Scalable Traffic Dependence Analysis for Detecting Android Malware Activities*

Hao Zhang, Danfeng (Daphne) Yao, and Naren Ramakrishnan
Department of Computer Science, Virginia Tech
Blacksburg, VA, USA
{haozhang, danfeng, naren}@cs.vt.edu

Keywords

Anomaly detection, Mobile security, Network security

The openness of Android application development mechanism poses security challenges to smartphone users. Malicious apps (malware) may be created by repackaging popular apps. At runtime, they directly fetch and run code on-the-fly without the user's knowledge [6]. Thereafter, malicious apps may spy on the victim users, stealthily collect and exfiltrate user's information. Therefore, they threaten the data confidentiality and system integrity on Android devices.

Existing static analysis solutions inspect the source code, binaries or call sequences for anomalies. For example, SCS-Droid [7] identifies the malicious apps by extracting the subsequences of system calls. However, dynamic code loading, Java reflection-based method invocation, data encryption, and self-verification of signatures are commonly seen in the malware code [4]. These types of code obfuscation make static analysis based detection challenging. Dynamic analysis, as a complementary to the static analysis, detects the runtime behaviors of the malicious apps (e.g., verifying the sensitive information flows through the apps [3]).

In this work, we aim at identifying malicious apps by analyzing their dynamic behaviors, specifically the network traffic. Our goal is to profile the normalcy of the mobile traffic patterns of benign apps, and thus to *detect the malicious network requests that are sent without user's awareness.*

We propose a *triggering relation* model to formalize the causal relations of app-generated network requests in Android. The problem of dependency discovery on network data was first introduced in [12]. The dependency of network requests is defined as the *triggering relationship* (TR). The triggering relation of two network requests r_i and r_j exists if r_j cannot be issued unless r_i is sent out first. This temporal and causal relationship can be represented in a directed graph that is referred to as *triggering relation graph* (TRG). We refer the first request that triggers others as the *root-trigger* request. It may be triggered by legitimate user activities or generated by benign apps. The model depicts the dependency of network events and explains how one triggers the other. Furthermore, the graph allows one to understand the causality among traffic and find anomalous events (i.e., requests). The dependencies and semantic information of requests are useful for human experts' cognition, reasoning, and decision-making in cyber security [2, 11].

In our threat model, the behaviors of malware include the unauthorized network activities, e.g., stealthy outbound requests without user's awareness, piggyback requests with malicious code, and other types of out-of-order requests. These behaviors existing in a wide range of malware families cause sensitive information leaking and system abusing. The repackaged apps and drive-by download attacks are the common initial infection vectors. After the malicious apps are installed, the requests sent to remote hosts could leak user's information or conduct bot activities for profits. Their behaviors often include the stealthy network communication to remote servers (e.g., command and control).

Our model focuses on the stealthy network activities via HTTP, because HTTP is the protocol of choice for most app developers to implement communication with remote servers and is hardly blocked by anti-virus tools. Our proposed technique can detect these types of activities without knowing any signatures of the malware, and thus applicable to detect the new (zero-day) malicious apps.

We describe a machine learning approach for discovering the triggering relationship of Android network requests. The learning-based solution enables the detection of general stealthy malware activities on a large scale. Straightforward solutions [9, 10] either reason the dependency using heuristic algorithms or limit the causality analysis on one specific application (e.g., browsers), which cannot achieve our goal in Android.

The main operations in our analysis are data labeling, pairing, TR analysis and detection. To label the triggering relations, we present a method based on the timing perturbation, which delays one request and see if others would be affected. The rationale behind this method is that the delay of an individual request will be propagated to the requests that are triggered by it. Therefore, the delay injection method is ideal for generating rules on a small-scale dataset, and is sufficient for labeling and training purposes. In contrast, our proposed learning-based approach can be used for the analysis and testing of a large-scale dataset.

*This work has been supported in part by NSF grant CAREER CNS-0953638, ARO YIP W911NF-14-1-0535, and L-3 communications.

AISec'15, October 16, 2015, Denver, Colorado, USA.
ACM 978-1-4503-3826-4/15/10.
DOI: http://dx.doi.org/10.1145/2808769.2808770.

An important technical enabler of our solution is the ability to discover the triggering relation of pairs of requests. We refer to it as the *pairwise* triggering relation discovery. The pairwise comparison method has been proposed to solve the general relation discovery problem [5, 12]. In this work, we advance it in the Android context by improving the pairing efficiency and descriptive power of tree structures. The pairing operation is performed on any two requests for whose time difference is less than a threshold τ. It leverages comparison functions and generates a set of pairwise features, e.g., the difference of two timestamps, the similarity index of two URLs, etc. Our pairing mechanism does not require any *priori* domain knowledge or assumptions to filter the potential requests within τ. In the TR analysis operation, we train classifiers to predict triggering relations on pairs. The predicted results are in the form of $(r_i, r_j) \rightarrow l_{ij}$, where r_i and r_j are paired requests, and l_{ij} describe the relationship of r_i and r_j, e.g., the parent-child or sibling relation.

The predicted pairwise relations are used to construct the *full* triggering relation graph. The TRG reveals the relationship of network packets sent from all types of apps, and thus it has two security applications: *i)* identifying the malicious requests that are not triggered by a legitimate cause, and *ii)* generating a whitelist of automatic updates and notifications. The TRG allows one to quickly identify the *root-triggers* of observed network events. Thereafter in the detection operation, our solution classifies the root-triggers to identify malicious requests. We propose to integrate the knowledge from our findings on the TRG and generate the dependency-based features. Besides, we extract temporal, lexical and host-based features that are introduced to detect malicious domains and suspicious URLs [1, 8]. Both TR analysis and detection include training and testing, as the standard operations for the machine learning approach.

To evaluate our solution, we collect 20+ GB network and system logs for a continuous 72-day period from a Nexus 7 tablet. We define *TR accuracy* as the ratio of the number of requests correctly identified its trigger to the total number of requests. This metric evaluates the effectiveness of the classifiers used in the TR analysis. We vary the sizes of training data and test the data of each new day. We use five different training sets: 1-day, 5-day, 10-day, 15-day, and 20-day, each of which means data from i day(s) prior to the test day. Figure 1 shows that all TR accuracy results are above 99.0% for training data that are larger than 5 days. C4.5 classification algorithm outperforms the other two classifiers across all training data sizes. The TR accuracy converges quickly to its highest value, which shows our approach to discovering triggering relations is scalable and only requires training on the recent traffic data. In the detection operation, we classify the root-triggers to identify the malicious requests. Results show that by using the random forest classifier, the false positive rate is 1.1% and we are able to detect 99.7% malicious requests from all malicious apps. We identify all 10 malicious apps that behave like Trojan, spyware, or bots. In comparison, only one is reported by our organization IDS.

Our contributions and experimental findings are summarized as follows.

- We propose a triggering relation model to formalize the dependency of network requests and traffic-generating user inputs on the Android context. The model reasons about the root-triggers of observed traffic, and thus can detect stealthy malware activities.

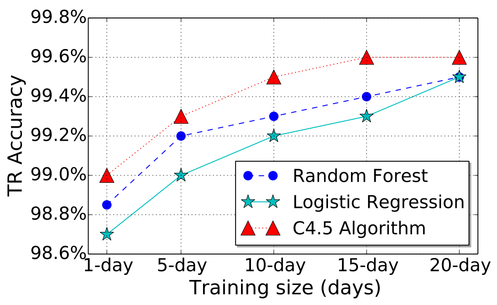

Figure 1: The TR accuracy of three machine learning classifiers tested on Android network requests.

- We introduce a delay injection technique to discover the triggering relations of Android network events. Our learning-based approach enables the inference of traffic dependencies and reasons about the root-cause of network activities, which makes our solution beyond the conventional binary classification ones.
- We conduct our experiments based on real-world network and system data. Our results show that the triggering relations of network traffic on Android can be inferred at the accuracy of 99.6%. We confirm the detection capability of our approach by pinpointing the sparse anomalies out of voluminous traffic data.

The significance of our work is that it provides insights of the Android network traffic dependency and demonstrates the use of structural and semantic information in reasoning about network behaviors and detecting stealthy anomalies.

References

[1] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *NDSS'11*.

[2] K. O. Elish, X. Shu, D. Yao, B. G. Ryder, and X. Jiang. Profiling user-trigger dependence for Android malware detection. *Computers & Security*, 49:255–273, 2015.

[3] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM TOCS*, 32(2):5, 2014.

[4] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of Android application security. In *USENIX'11*, 2011.

[5] L. Getoor and C. P. Diehl. Link mining: a survey. *SIGKDD Explor. Newsl.*, 7(2):3–12, December 2005.

[6] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *WiSec'12*, pages 101–112, 2012.

[7] Y.-D. Lin, Y.-C. Lai, C.-H. Chen, and H.-C. Tsai. Identifying Android malicious repackaged applications by thread-grained system call sequences. *computers & security*, 39:340–350, 2013.

[8] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In *KDD'09*, pages 1245–1254, 2009.

[9] C. Neasbitt, R. Perdisci, K. Li, and T. Nelms. ClickMiner: Towards forensic reconstruction of user-browser interactions from network traces. In *CCS'14*, pages 1244–1255.

[10] G. Xie, M. Iliofotou, T. Karagiannis, M. Faloutsos, and Y. Jin. ReSurf: Reconstructing web-surfing activity from network traffic. In *IFIP Networking Conference, 2013*, pages 1–9.

[11] H. Zhang, M. Sun, D. Yao, and C. North. Visualizing traffic causality for analyzing network anomalies. In *IWSPA'15*, pages 37–42, 2015.

[12] H. Zhang, D. Yao, and N. Ramakrishnan. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *ASIACCS'14*, pages 39–50.