customized Protocol Cluster Analysis Method based on Reinforcement Learning

Peiying Wu
College of Computer Science
Sichuan University, 610065
Chengdu, China

Xiaohui Li*
School of Cyber Science and Engineering
Sichuan University, 610065
Chengdu, China
lixiaohui@scu.ecu.cn

Junfeng Wang
College of Computer Science
Sichuan University, 610065
Chengdu, China
wangjf@scu.edu.cn

***Abstract*—The network protocol defines the rules of communication between two or more hosts on the Internet. As an indispensable means of information transmission in the network, its structure and definition are directly related to communication security. In the formulation of network security policies, the reverse analysis of unknown protocols plays an irreplaceable role, which is extremely important to the research and evolution of network security. It has a wide range of applications in malicious code detection, deep packet detection, and efficient fuzzing. In this paper, a series of researches, implementations and comparisons are made on the related technologies of reverse analysis of customized protocols. The static analysis technology route based on network tracking is selected, and it design and implement the algorithm model RLPA, which uses reinforcement learning DQN to cluster customized protocols. At the same time, the representative common clustering methods are implemented for experimental comparison. The analysis results indicate that the unsupervised model is not effective for the resolution and classification of customized protocols, while the reinforcement learning and supervised models have superior performance. Among them, the reinforcement learning model It can maintain a stable classification effect in the face of different unfamiliar datasets.**

***Index Terms*—customized protocol parsing, protocol reverse engineering, clustering, reinforcement learning**

I. Introduction

Network security is an important strategic issue for any country. It occupies an important position in national security issues and also relates to all aspects of people's lives. As an indispensable means of information transmission in the network, the structure and definition of the protocol are directly related to the communication security. However, only a very small fraction of the vast number of network protocols active on the Internet is documented, with 53% of network traffic being encrypted in 2016, growing to 87% in 2019 [1]. While some applications use encrypted private protocols with the original intention of protecting security and privacy, encryption does not represent absolute security, and a large proportion of these protocols support unauthorized communication, such as malware hidden in encrypted traffic that constituted the 2017 The main body of malicious attacks [2], unknown private protocol is the key factor. The data of online communication is rapidly becoming opaque in an all-round way, and the analysis of unknown protocols plays an indispensable role in malicious code analysis, intrusion detection, and decision-making for network security issues. The high demand for incident response and threat intrusion detection has promoted the development of unknown protocol reverse analysis, that is, protocol reverse engineering (PRE).

Protocol reverse engineering is the process in which protocol parameters, formats, and semantics are inferred in the absence of formal specifcations[3]. The reverse analysis of unknown protocols is of great significance to the research and development of network security, and has a wide range of applications, for network security engineers, software developers and other relevant roles, it is very important to master the basic format and relevant knowledge of network transmission protocols[4]. Malware protocol analysis, as one of the most typical problems in the field of network security, relies on protocol reverse analysis for implementation. Most malware uses network protocols to communicate with its command and control servers. For instance, the general botnet intrusion and control of distributed hosts are realized by using unique and customized application layer network protocols. Reverse analysis of such protocols can help identify some key information, including the location of botnet hosts, upcoming attacks, attack targets, etc., to help predict attacks and make corresponding countermeasures[5]. Another application direction of protocol reverse engineering is software security audit[6], whose main goal is to request components in different situations to check if they can handle communication correctly. Protocol models can be used to develop intelligent fuzzers to test the robustness of protocol implementations. Meanwhile, Protocol reverse engineering can detect the normalization and consistency of the implementation of known network protocols by application software, and test the performance. Reverse engineering can get the protocol model from the software development source code and the underlying configuration, so as to check whether the implementation of the protocol conforms to the specification; in addition, reverse protocol engineering can also be used to support interoperability technology implementation, a typical application is the Samba project, It enables the interoperability of Linux and Windows systems through the open-source implementation of the SMB/CIFS protocol[6].

Since the protocol specification of a customized protocol is kept secret in the hands of the developer or owner, protocol reverse engineering is often the only way to analyze the customized protocol. As the features and functions change, the protocol continues to evolve, and the completeness and applicability of the previously known protocol specifications are reduced, making it increasingly difficult to analyze customized protocols; at the same time, although the reverse engineering of automated protocols is under research and development, so far Most of the reverse parsing process is done manually, which is time-consuming and has a high error rate. Therefore, improving the accuracy, automation, and speed of protocol reverse engineering is a key issue for modern network security and improving the Internet communication environment.[7]

From the input point of view, there are two categories of current protocol reverse analysis methods: static analysis methods based on network traces and dynamic analysis methods based on execution traces[7]. The two inputs of protocol reverse engineering are execution tracking and network tracking. Execution trace is the code executed by the communicating application on multiple hosts at one time. Dynamic stain analysis is used to track and analyze communication data, to identify the semantic and structural information of the field. This method is based on the use of memory data reflected by program instructions in the protocol processing process to parse the protocol format; network tracking is the real traffic captured from the network using packet capture tools, combined with natural language processing, statistics and other algorithms to analyze the data in the data stream. Semantic syntax, field format and other information and the frequency of occurrence of keywords are counted and processed to achieve the purpose of identifying the protocol type to which different traffic belongs.

Based on the above background, our method selects the static analysis method based on network tracking, researches the clustering technology in the reverse analysis of customized protocols, and conducts experiments and comparisons to achieve a variety of network traffic text clustering methods. For the clustering algorithm, this paper implements two classical training models, unsupervised and supervised, and innovatively proposes to use reinforcement learning for text clustering. The experimental design is to change the collection method and source of the data set many times and improve the preprocessing process through code implementation, select the most effective data set, adjust and optimize the model structure and parameters, and use the same data. Set is the comparison of clustering analysis using multiple models as input, and observe the effect.

II. Related work

In this section, we describe the background knowledge of describing the work of Protocol Reverse Engineering, and briefly present the mainstream technical methods and related core concepts in this field. Then, introduces the two main categories of methods, dynamic analysis and static analysis, in protocol reverse engineering.

A. Protocol Reverse Engineering

Protocol Reverse Engineering is the process of inferring protocol parameters, formats, and statements without a formal specification. PRE performs segmentation of protocol format and recovery of Protocol Finite State Machine by analyzing traffic or program trace data[5]. The protocol finite state machine defines the time, sequence and state of the transition of the message between the two hosts, which is the state and sequence of the transition from one form to another in terms of syntax and semantics according to the protocol; the protocol field is represented as a priority field from left to right or reverse positioning.

The core steps of PRE analysis lie in format extraction, field definition and reasoning. From the perspective of control, the analysis methods of PRE can be divided into two categories: active analysis and passive analysis. Active analysis will use specific parameters to manipulate the system and conduct inspection and observation during the control process, while passive analysis will obtain various data for observation and analysis; from the perspective of input, it can be further divided into inference based on Execution Traces and inference based on Network Traces. The two inputs of protocol reverse engineering are execution tracking and network tracking. Execution trace is the code executed by the communicating application on multiple hosts at one time; a network trace is to use tools to capture network traffic and store it as an analyzable file.

PRE receives the input of network tracking or execution tracking, analyzes it in combination with the standard format of known protocols and other information, and outputs target results through a series of processing methods such as data processing, feature extraction, algorithm modeling, and semantic reasoning, and deploys them to demand scenarios. Simultaneous analysis of the results can reveal more questions and information[7].

B. Dynamic Analysis Based on Execution Tracings

Execution tracing is a piece of code or a tracing tool that is executed during an operation process of an application between multiple communication hosts. Dynamic analysis based on execution tracing captures the internal process information of binary executable files and utilizes protocol processing. It uses dynamic stain analysis technology to analyze data usage and analysis process, and identifies the semantic and structural information of the fields. This method analyzes the relevant data of the programs and instructions called by the system in the process of processing the protocol, including the use frequency of each instruction, network configuration parameters, system call parameters, operation content logs, and the polluted word of the protocol message section, instruction address, instruction dependency chain and pollution propagation relationship and other characteristics[5].

The Analytical methods based on execution tracing extract static or run-time features that are closely related to protocol handlers, and by using program analysis techniques, it is often possible to reverse-recover clean protocol processing.

However, platform-specific instructions make code or tools based on execution traces difficult to port, and this method requires access to the executable file of the protocol program. Since the executable file of the protocol program is almost inaccessible, the implementation complexity and Considering the difficulty, this paper does not adopt the dynamic analysis method based on execution tracking.

C. Static Analysis Based on Network Tracings

As the rules and conventions of Internet data transmission, the structural characteristics of protocols are relatively stable and obvious. Different protocols and their properties (including syntax, semantics, time, etc.) reflect different characteristics, so the field structure of the protocol can be used to extract a very effective analysis and identification basis. The static analysis based on network traces is simple to operate, has strong timeliness, and runs fast, and can quickly obtain results when there are many message samples and protocol types. The analysis method is to mine majority of network communication traffic data and make full use of the sequence features in them, mainly including field information such as bits, bytes, and message byte frequencies. By extracting the similar features of the data text format, identify the field boundaries, protocol syntax and semantic information[7].

This method does not need to directly access the protocol processing application program, so it is fit to intercept the traffic data between two communication hosts for processing and analysis. In general, a specific protocol field carries a specific semantics, and its information content is different, resulting in different data distribution. The prominent feature of this analysis technology is to extract the feature differences of protocol fields, so as to achieve the purpose of distinguishing fields with different structure and semantics[8]. At the same time, the stripped protocol binary executable file exposes fewer semantic features than the source data, Although dynamically executed binary files can more accurately reflect the action position and execution path of the protocol, such files are highly dependent on the environment and platform, and there are also some other constraints, which can possibly lead binary files to not execute correctly. Moreover, for the analyst, neither the client nor the server can be accessed, so it is impossible to obtain the details of the program execution process, which makes traffic data the only externally available information, and the traffic is the most important observation in static analysis technology of network tracking feature.

Based on the above background premise, the technical route of static analysis based on network tracking, which takes network traffic as the analysis object, is established for our goal. The purpose of identifying different customized protocol types.

III. RLPA Customized Protocol Analysis Algorithm

For the proposed method, the process is first described the complete process of our method, then briefly introduces the relevant basic concepts of reinforcement learning. Finally we introduces the main innovative part of this method in detail: the reinforcement learning-based network traffic text clustering algorithm model (Reinforcement Learning Protocol Analysis, RLPA) design details.

A. Framework of Algorithmic Flow

At present, the common applications with huge customer traffic on the Internet basically encrypt and encapsulate their application layer data, and use private customized protocols defined by enterprises or developers for transmission. Therefore, our method selects applications or websites which have large real-time transmission traffic for web traffic capture, and cuts the publicly known protocol parts such as MAC address, IP address, TCP/UDP protocol header, etc. in the captured network traffic text, and the remaining part is input as one-dimensional text data after preprocessing. The text clustering model extracts and divides the field format and structural characteristics of traffic, so as to achieve the purpose of directly classifying the source of unknown traffic. The schematic flow of the entire algorithm framework is shown in Figure 1:

The specific process described in the above figure is: first, collect network traffic text data based on a common private protocol, and perform data preprocessing to convert it into one-dimensional text data that can be used by the model. Input the text data into the established algorithm framework, the environment continuously throws text data to the model for classification, and gives reward feedback, the model continuously interacts with the environment and learns experience until the accumulated reward reaches the set threshold.

B. Traffic Text Classification based on Reinforcement Learning

Reinforcement learning is a training model that rewards correct behaviors and punishes wrong behaviors. The agent under training makes the choice of the next decision by sensing the feedback information of the environment, and learns the experience through multiple attempts and environment interaction. The essence of reinforcement learning is the process of finding the decision that can get the best result, and it is rarely applied to the field of classification problems before, but our method uses it to make the first attempt to solve the problem of text clustering with unknown protocol traffic.

a) Environment interaction logic: Reinforcement learning is to learn strategies through the state and rewards of environmental feedback in the interaction with the environment[9], and apply it to the scene of text clustering, just like a game environment that constantly generates text, for the agemt the ultimate objective is to learn to identify a Which app the traffic text came from. At the beginning, it is random guessing, the environment will give a feedback signal, the correct answer is rewarded, the wrong answer is punished, and the agent improves its score through continuous interaction. Finally, when the score is higher than a threshold, it is considered that the agent has learned the ”game”, that is, it has learned to classify the traffic text.

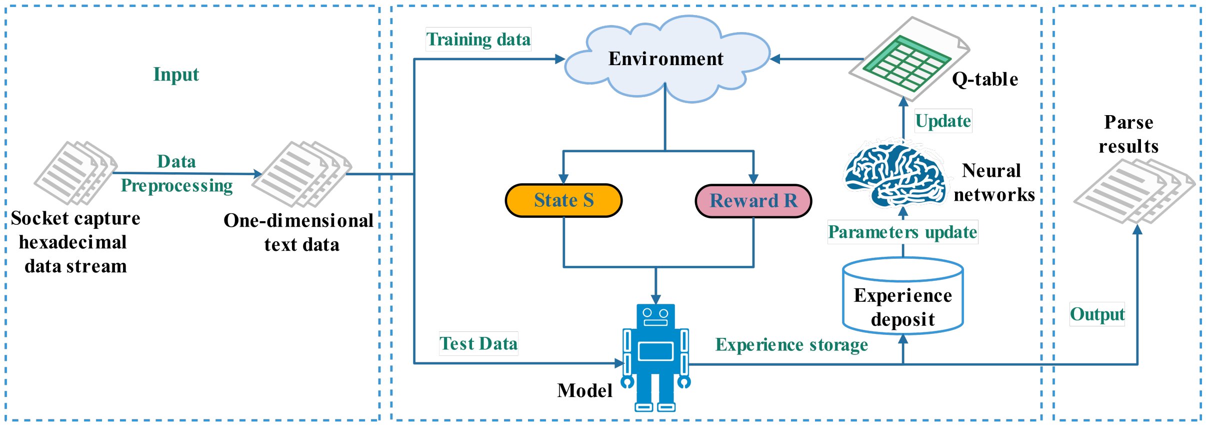


Fig. 1. Flowchart of the algorithm framework

Our method is designed based on this idea. The overall process is: the environment randomly returns a piece of text data; the agent makes an action to the environment, that is, guesses 0-4 of the application category number to which the source of the text belongs; the environment returns the reward and the next random traffic text data. The reward and punishment mechanism is to add points for correct prediction, minus points for wrong prediction, and return 0 points for initialization actions.

b) Calculation of Q value: One of the reinforcement learning methods, Q-Learning, is a value-based algorithm that inputs the current state s and outputs Q(s, a), which represents the expectation of the benefit that can be obtained by taking action a in state s. A Markov decision process, Q-Learning starts from the current state and, at any and all successive steps, finds an optimal policy based on the principle of maximizing the expected value of the total return.

The Bellman equation is the core equation in reinforcement learning algorithms and builds the recursive relationship between the value functions before and after state transitions.

$$\begin{aligned} Q\left(S_0, A_0\right) = Q\left(S_0, A_0\right) + \alpha \left[R\left(S_0, A_0\right) + \right. \\ \left. \gamma max\left\{Q\left(S_i, A_0\right), Q\left(S_i, A_1\right), \ldots\right\} - Q\left(S_0, A_0\right)\right) \end{aligned} \quad (1)$$

It is a dynamic programming equation. The left side of the equation is the new Q(S_0, A_0), and the right side of the equation is the new Q(S_0, A_0); S0 is the current state, A_0 is the current action, A_i is the next action selected, and Si is the state after A_i is selected; α is learning rate, γ is the degree of attenuation, indicating the degree of dependence on the future.

Traditional Q-learning stores Q values in table form, which is essentially an exhaustive idea. As the complexity of the problem and the amount of data increase, there are too many types of states that may exist, resulting in bottlenecks caused by dimensional disasters. Then Deep Q-Network was proposed to solve this problem[10]. DQN is an advanced Q-Learning algorithm based on neural network, which is integrated into the neural network to generate the corresponding Q value, instead of storing the Q value in the traditional table format.

In our method, if the Q-table is constructed directly using text data as a state, only a single character can be used as a feature, and the structural and grammatical features of the entire text cannot be learned. Therefore, the DQN algorithm is selected, which first uses CNN to encode the text data, and then uses the fully connected layer to output the corresponding Q value.

The steps of estimating and outputting the Q value by the neural network are: convert the input one-dimensional text data into a feature vector after processing by CNN, then input it to the fully connected layer and output the corresponding source application classification number, and input the current state to take action, and the dot product of the two is obtained to obtain the Q value estimate corresponding to the current pair of state-action pair.

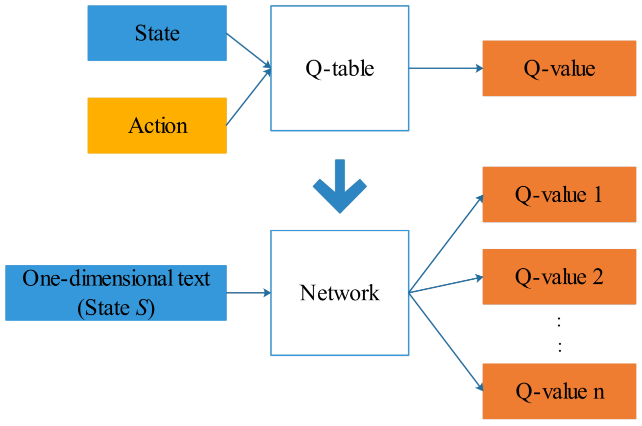


Fig. 2. Fitting Q-Functions Using Neural Networks

c) Dual network structure: In basic Q-Learning, the agent's optimal policy always chooses the best behavior in a given state. The background assumption of this move is that the optimal action has the largest estimated Q value, but the actual situation is that the agent initially knowing nothing about the environment, the Q value needs to be estimated and updated, and it is impossible to determine whether the action with the largest estimated Q value is the best action. In fact, the Q value of the best behavior is not the maximum Q value in most cases, and the optimal policy method adopted by the agent can make the Q value move closer to the target quickly, but it will lead to overestimation of the Q value, resulting in suboptimal strategic question.

The Google DeepMind team introduced a Double-DQN architecture for this Q-Learning optimistic estimation problem[11]. DDQN has two Q networks with the same structure, one is the main for action decision-making, and the other is the target for calculating the Q value. It does not directly select the one with the largest Q value among all actions, instead, it first finds it in the current Q network. The action corresponding to the maximum Q value is obtained, and then the action is used to calculate the target Q value in the target network. Its calculation formula is as follows:

$$y_j = R_j + \gamma Q'\left(\Phi\left(S_j'\right), argmaxQ\left(\Phi\left(S_j'\right), \alpha, \omega\right), \omega`\right) \quad (2)$$

Among them, y_j is the current target Q value; Q represents the current Q network of model1, Q' represents the target Q network of model2; R_j is the reward corresponding to the current state-action; $\Phi(S_j')$ is the feature vector of the state; α is the learning rate; ω is the network parameter of the current Q network; ω' is the network parameter of the target Q network.

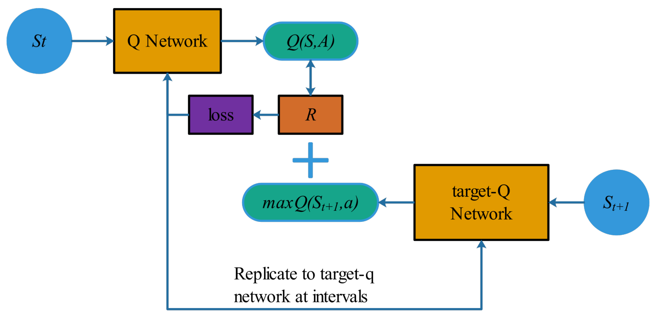


Fig. 3. Schematic diagram of dual network DQN process

The Main network, which is responsible for the action decision, always calculates the Q value according to its own weight, and uses the maximum value calculated by the target network to update the parameters of the policy function to make decisions about the action;the target network responsible for value estimation updates its own parameters by observing the Main. The two networks have exactly the same structure, so the parameters of the Main network are copied to the target network every certain number of steps.

d) Simplified experience playback steps: Experience replay is a memory replay technique in reinforcement learning, which stores the (St, At, Rt, St+1) quintuple experienced by the agent in each step in an array named memory, and then randomly samples A batch of experience is used for learning. The key reason for using experience replay is that in general reinforcement learning problems, the state St and the next state St+1 are considered to be strongly correlated, and the update of Q(S0, A0) in the Bellman equation depends on Q(Si, Ai).

Experience replay can break the correlation between consecutive samples and avoid continuous samples from causing the variance of parameter update to increase. Neural networks require training data to be as low as possible, and experience replay is used in reinforcement learning frameworks based on the context of continuous problems to ensure data independence.

The general reinforcement learning framework includes the steps of using experience playback to optimize parameters, but because our purpose is to solve a classification problem, it is a discrete problem rather than a continuous one; the data is independent and identically distributed, and each traffic data is independent of each other, so there is no certain correlation before and after. It does not need experience replay to solve the problem of strong correlation of data.

Therefore, the DQN framework of our method does not integrate the next state St+1 into the calculation of the current Q value update, simplifies the structure, and reduces the amount of calculation.Thus, the above Bellman equation can be simplified as follows:

$$Q\left(S_0, A_0\right) = (1-\alpha)\, Q\left(S_0, A_0\right) + \alpha R\left(S_0, A_i\right) \quad (3)$$

IV. Experimental Verification

Based on the network tracking-based protocol reverse method and static network traffic text clustering algorithm proposed above, this paper develops a relatively complete customized protocol parsing system including data capture and protocol clustering. In terms of clustering algorithm, the above DQN algorithm model is implemented, and four models of KMeans, Agnes, KNN, and CNN are implemented for comparison.

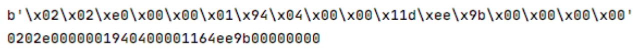
A. experimental environment

We use Python3.8 as the development language, the operating environment of the experiment is Mac OS10.13 with Pycharm Edu 2021.3.3, and the network traffic data is captured from Tencent video app, Youku video app, iQiyi app, Mango TV app, and bilibili web version

B. Data set

For the purpose of verifying the validity of the algorithm model implemented by this method, this paper intends to use the real scene private protocol data set and the Internet public data set for comparison and verification. The specific data set includes the following contents:

- Video playback private protocol data set: Each time a video app is opened and run, the socket program is used to capture packets. In order to obtain enough valid data, each app captures 20,000 pieces of data. After the invalid data is filtered by data preprocessing, each app has 5000 pieces of data for experiments.



```
b'\x02\x02\xe0\x00\x00\x01\x94\x04\x00\x00\x11d\xee\x9b\x00\x00\x00\x00'
0202e0000001940400001164ee9b00000000
```

Fig. 4. Socket capture source data and hexadecimal conversion example.

- Compare data set: In order to further accurately verify the effect of the model and avoid the over-fitting phenomenon caused by the special structure of the captured data set, our experiment uses the Android Adware and General Malware Dataset[9]] for experimental comparison. The traffic of this data set is captured on real Android devices installed with various applications, and processes them into csv format files, which contain the traffic of more than 1900 applications and are divided into three categories: adware, malware, and common software. The comparative experiment uses 4745 pieces of software traffic of each type.

The steps of data preprocessing are as follow.

TABLE I
STEPS OF DATA PREPROCESSING

Step	Content
Step 1	Convert a hexadecimal character to the corresponding decimal number, each hexadecimal character is a processing unit
Step 2	Take the average of all data lengths of the five apps, and cut all data to a uniform length
Step 3	Fill in the null part of each row of data using homogeneous mean interpolation
Step 4	Use fit_transform to fit some data, find out indicators such as mean and variance, and then transform the data, including standardization, normalization, etc.
Step 5	Perform PCA dimensionality reduction on part of the data of the five apps. The parameter is 0.98 to retain 98% of the valid information, and obtain the data length of each app after dimensionality reduction.
Step 6	Take the average value x of the length after dimensionality reduction, and perform PCA dimensionality reduction with the parameter of integer x on all data, so that all data lengths are uniformly reduced to x
Step 7	Perform fit_transform processing on the dimensionally reduced data again

C. Analysis of Results

- KMeans

For the unsupervised clustering model, the prediction accuracy cannot be directly obtained like the supervised model, and the clustering effect evaluation is performed using the evaluation index for unsupervised clustering. There should be 5 types of accurate clusters in this experiment, and the K value is 2-10 for training, and it is checked whether the K value with the best evaluation effect is 5, so as to assist in judging whether the clustering effect is accurate.

Evaluation index: The closer the Silhouette Coefficient is to 1, the better; the higher the Calinski-Harabasz score, the better. When the count of real clusters is greater than the value of the parameter K, with the increase of the value of K, the values of the two will also increase significantly; after the value of K reaches the actual number of clusters, the increase of the value of K no longer has too much effect on the degree of aggregation. Influence, the range of change is reduced, and finally stabilized.

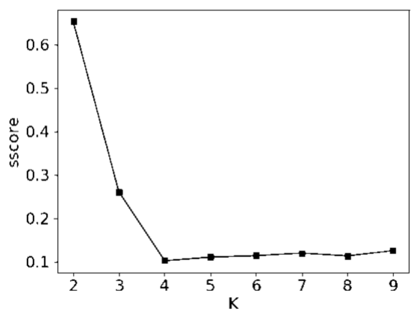


Fig. 5. Silhouette Coefficients for video app datasets.

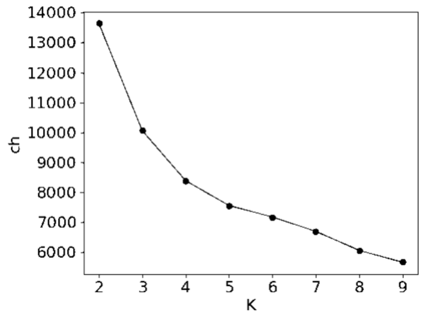


Fig. 6. Calinski-Harabasz score for video app datasets.

TABLE II
KMEANS TRAINING RESULT EVALUATION VALUE OF VIDEO APP DATASET

K Value	Silhouette Coefficients	Calinski-Harabasz score
2	0.652991	13643.059605
3	0.260383	10060.567892
4	0.102143	8392.130427
5	0.106239	7563.166399
6	0.111139	7180.266612
7	0.121585	6700.355679
8	0.114103	6067.188146
9	0.118053	5669.182936
Average	0.198329	8159.489712

It can be seen from the distribution of evaluation indicators in Figure 5, Figure 6 and Table II that the training results obtained by this model believe that 2 or 3 are the real number of species, which is inconsistent with the actual value of 5, and the performance of the silhouette coefficient and Calinski-Harabasz score when K is 5 are very bad.

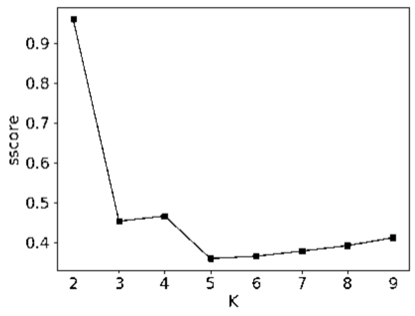


Fig. 7. Silhouette Coefficients for Android dataset.

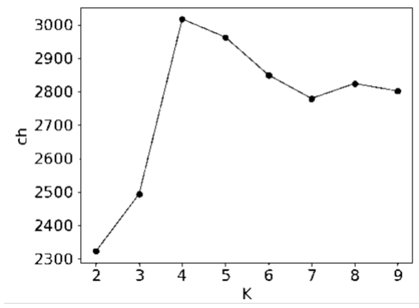


Fig. 8. Calinski-Harabasz score for Android dataset.

TABLE III
KMEANS TRAINING RESULT EVALUATION VALUE OF ANDROID DATASET

K Value	Silhouette Coefficients	Calinski-Harabasz score
2	0.959698	2323.664049
3	0.453894	2496.063979
4	0.466851	3016.911699
5	0.360408	2978.173265
6	0.366110	2849.757736
7	0.379042	2777.929478
8	0.392565	2811.220668
9	0.412769	2819.390079
Average	0.473917	2759.138869

As for the Android dataset, it can also be seen from the above results that the KMeans model thinks that the best K values in the data set are 2 and 4, which is inconsistent with the actual number of categories 3. Therefore, it is judged that the KMeans clustering model has poor clustering effect on the customized protocol data in the project.

- Agnes

The Agnes model observes the effect of the number of samples in each cluster after clustering the statistical model. The clustering results of the two data sets are as follows.

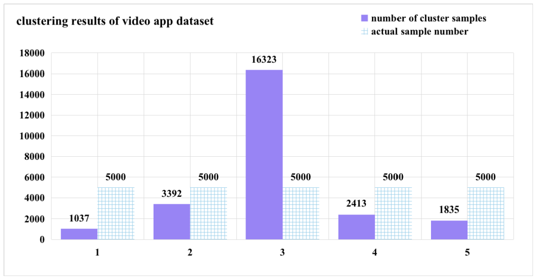


Fig. 9. clustering results of video app dataset.

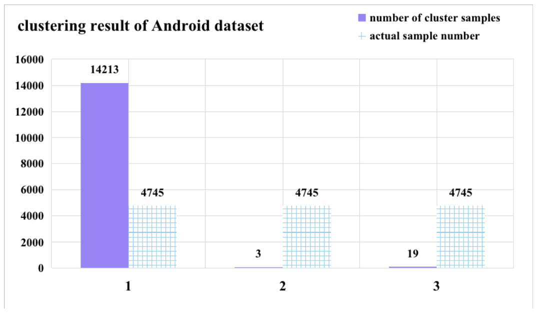


Fig. 10. clustering results of Android dataset.

The ideal clustering effect should be that the number of samples in each cluster is roughly close to 5000. The ideal result of the Android dataset should be that the number of samples in each cluster is roughly close to 4745, and the number of samples in each cluster in the two clustering results is too large. It can be seen that the Agnes model is not effective for clustering the customized protocol data in this experiment.

- Supervised Learning and Reinforcement Learning

These three algorithms can directly calculate the precision rate, accuracy rate and recall rate as evaluation indicators. The experimental results of KNN, CNN, and reinforcement learning models are shown in figure11.

For the experimental results, it is obvious that the overall effect of supervised learning is much better than that of unsupervised learning. The performance of the three algorithm models is very good when using the video traffic data captured and processed by our method, but when using the Android data set, the performance of the three all declined, indicating that there may be some overfitting in the data processing and model structure of our method.

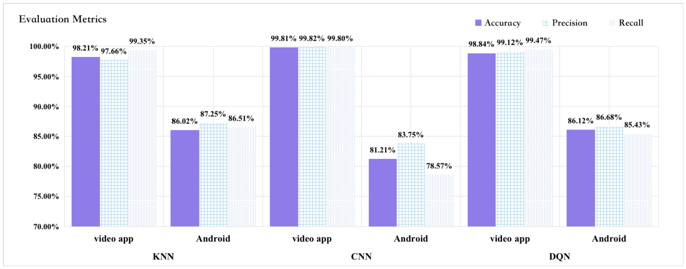


Fig. 11. Clustering results of KNN, CNN, DQN.

Under the premise of different sources and structures of customized protocol traffic data, all three can maintain an accuracy rate of more than 80%. Among them, CNN has a large change, and the three indicators of KNN and DQN are all maintained at about 85%.

V. Conclusion

During the completion of this paper, we consulted a large number of relevant papers in the field of reverse engineering of customized protocols, studied the core concepts and mainstream technical methods, and selected a static analysis method based on network tracking to implement.

At the same time, the machine learning clustering algorithm is carefully studied, and the use of reinforcement learning method for text clustering is innovatively proposed to achieve a variety of network traffic text clustering methods, and experiments are designed for research and comparison.

In terms of achievements, this project selected a static analysis method based on network tracking, used python socket programming to capture network traffic and preprocessed it into a data set; proposed a reinforcement learning clustering model based on dual-network DQN. Finally, it is verified by experiments that the supervised learning clustering model and the reinforcement learning model have good effects in the application of customized traffic clustering analysis.

Since time is limited, the improvement of the model structure and the experimental design are not very perfect, and some related parameters have not been compared with all the values. Therefore, the validity comparison of the model and the performance and effect improvement work need to be further improved.

The application of reinforcement learning in the field of clustering is very rare. The characteristics of the model itself are not suitable for the clustering problem. DQN is basically the same as KNN and CNN models in terms of accuracy, but has some disadvantages in terms of timeliness and algorithm complexity. There is still a lot of room for exploration in the application of reinforcement learning in the field of classification, and there is still a lot of room for improvement and improvement.

References

[1] Mary Meeker,Internet Trends 2019, https://www.bondcap.com/report/itr19/(accessed March 23 2021).

[2] Cisco,Cisco 2018 Annual Cybersecurity Report, 2018, https://www/csico.com/c/dam/m/digital/elq-cmcglobal/witb/arc2018/acr/2018final.pdf(accessed 23 March 2021).

[3] J. Narayan, S.K. Shukla, T.C. Clancy, A survey of automatic protocol reverseengineering tools, ACM Comput. Surv. 48 (2015) http://dx.doi.org/10.1145/2840724.

[4] Jack Halon, Reverse Engineering Network Protocols, 2022, https://jhalon.github.io/reverse-engineering-protocols/.

[5] Baraka D. Sija, Young-Hoon Goo, Kyu-Seok Shim, Huru Hasanova, Myung-Sup Kim.A Survey of Automatic Protocol Reverse Engineering Approaches, Methods, and Tools on the Inputs and Outputs View. Hindawi Security and Communication Networks Volume 2018, Article ID 8370341, 17 pages, https://doi.org/10.1155/2018/8370341

[6] Julien Duchene, Colas Le Guernic, Eric Alata, Vincent Nicomette, Mohamed KaÃ¢niche. State of the art of network protocol reverse engineering tools. Journal of Computer Virology and Hacking Techniques, Springer, 2018, 14 (1), pp.53-68. ff10.1007/s11416-016-0289-8ff. ffhal-01496958f.

[7] Yuyao Huang, Hui Shu, Fei Kang, Yan Guang. Computer Communications 182(2022)238-254. www.elsevier.com/locate/comcom.

[8] Pan Fan, et al., Overviews on protocol reverse engineering, Appl. Res. Comput.28 (Aug) (2011) 2801–2806.

[9] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, Anil Anthony Bharath, A Brief Survey of Deep Reinforcement Learning, EEE Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding (arXiv extended version),Submitted on 19 Aug 2017 (v1), last revised 28 Sep 2017 (v2).

[10] MoFan, https://mofanpy.com/tutorials/machine-learning/ML-intro/DQN/

[11] Arash Habibi Lashkari, Andi Fitriah A. Kadir, Hugo Gonzalez, Kenneth Fon Mbah and Ali A. Ghorbani, “Towards a Network-Based Framework for Android Malware Detection and Characterization”, In the proceeding of the 15th International Conference on Privacy, Security and Trust, PST, Calgary, Canada, 2017.

[12] Hado van Hasselt, Arthur Guez,David Silver, Deep Reinforcement Learning with Double Q-Learning.Vol.30 No.1(2016):Thirtieth AAAI Conference on Artificial Intelligence.

[13] O. Esoul, N. Walkinshaw, Using segment-based alignment to extract packet structures from network traces, in: 2017 IEEE International Conference on Software Quality, Reliability and Security (QRS), Prague, Czech Republic, Jul.2017, pp. 398–409. http://dx.doi.org/10.1109/QRS.2017.49.

[14] F. Sun, S. Wang, C. Zhang, H. Zhang, Clustering of unknown protocol messages based on format comparison, Comput. Netw. 179 (2020) 107296, http://dx.doi.org/10.1016/j.comnet.2020.107296.

[15] X. Luo, D. Chen, Y. Wang, P. Xie, A type-aware approach to message clustering for protocol reverse engineering, Sensors 19 (3) (2019) 716, http://dx.doi.org/10.3390/s19030716.

[16] Min Liu, Chunfu Jia, Lu Liu, Zhi Wang, Extracting sent message formats fromexecutables using backward slicing, in: 2013 4th International Conference on Emerging Intelligent Data and Web Technologies (EIDWT), pp. 377–384.

[17] K. Anastasis, M. Michail, ICSREF: A Framework for Automated Reverse Engineering of Industrial Control Systems Binaries, 2019, http://dx.doi.org/10.14722/ndss.2019.23271.

[18] G. Balakrishnan, T. Reps, Analyzing memory accesses in x86 executables, in: International Conference on Compiler Construction, CC 2004. Lecture Notes in Computer Science, vol. 2985. Springer, Berlin, Heidelberg. http://dx.doi.org/10.1007/978-3-540-24723-4-2.

[19] G. Bossert, Exploiting Semantic for the Automatic Reverse Engineering of Communication Protocols, Supelec, NNT: ´2014SUPL0027, 2014.

[20] S. Kleber, Survey of protocol reverse engineering algorithms : Decomposition of tools for static traffic analysis, IEEE Commun. Surv. Tutor. PP (2018) 1, http://dx.doi.org/10.1109/COMST.2018.2867544.

[21] D. R. Fletcher Jr., Identifying Vulnerable Network Protocols with PowerShell, SANS Institute Reading Room site, 2017.