Research on Android Malware Detection Based on Bayesian Network

Jiali Zhang[1, a], Chengxun Chen[2, b]

[1]Department of Computer Science and Engineering, Guangzhou College of Technology and Business, Guangzhou 510850, China;

[2] Huizhou TCL Mobile Communication Co., Ltd, Huizhou 516003, China.

[a]garyzhang1005@qq.com, [b]675468964@qq.com

Keywords: Bayesian network; Android malwares; app security detection; app permission; forward analysis; backward reasoning.

Abstract: Objective As the Android apps spring up at present, combined with the open source nature of Android system, malicious codes are easily embedded in Android apps, leading to a serious threat to users. However, most detection methods based on app permission features have neglected the correlation among permissions, resulting in a poor practicability and a high false alarm rate. Therefore, a malware detection method based on Bayesian network was put forward in this paper. Method The permission data of a range of Android apps were analyzed to determine the Bayesian network structure and parameter distribution on basis of expert knowledge. The open source Android app data set was introduced to verify the model, and multiple detection algorithms integrated with multiple indicators were adopted for comparison, so as to discover the maximum possible features of the malware based on the network structure. Results According to the analysis on accuracy, precision, recall, and F1 value, the indicators of this method are higher than those of logistic regression and random forest methods. The location where the malicious code is most possible embedded could be reasoned backwards by this method. Conclusion The method is accurate and feasible to locate the permission to generate malicious behavior finally in case of known malwares, providing a basis for locating the malicious code.

1. Introduction

With the progress and rapid development of technology and network in recent years, mobile intelligent devices have been comprehensively popularized and widely used in various industries in the society. Mobile apps have changed people's lives to a great extent and have been closely bound people since its emergence. Most of them may involve the user's personal information, payment information, positioning information, etc. Therefore, the security of mobile apps has attracted broad attention. As of the third quarter of 2019, benefitting from the open source service of Android, its operating system accounted for much more than the iOS operating system in the mobile operating system of the CNCERT/CC (National Computer Network Emergency Response Technical Team/Coordination Center of China)[1]. Attributing to the openness of the Android ecosystem, users may download any desired apps from any third-party platform, and developers may upload the developed mobile apps to any third-party platform for users to download, causing malwares to be very active on the Android platform. According to the 2019 Mobile Security Status Report released by the 360 Internet Security Center in February 2020, about 1.809 million new malware samples were intercepted on the mobile terminal, and the malicious behaviors of which are designed with diverse types, including privacy theft, remote control, and rogue behaviors, malicious deductions, and fraud software[2], posing a serious threat to the security when users use mobile phones. Major mobile phone security platforms have put their emphasis on the research of security detection of mobile apps, which is also a hotspot in the field of information security.

Existing methods to detect malwares have certain effects, but malwares continue to enhance their capacities to camouflage and deceive the detection system and evade detection. The functions of

malwares must also be in necessary connection with permissions. Permissions are the best embodiment of Android applications with specific functions, which is enforced by Android in terms of security design. Therefore, an array of experts and scholars generally ignored the correlation between permissions and reduce the detection accuracy when researching app permissions, which increased the false alarm rate to a certain extent. How to establish a network model of interrelated features based on existing Android application permission feature information to achieve efficient and reliable identification of malicious applications is a problem that needs to be solved urgently in Android application detection.

In order to solve this problem, the permission feature was proposed as a node, the influence relationship as a path, and the influence degree as a path parameter in this paper to establish a Bayesian network structure and solve the problem that the structure learning algorithm occupied a large amount of memory. A priori probability and conditional probability distribution were obtained based on the statistics of historical data, a complete Bayesian network was established, and then the Bayesian network inference algorithm was introduced to detect the security status of the Android app, and the permission features of the malware were reasoned backwards, thereby providing a basis for quickly locating malicious code in the future. According to the results, the Bayesian network established with this method can effectively detect the security situation of Android apps, analyze the specific permissions that affect the security situation, and acquire a good detection efficiency.

2. Related Research

The malwares can be roughly detected with two categories of methods: signature-based detection technology, and behavior-based detection technology[3]. In view of signature-based detection technology[3], QIN Zhongyuan, et al.[4] proposed to generate signatures from API, Method, Class, and APK on basis of smali files, and extract common signatures from the knowledge base according to different categories and match them to judge the category of malwares and define the malicious code retrospectively. This method effectively solved the problem of locating malicious codes. NING Zhuo, et al.[5] discovered the data flow patterns frequently operated by malwares automatically by resorting to data type analysis technology, and enhanced the ability to detect malwares with code obfuscation in combination with the improved multi-level signature detection algorithm. The signature-based detection technology has been applied in major security detection platforms. However, it can only detect malwares falling under the signature library, ineffective to detect malwares beyond the scope of signature library.

The behavior-based detection technology is mainly achieved through static detection and dynamic detection[6][7]. Bhatia, et al.[8] automatically and randomly triggered the non-pass function of the app via Android Monkey, and called the strace software to monitor the API called by the running app, comprehensively output the statistical data of api calling, and distinguished the malwares on basis of machine learning algorithm. This method effectively realized the ability to automatically identify malicious applications and improved the execution efficiency, but it had a low accuracy because Android Monkey cannot effectively trigger all functions of the app. Feng, et al.[9] proposed an optimization algorithm for multi-feature selection to identify the feature of the collected data and eliminate irrelevant ones through the feature selection algorithm, and finally distinguished the malwares through the integrated learning algorithm and improve the distinguishing accuracy of the system, but it takes a lot of time in resource processing.

Static detection technology is to detect and interpret the decompiled code and determine whether it is a malware, enjoying a merit of high code coverage, which can detect the malwares before the app is executed, but it may be ineffective under the circumstance of code obfuscation and encryption[10][11][12]13]. Mariconti, et al.[14] adopted Package and Family mode to abstract the API call sequence with API as the main research object of the static detection method, and distinguished the malwares in combination with the classification algorithm. However, the classification feature dimension obtained by this method was high, because the effect of dimensionality reduction via PCA was not ideal, and a lot of resources were consumed for

classification. According to McLaughlin, et al.[15], it was simpler to use CNN than the feature sequence generated by N-gram by training the improved Dalvik opcode sequence in CNN convolutional neural network, and learning feature sequences of malwares, which also acquired a higher accuracy.

3. Principle of Bayesian Network

3.1 Bayesian Network

In the 1980s, Judea Pearl first put forward the Bayesian Network (BN), a probabilistic graphical model, which was an extension to Bayesian method[16]. At present, Bayesian network is one of the most effective theoretical models in uncertain knowledge expression, result prediction, cause deduction and probability calculation. Bayesian network is designed in a directed acyclic graph (DAG) consisting of nodes and directed edges. Nodes are the representation of the features, expressed with a random variable. DAG explains the dependency relationship among features. A directed edge represents the relationship among features. Each node has a conditional probability function to represent the probability distribution of the node under the condition that the parent node occurs, called as a conditional probability table (CPT), which describes the degree of dependence among features by a quantitative way.

In view of mathematical description in Bayesian networks, $B =< G, \theta >$, where G is the network structure, indicating the dependency relationship structure among nodes, θ is the network parameters, describing the degree of dependence among features. Assuming the parent node π_j of the node X_i, there are multiple parent nodes of the node in the network, and the set of parent nodes of the node X_i in G is represented by $pa(X_i)$.The probability of the node X_i under the condition of the parent node π_j is:

$P(X_i \mid \pi_i)$

It can be deduced that the CPT of the node X_i in the parent node set $pa(X_i)$ is:

$\theta_{X_i|\pi_j} = P(X_i \mid pa(X_i))$

The joint probability distribution of the entire Bayesian network is:

$P(X_1, X_2, \ldots, X_n) = \prod_{i-1}^{n} P(X_i \mid pa(X_i))$

The above equation is denoted as Equation 1.

3.2 Learning of Bayesian network

Learning of Bayesian network mainly covers network structure learning and parameter distribution learning. Network structure learning is to analyze the training data set, identify the features, determine the nodes, find out the correlation among the features, acquire the dependency relationships among the nodes, and finally draw the network structure to lay a foundation for parameter learning. Research scholars have put forward different solutions to the current problem of structural learning, which can be summarized into two categories. (I) Expert experience: The dependency relationship between features is judged based on the experience knowledge of field experts, so as to establish a Bayesian network structure, its advantage lies in clear structure, maximized logical relationship, and consistent field business, and its disadvantage lies in that it relies too much on the experience and knowledge of field experts, which is more subjective, and features and relationships may be omitted inevitably; (II) Sample learning: If it is difficult to acquire the experience and knowledge of field experts, a large number of data sets need to be collected, trained and analyzed through network structure algorithms, so as to discover the dependency relationship of the features and construct the Bayesian network structure with the highest degree of fitting to the

training set, which is a NP-hard issue[17]. At present, the method based on statistical analysis is mainly used, and the dependency relationship among variables is usually analyzed using statistical or information theory methods to obtain the optimal network structure. In addition, a score search method is usually used, in which, the search strategy and scoring criteria are combined to construct a Bayesian network structure based on the structure space established by the nodes, it has an advantage that a highly fitting network structure to the training sample can be obtained, and its disadvantage lies in that different network structures are produced due to different network structure learning algorithms, and it is also susceptible to noise interference, reducing the generalization ability of the network structure.

3.3 Forward analysis and backward reasoning of Bayesian network

Let the root node X_i, $i \in N$, and $i < n$, the intermediate nodes are B_j, $j \in N$ and $j < m$, n and m refer to the number of root nodes and intermediate nodes, respectively, and their states are a_i =0,1,...,i-1, b_j =0,1,...,j-1, t=0,1,...,r-1, $X_i^{a_i}$, $B_i^{b_i}$, T_t respectively represent the status of leaf node, intermediate node and root node.

A Bayesian network structure is established for forward analysis through the relationship of the structure to obtain the probability of occurrence. In the network structure, from top to bottom, the probability distribution of the root node is deduced from the prior probability distribution of the root node and the conditional probability part of the intermediate node, thus obtaining the probability of occurrence of final event results.

Assuming that the root node X_i is in the state $X_i^{a_i}$, the conditional probability that the leaf node T is in the state T_t is:

$$P(T = T_t \mid X_i = X_i^{a_i}) = \frac{P(X_i = X_i^{a_i}, T = T_t)}{P(X_i = X_i^{a_i})}$$

The above equation is denoted as Equation 2.

According to backward reasoning, the posterior probability distribution of each root node is inferred from the bottom of the network structure, that is, the largest possible cause of the event is discovered after the event has occurred.

Assuming that the leaf node T is in the state T_t, the conditional probability distribution of each root node X_i in the state $X_i^{a_i}$:

$$P(X_i = X_i^{a_i} \mid T = T_t) = \frac{P(X_i = X_i^{a_i}, T = T_t)}{P(T = T_t)}$$

The above equation is denoted as Equation 3.

4. Research methods

The malicious risk of Android apps is analyzed and detected to determine the features, set nodes, discover the relationship between the characteristics, and establish a network. The specific steps are as follows:

4.1 Bayesian network was established for detection of mobile apps

There have been a range of detailed studies on the permission usage and impact relationship in Android Manifest.xml file in the previous researches on mobile app detection.500 mobile apps were collected in the mobile app sample library provided by the Laboratory Center of TCL Mobile Communication, including 280 malwares and 120 legitimate apps. The malicious behaviors of malwares are generally summarized as: downloading malicious installation packages, malicious sending and receiving text messages, maliciously making calls, malicious advertisements, rootkits, and malicious reading of call records, malicious reading of address book, malicious collection of

user voice, and maliciously stealing geographic location. Malwares may realize malicious behaviors by opening corresponding permissions. The permissions of Android apps were monitored and analyzed by detecting the data of 500 mobile apps, and the usage of malwares and legitimate apps on permissions was analyzed as shown in Figure 1.

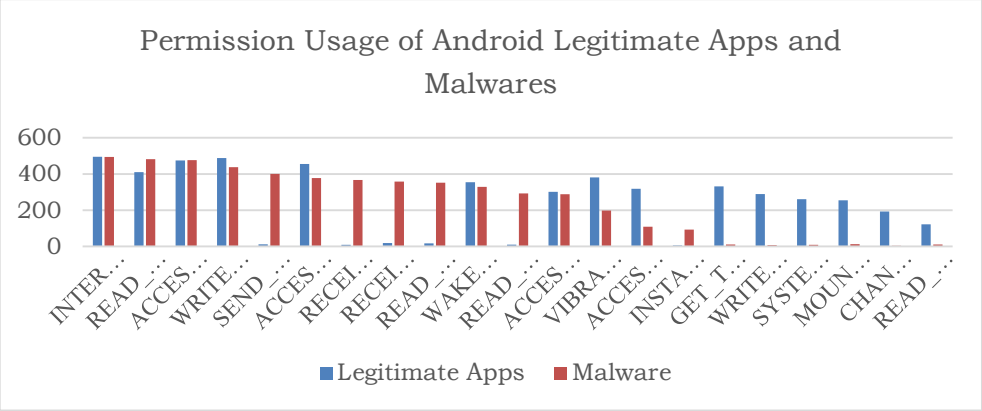

Fig. 1 Permission Usage of Android Legitimate Apps and Malwares

As shown in Figure 1, we can find that INTERNET, READ_PHONE_STATE, ACCESS_NETWORK_STATE, WRITE_EXTERNAL_STORAGE, ACCESS_WIFI_STATE, WAKE_LOCK, ACCESS_FINE_LOCATION were used by both legitimate apps and malwares, with a high utilization rate. The usage rate of GET_TASKS, WRITE_SETTINGS, SYSTEM_ALERT_WINDOW, MOUNT_UNMOUNT_FILESYSTEMS, CHANGE_WIFI_STATE, READ_LOGS were more used in legitimate apps than malwares. SEND_SMS, RECEIVE_BOOT_COMPLETED, RECEIVE_SMS, READ_SMS, READ_CONTACTS, INSTALL_PACKAGES were more used in malwares than legitimate apps. The above analysis provided a basis for identifying features and subsequent parameter determination. The impacts of the permission function can be divided into network risk impact, short message risk impact, equipment status risk impact, communication analysis impact, installation package risk impact, and location risk impact according to their effect. According to the authority function and the causal relationship that affects the effect, the Bayesian network structure was established on basis of repeated verification with expert experience and knowledge, as shown in Figure 2, where, $X_i, i \in [1,20]$ refers to each root node, $B_j, j \in [1,5]$ refers to the intermediate node, and T refers to the root node.

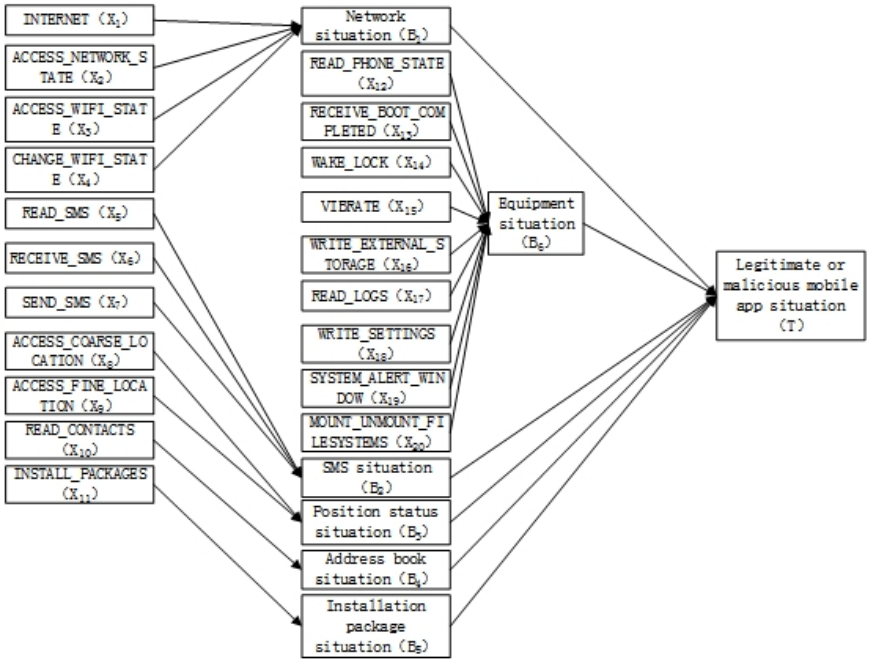

Fig. 2 App Permission Model Based on Bayesian Network

4.2 Build root node parameters

In the sampled data sets of 500 Android app, the statistical data of the status of each root node were output, that is, a_i ={enable, disable} = {1,0}, the risk status level of intermediate nodes and leaf nodes were b_j ={high risk, medium risk, low risk} = {H, M, L}. According to the statistical data in the detection, the statistical data of conditional probability of each state of each root node was output, as shown in Table 1. The cause and effect relationship of each authority was summarized to determine the conditional probability distribution of each state of the intermediate node and the leaf node. The selection B_2 and B_3 display of the intermediate node are as shown in Table 2 and Table 3, and the conditional probability distribution of leaf nodes is shown in Table 4.

Table 1 Conditional Probability Distribution of Each Root Node

Node	Status		Node	Status	
	1	0		1	0
X_1	0.989	0.011	X_{11}	0.099	0.901
X_2	0.952	0.048	X_{12}	0.892	0.108
X_3	0.833	0.167	X_{13}	0.376	0.624
X_4	0.197	0.803	X_{14}	0.684	0.316
X_5	0.369	0.631	X_{15}	0.579	0.421
X_6	0.377	0.623	X_{16}	0.926	0.074
X_7	0.413	0.587	X_{17}	0.133	0.867
X_8	0.427	0.573	X_{18}	0.296	0.704
X_9	0.59	0.41	X_{19}	0.27	0.73
X_{10}	0.303	0.697	X_{20}	0.268	0.732

Table 2 Conditional Probability Distribution of Intermediate Nodes B_2

X_5	X_6	X_7	$P(B_2 = b_j \mid X_5, X_6, X_7)$, $b_j = \{H, M, L\}$		
			$b_1 = H$	$b_2 = M$	$b_3 = L$
1	1	1	1	0	0
1	1	0	0.939	0.052	0.009
1	0	1	0.939	0.058	0.003
1	0	0	0.701	0.229	0.07
0	1	1	0.939	0.049	0.012
0	1	0	0.662	0.251	0.087
0	0	1	0.525	0.333	0.142
0	0	0	0	0	1

Table 3 Conditional Probability Distribution of Intermediate Nodes B_3

X_8	X_9	$P(B_3 = b_j \mid X_8, X_9)$, $b_j = \{H, M, L\}$		
		$b_1 = H$	$b_2 = M$	$b_3 = L$
1	1	0.489	0.321	0.19
1	0	0.478	0.349	0.173
0	1	0.364	0.428	0.208
0	0	0	0	1

Table 4 Conditional Probability Distribution of Leaf Node T (Part)

B_1	B_2	B_3	B_4	B_5	B_6	$P(T = T_t \mid B_1, \ldots, B_6)$, $T_t = \{H, M, L\}$		
						$T_1 = H$	$T_2 = M$	$T_3 = L$
H	H	H	H	H	H	1	0	0
H	H	H	H	H	M	0.853	0.147	0
H	H	H	H	H	L	0.819	0.181	0
...	...	...	...	...	...	...	...	...
H	H	H	M	M	M	0.721	0.21	0.069
H	H	H	M	M	L	0.706	0.281	0.013
...	...	...	...	...	...	...	...	...
L	L	L	L	L	H	0.078	0.692	0.23
L	L	L	L	L	M	0.022	0.29	0.688
L	L	L	L	L	L	0	0	1

5. Test results and analysis

In order to verify the feasibility and effectiveness of this method, the test was carried out on 100 Android legitimate apps and 200 malwares with data sourced from AndroZoo's public data, covering decompilation, permission setting, data extraction, data structuring, and model verification, the

results obtained by the research scholars using logistic regression, random forest algorithm and this method were introduced to establish a confusion matrix, and verify the verification index definition respectively from the 4 indicators (accuracy, precision rate, recall, F1 value), as shown in Table 5. Backward reasoning and analysis were carried out to draw conclusion that malwares were most likely to be used with specific permissions. Code detection was performed on the permission-related functions with malicious risks in turn to discover the locations where the malicious codes were most likely embedded, and corresponding protective measures were taken. Meanwhile, permissions with a high risk were included to key monitoring objects of mobile apps.

Table 5 Definition of Indicator Equation

Indicator	Indicator equation
Accuracy	$accuracy = \frac{TP + TN}{TP + FP + TN + FN}$
Precision	$P = \frac{TP}{TP + FP}, \overline{P} = \frac{1}{n}\sum_{i=1}^{n} P_i$
Recall	$R = \frac{TP}{TP + FN}, \overline{R} = \frac{1}{n}\sum_{i=1}^{n} R_i$
F1 value	$F_1 = \frac{2 \times \overline{P} \times \overline{R}}{\overline{P} + \overline{R}}$

Where, TP refers to the number of actual samples predicted as correct samples, FP refers to the number of incorrect samples predicted as correct samples, FN refers to the number of correct samples predicted as incorrect samples, TN refers to the number of incorrect samples predicted as incorrect samples.

5.1 Forward analysis and testing

In equation (2), the security level of the Android app was acquired through analysis combined with the established Bayesian network structure. The following three groups of experiments were conducted using Bayesian network, logistic regression, and random forest, respectively, and the effect was measured from the indicators of accuracy, precision, recall, and F1 value.

(1) The confusion matrixes of three sets of algorithms were established, as shown in Table 6.

Table 6 Confusion Matrixes of Three Sets of Algorithms

		Bayesian Network			Logistic regression			Random forest		
		Predicted security level								
		High	Middle	Low	High	Middle	Low	High	Middle	Low
True security level	High	173	27	0	161	29	10	171	28	1
	Middle	8	31	3	5	34	3	2	30	10
	Low	0	3	55	1	6	51	2	4	52

(2) Accuracy, precision rate, recall, F1 value

The comparison of the indicators of the three methods is shown in Table 7.

Table 7 Comparison among Various Indicators of Bayesian Network, Logistic Regression and Random Forest

Prediction algorithm	Accuracy	Precision	Recall	F1 value
Bayesian Network	0.863	0.804	0.815	0.801
Logistic regression	0.82	0.752	0.701	0.692
Random forest	0.843	0.762	0.754	0.716

As shown in the table above, the accuracy of the three algorithms has reached more than 80%, and the accuracy of the Bayesian network algorithm is better than the other two algorithms. In order to solve the sample imbalance, the indicators of this method were higher than those in other two methods for the analysis on accuracy, precision, recall, and F1 value. Therefore, the test results show that the prediction model for security monitoring of Android apps designed with Bayesian network performs better and has higher prediction capacity than the traditional prediction method.

5.2 Backward reasoning and analysis

The security degree of the known app was analyzed by resorting to the backward reasoning and analysis technology of Bayesian network, the posterior probabilities of each root node in different security states are calculated through equation 3, and the posterior probabilities were sorted in size to analyze the feature node that most likely causes an increase in security risk. Set T=H to analyze the posterior probability distribution of each root node when the security status of the Android application is at high risk, as shown in Figure 3. According to the figure, the startup probability is the highest when X_7 and X_{13} are at high risk, the probability is relatively high when X_5, X_6, X_{10} and X_{11} are at high risk, and the probability is slightly high when X_1 and X_2 are at high risk. Such permissions as SEND_SMS (X_7), RECEIVE_BOOT_COMPLETED (X_{13}), READ_SMS (X_5), RECEIVE_SMS (X_6), READ_CONTACTS (X_{10}), INSTALL_PACKAGES (X_{11}), INTERNET (X_1), ACCESS_NETWORK_STATE (X_2) are most likely to be enabled when the apps are at high risks. Taking a dating malware as a test example, the relevant technicians monitored the foregoing permissions one by one in order to find malicious code. When malicious code was detected in the code related to the SEND_SMS function, set $P(X_7^{a_1})$ and update the Bayesian network, and then make a specific analysis; If the functions related to SEND_SMS (X_7) are checked to be safe, then check X_{13}, X_5, X_6, X_{10}, X_{11}, X_1, X_2 successively to discover the most likely location of malicious code, intercept and capture the malicious code, analyze its behavioral intention, and make corresponding protection measures. In Android app risk monitoring, this method provides an important basis for discovering malicious functions of related code with specific permissions as the entrance.

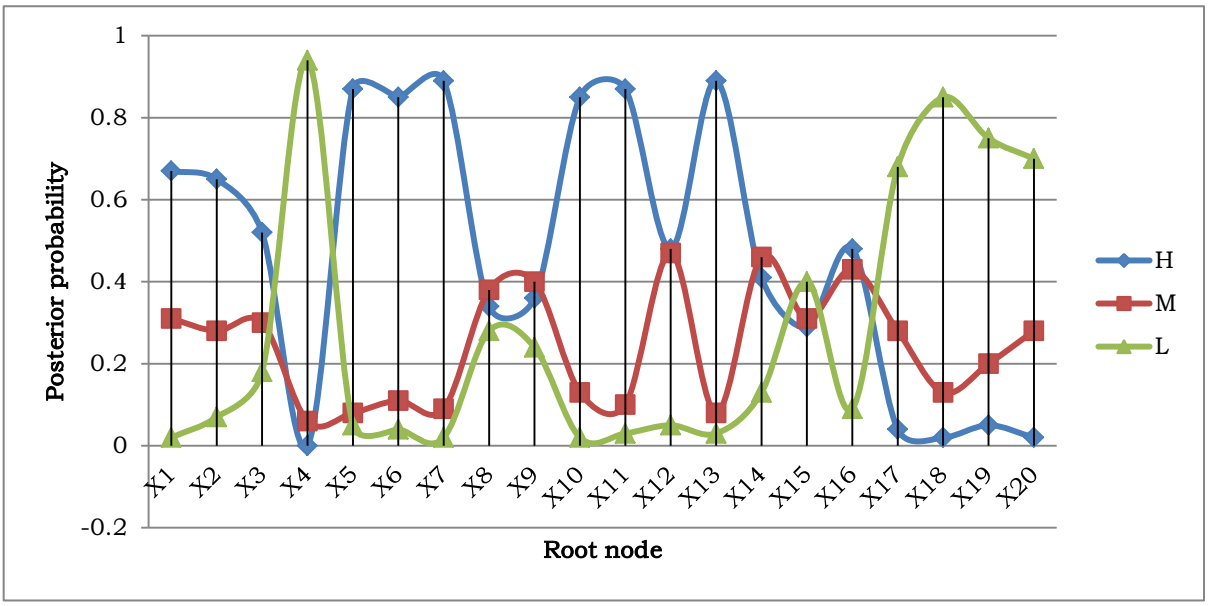

Fig. 3 Posterior Probability Curve of Each Root Node in Different Safe States When T=H

6. Conclusion

In view of the correlation among app permissions, a detection method of Android malware was put forward based on Bayesian network in this paper. By virtue of the data analysis of Android app permissions, a Bayesian network was quickly established combined with expert experience to visualize the impact of permissions, intuitively and clearly display the path of the security impact of each permission on Android apps. 100 legitimate apps and 200 malwares were determined as samples to test this method, and through testing with multiple detection algorithms, and comparison with other methods in multiple indicators, the feasibility and accuracy of the method are fully proved. By resorting to the backward reasoning technology of Bayesian network, this method can successfully locate the permissions associated with malicious behavior, providing a basis for discovering malicious code.

In the face of the complex network environment and the diversification and complexity of malwares, many uncertain factors may occur to the detection method. The follow-up work mainly falls to the following two points:

(1) Strengthening the ability of detection methods. On the basis of this research, obtain the public malware in real time via data crawling technology, update the model, and strengthen the robustness of the model.

(2) Enhancing to trace the source of malicious behavior. Further research should be carried out in terms of tracing to the source of permissions that trigger malicious behaviors, and great efforts should be made to add algorithms such as machine learning, enhance the ability to find and locate, and provide references for tracing malicious code.

References

[1] CNCERT/CC. Analysis of the proportion of domestic operating systems and browsers in the third quarter of 2019 [EB/OL].2019-12-13 https://www.cert.org.cn/publish/main/68/2019/20191213093128213770979/20191213093128213770979_.html.

[2] 360 Internet Security Center. Mobile phone security status report in 2019 [EB / OL]. 2020-02-04. https://zt.360.cn/1101061855.php?dtid=1101061451&did=610435085.

[3] XU Kaiyong, XIAO Jingxu, GUO Song, et al. Android malware detection based on improved artificial bee colony algorithm [J]. Computer Science, 2019, 46 (S2): 421-427.

[4] QIN Zhongyuan, WANG Zhiyuan, WU Fubao, et al. Android malware detection based on multi-level signature matching algorithm [J]. Application Research Of Computers, 2016,33 (03): 891-895.

[5] NING Zhuo, SHAO Dacheng, CHEN Yong, et al. Android malware detection system based on signature and data flow pattern mining [J]. Computer Science, 2017, 44 (S2): 317-321.

[6] GE Wenqi, YANG Qing, LIAO Junguo, et al. Research on feature weighted deep learning of Android malicious detection system [J / OL]. Computer Engineering: https://doi.org/10.19678/j.issn.1000-3428.0056277.

[7] LU Zhengjun, FANG Yong, LIU Liang, et al. Android malicious behavior detection method based on context information [J]. Computer Engineering, 2018, 44 (07): 150-155.

[8] Bhatia T , Kaushal R . Malware detection in android based on dynamic analysis[C]// International Conference on Cyber Security & Protection of Digital Services. IEEE, 2017.

[9] Feng P,Ma J,Sun C,et al. A Novel Dynamic Android Malware Detection System With Ensemble Learning [J]. IEEE Access,2018,6:30996-31011.

[10] LIU Qixu, WANG Baizhu, HU Enze, et al. Java backdoor detection method based on function code fragments [J]. Journal of Cyber Security, 2019, 4 (05): 33-47.

[11] Vidal J M , Monge M A S , Villalba, Luis Javier García. A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences[J]. Knowledge Based Systems, 2018:S0950705118301424.

[12] Sun M , Li X , Lui J C S , et al. Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android[J]. Information Forensics and Security, IEEE Transactions on, 2017, 12(5):1103-1112.

[13] Varma P R K,Raj K P,Raju K V S. Android moblie security by detecting and classification of malware based on permissions using machine learning algorithms. IoT in Scial,Mobile,Analytics and Cloud[C].Piscataway:IEEE,2017.

[14] Mariconti E , Onwuzurike L , Andriotis P , et al. MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models[J]. 2016.

[15] McLaughlin N,Martinez del Rincon J,Kang B J,et al.Deep android malware detection.Data and Application Security and Privacy[C]. New York:ACM,2017.

[16] PENG Yijin. Research on the evaluation of damage effects based on Bayesian network [J]. Modern Electronics Technique, 2018, 41 (11): 22-26.

[17] Siegfried M. Rump. Verified bounds for the determinant of real or complex point or interval matrices[J]. Journal of Computational and Applied Mathematics,2020,372.