SCIENCE CHINA
Information Sciences

• LETTER •

March 2020, Vol. 63 139103:1–139103:3
https://doi.org/10.1007/s11432-018-9615-8

Hybrid malware detection approach with feedback-directed machine learning

Zhetao LI[1,2], Wenlin LI[1,2], Fuyuan LIN[3,4], Yi SUN[3,4], Min YANG[5,6,7,8], Yuan ZHANG[5,8] & Zhibo WANG[9*]

[1]*Hunan Provincial Key Lab of Internet of Things and Information Security, Xiangtan University, Xiangtan* 411105*, China;*
[2]*The College of Information Engineering, Xiangtan University, Xiangtan* 411105*, China;*
[3]*Institute of Computing Technology, Chinese Academy of Sciences, Beijing* 100190*, China;*
[4]*University of Chinese Academy of Sciences, Beijing* 100049*, China;*
[5]*School of Computer Science, Fudan University, Shanghai* 200433*, China;*
[6]*Shanghai Institute of Intelligent Electronics & Systems, Shanghai* 200433*, China;*
[7]*Shanghai Institute for Advanced Communication and Data Science, Shanghai* 200433*, China;*
[8]*Shanghai Key Laboratory of Data Science, Fudan University, Shanghai* 200433*, China;*
[9]*School of Cyber Science and Engineering, Wuhan University, Wuhan* 430072*, China*

Received 20 July 2018/Accepted 25 September 2018/Published online 11 February 2020

Citation Li Z T, Li W L, Lin F Y, et al. Hybrid malware detection approach with feedback-directed machine learning. Sci China Inf Sci, 2020, 63(3): 139103, https://doi.org/10.1007/s11432-018-9615-8

Dear editor,
In recent years, the number of mobile malware has increased at an alarming rate. According to a report from G DATA [1], there are approximately 9000 new Android malware instances each day. Such malicious applications pose grave threats to the security of the Android ecosystem.

Machine learning is quite effective at automatically detecting malicious samples. According to the previous research, there are two methods to detect malware – static detection and dynamic detection. The static detection method detects the application by analyzing the source code files of the application and extracting features such as permissions, XML features, and API calls. For example, Enck et al. [2] analyzed the permission usage of applications to discover malicious behaviors. Shabtai et al. [3] used features from XML files to dissect applications, and FlowDroid [4] collected system API calls to keep track of the path from the sources to the sinks to detect malicious activities. This detection method is simple and fast but is unable to detect sophisticated malware, and its accuracy is not satisfactory.

In contrast, dynamic features that capture the actual application behavior characteristics could more accurately express the intention of the application. For example, TaintDroid [5] detected malware by tracking privacy-sensitive data streams in real time. Crowdroid [6] used the crowdsourcing system to obtain traces of applications' behaviors and then provided the classification result by dissecting the trace logs. Although this method can accurately distinguish malware, it also raises another issue that we require considerable time to observe the real-time behavior of the application. Furthermore, some existing researches combined static features and dynamic features to detect malware [7, 8]. However, they only focused on accuracy and ignored detection time consumption.

To balance accuracy and time consumption, we propose a new detection framework called Hyda that comprises two phases, static analyzer and dynamic analyzer. The static part adopts the voting mechanism and the multi-classifier model to filter the malware, and the dynamic stage makes the final decision according to the characteristics of the application runtime. In contrast to the static detection method, this approach can reach higher detection efficiency. Compared with the dynamic

* Corresponding author (email: zbwang@whu.edu.cn)

Possible intent	Patterns
Send contact	(a) Create contacts ContentResolver (b) Query contacts ContentResolver (c) Create HttpClient (d) Set HttpPost entity (e) Execute POST request to a given URL
Send SMS to subscribe paid content	(a) Get SmsManger object (b) Send subscription message to speci_ed mobile number (c) Register BroadcastReceiver with SMS IntentFilter
Lock screen to blackmail	(a) Get "device policy" SystemService (b) Reset password with SystemService (c) Lock screen

(a)

	Precision (%)	Accuracy (%)	Recall (%)	FPR (%)	Proportion (%)
Static analyzer	97.22	97.05	63.93	1.77	77.6
Dynamic analyzer	98.81	98.68	99.20	2.33	22.4
Hyda	97.96	97.60	97.06	1.90	100

(b)

	Precision (%)	Accuracy (%)	Recall (%)	FPR (%)	ADT (s)
Dynamic detection	97.09	98.30	99.09	2.31	302.5
Hyda	97.96	97.60	97.06	1.90	75.5

(c)

Figure 1 (a) Examples of API frequent patterns and corresponding intents; (b) the effectiveness of Hyda; (c) performance comparison with dynamic detection.

method and other hybrid method, Hyda reduces detection time, improves detection performance, and is more suitable for large-scale malware detection.

System design. Hyda is mainly divided into two stages: static analyzer and dynamic analyzer. The static analyzer is composed of the following modules: decompile module, static features extraction module, feature selection module, and voting module. The extracted static features include APK features, XML features, and API features. Specifically, the API features contain a feature called API frequent features. We obtain it using the data mining algorithm FP-Growth, which represents some behavior wherein a series of APIs combined together would be required to accomplish it. The frequent pattern in data mining field is always used to find the associated items and customers' potential demands. Therefore, every frequent pattern of API calls are just like a sequence chart of API calls, which could show us the true intent of a function or a class. Figure 1(a) lists some examples of API frequent patterns and their corresponding intents.

After extracting the static features, we gain the feature set of a total of 757 dimensions. However, numerous features would slow down the process of model training and cause the overfitting problem. Therefore, we sort and select the key features according to the information gain value of the features in the feature selection module. Information gain is a metric that describes how much amount of classed information can a feature bring. The calculation process of the information gain value is as follows:

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}. \tag{1}$$

Let C_k be the k-th type of software, and K is always 2 in Hyda. $|D|$ is the size of the training set, and the entropy $H(D)$ is defined by (1).

Let A be the feature to be examined. D_i is the i-th part of the original training set that is divided by the values of feature A. $H(D_i)$ is the entropy of the data set D_i, and $H(D|A)$ is the entropy of D under condition A.

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i). \tag{2}$$

Finally, the information gain value of feature A is calculated as

$$g(D, A) = H(D) - H(D|A). \tag{3}$$

In the voting module, we design a multi-classifier model and confidence mechanisms. Specifically, the multi-classifier model comprises five different classifiers in parallel. For the test sample, each classifier returns a true or false classification result. Meanwhile, the confidence mechanism is defined as follows: if classification results of all classifiers are consistent, the model returns a high confidence message and classification result. Otherwise, the model returns a low confidence and adds this application to the dynamic analyzer phase. Confidence is a measure of reliability of the static detection result. It is the intention that improves the accuracy of static detection and utilizes it as a switch to trigger the dynamic analyzer.

When the static analyzer is not so confident for classifying an application, it comes to the second phase, i.e., dynamic analyzer. The dynamic analyzer part is mainly divided into the following modules: upload module, application execution module, feature extraction module, and classification module. In the application execution module, we run the application and command line tool MonkeyTest together on the real machine. In particular, the Android system running on the real machine is a customized Android system, and all its system APIs are replaced by hook functions, which can record the application runtime characteristics and device resource usage. In detail, these dynamic features that we extracted include whether to send text messages, the number of bytes uploaded by the network, the number of bytes downloaded by the network, the time spent on the network, CPU usage, and memory usage. Finally, the classifier model classifies the application based on these features.

Experimental verification. The dataset used in the experiments is collected in various application platforms. There are 4138 applications in our dataset, including 2000 benign applications and 2138 malware. Specifically, the effectiveness of Hyda is shown in Figure 1(b). We observe that the static analyzer can filter out 77.6% of the test samples, and its accuracy can reach 97.05%, which is 5.2% higher than the existing static detection method ASCA [3]. The dynamic analyzer tested the remaining 22.4% of the samples, and the accuracy rate reached 98.68%. In summary, we observed Hyda's accuracy to be 97.60%, precision to be 97.96%, and recall rate to be 97.06%.

We compare Hyda with other existing studies, which is 5.8% higher than the static detection method ASCA [3], and only 0.2% lower than the pure motion detection method DroidMat [9]. Furthermore, the performance comparison between Hyda and dynamic methods is shown in Figure 1(c). Although Hyda's accuracy is 0.7% lower, the average detection time (ADT) of Hyda is only a quarter of the pure dynamic detection method, and its precision is 0.87% higher. The experimental results demonstrates that Hyda's efficiency is close to dynamic detection, the detection time is shorter, and it has better applicability.

Conclusion. We propose a hybrid scheme using multi-level analysis technology for Android malware detection. First, we adopt the static analyzer to filter large-scale samples and divide all samples into high-confidence and low-confidence samples. Then, we add the low-confidence samples to the dynamic analyzer. The experimental result verified the effectiveness of our proposed scheme. In general, compared with the traditional detection method, Hyda solves the contradiction between accuracy and detection time consumption and is suitable for large-scale malware detection scenarios.

Acknowledgements This work was supported in part by National Key Research and Development Program of China (Grant No. 2018YFB1003702), National Natural Science Foundation of China (Grant No. 61872274), and Hunan Provincial Natural Science Foundation of China for Distinguished Young Scholars (Grant No. 2018JJ1025).

References

1 G DATA. Threat situation for mobile devices worsens. https://www.gdatasoftware.com/news/2017/02/threat-situation-for-mobile-devices-worsens

2 Enck W, Ongtang M, McDaniel P. On lightweight mobile phone application certification. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, 2009. 235–245

3 Shabtai A, Fledel Y, Elovici Y. Automated static code analysis for classifying android applications using machine learning. In: Proceedings of International Conference on Computational Intelligence and Security, Nanning, 2010. 329–333

4 Arzt S, Rasthofer S, Fritz C, et al. FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. In: Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation, New York, 2014. 259–269

5 Enck W, Gilbert P, Chun B G, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. In: Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation (OSDI'10), Berkeley, 2010. 393–407

6 Burguera I, Zurutuza U, Nadjm-Tehrani S. Crowdroid: behavior-based malware detection system for Android. In: Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM '11), New York, 2011. 15–26

7 Islam R, Tian R, Batten L M, et al. Classification of malware based on integrated static and dynamic features. J Netw Comput Appl, 2013, 36: 646–656

8 Shi Y, You W Q, Qian K, et al. A hybrid analysis for mobile security threat detection. In: Proceedings of the 7th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON), New York, 2016

9 Wu D J, Mao C H, Wei T E, et al. DroidMat: Android malware detection through manifest and API calls tracing. In: Proceedings of the 7th Asia Joint Conference on Information Security, Tokyo, 2012. 62–69