POSTER: Semantics-Aware Rule Recommendation and Enforcement for Event Paths

Yongbo Li(✉), Fan Yao, Tian Lan, and Guru Venkataramani

George Washington University, Washington D.C. 20052, USA
{lib,albertyao,tlan,guruv}@gwu.edu

Abstract. With users' increasing awareness of security and privacy issues, Android's permission mechanism and other existing methods fall short to provide effective protection over user data. This paper presents SARRE, a Semantics-Aware Rule Recommendation and Enforcement system to detect critical information outflows and prevent information leakage. SARRE leverages runtime monitoring and statistical analysis to identify system event paths. Then, an online recommendation algorithm is developed to automatically assign and enforce a semantics-aware security rule to each event path. Our preliminary results on real-world malware samples and popular apps from Google Play show that the recommended rules by our system are effective in preventing information leakage and enabling protection policies for users' private data.

1 Motivation

With its increasing popularity among all smartphone platforms, Android continues to claim the largest share of malware [1], a lot of which collects and leaks users' private data. In addition, users' information can even leak out through apps downloaded from Google Play [7]. Information leakage and user privacy remain to be challenging problems for hardening smartphone security.

The limitations of Android's current permission-based security mechanism have been well recognized in prior work [6]. A number of proposals are made to tackle this challenging problem using techniques such as enhanced Access Control [6] and data obfuscation [8]. However, the burden of manually constructing extensive security rules for various apps still lies with smartphone users or app developers, who may find it overly convoluted and difficult to adjust on the fly. The problem is further complicated when different information flows accessing the same data require differentiated security rules. For instance, while GPS coordinates are routinely queried by information flows in map/tracking apps, it could raise serious privacy concerns if they are accessed by an alarm clock app, whether it contains repackaged malware or advertisement libraries collecting users' location. Recent studies have begun to investigate automated rule assignment in smartphone systems [5], but only consider a one-size-fits-all solution for each data source and fall short on providing fine-grained security rules for different information flows and app semantics.

B. Thuraisingham et al. (Eds.): SecureComm 2015, LNICST 164, pp. 1–5, 2015.
DOI: 10.1007/978-3-319-28865-9_33

2 Our Approach

We propose SARRE, a Semantics-Aware Rule Recommendation and Enforcement system that automatically assigns and enforces security rules for event paths to prevent information leakage. SARRE consists of four main parts: (i) *Event Monitor*, (ii) *Path Identifier*, (iii) *Rule Recommender*, and (iv) *Camouflage Engine*. The interconnections between different parts are depicted in Figure 1.

Event Monitor intercepts and logs timestamped events (within a configurable list) at Android's framework level. Monitored events include: (i) apps' calls to APIs that can be leveraged for data collection, processing, and transmission, such as API calls to access location and network services; and (ii) other system events or phone state changes that are not directly related to information flows, but facilitate characterization of them, for example, incoming phone calls and new SMS notifications frequently serve as different triggers for information flows. The log files generated by *Event Monitor* are encrypted and transmitted periodically to *Path Identifier* by *Secure Sender*.

Path Identifier quantifies the correlations between events within the log file through statistical analysis to construct an Event Graph for each app. Each vertex on the graph is a monitored event, and an arc between two events exists if and only if their correlation is statistically significant. A weight is assigned to each arc measuring the correlation significance. *Path Identifier* then leverages our path cover algorithm to extract the event paths with largest accumulated weights, covering all the events' occurrences in the log file. The Event Graph constructed for a malware sample *com.nicky.lyyws.xmall* [2], is depicted in Figure 2. Because of space limitation, the event names are shown in an abbreviated manner. The numbers at the end of vertices denote counts of event occurrences in the log file. Paths identified for this sample are also shown in the figure.

Next, *Rule Recommender* assigns fine-grained security rules to newly-identified event paths. A rule R_r is numerically denoted and $R_r \in [0, 1]$, indicating level of protection needed for the sensitive data associated with the event path. In specific, the recommender leverages two types of knowledge: (i) known security rules of similar paths and (ii) the corresponding apps' semantic information. A recommendation is made for an event path by calculating the weighted

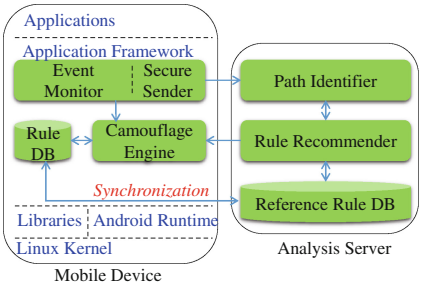

Fig. 1. Overview of System Design

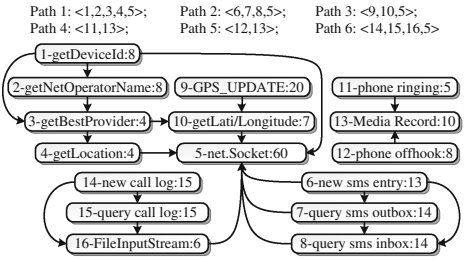

Fig. 2. Event Graph and paths of *nickispy*

average of the rules for K nearest event paths with matched semantic information. This approach allows SARRE to construct security rules that are both effective and in-context in an unsupervised manner.

Finally, recommended security rules are enforced by *Camouflage Engine* for the sensitive information flows on event paths at run time. The camouflage action is selected based on the underlying data property. For example, numerical GPS coordinates can be obfuscated by adding random noise to reduce their resolutions, while fields in structured data like contacts data can be selectively hidden depending on the recommended security rule.

3 Evaluation of Effectiveness

We prototyped SARRE on Android Open Source Project v4.1.2. We collected malware samples from an online sharing site [4], and top ranking apps on Google Play. Then, we manually select one from some pre-defined labels[1] such as *Games* and *Social* to each sample based on the app description. The label denotes the app's declared functionality, and serves as the semantic information in current evaluation. We present the effects of the recommended rules for two examples.

– *My Tracks:* Since malware normally doesn't present harvested data when stealthily eavesdropping on users' location, we use a tracking app *My Tracks* to emulate malware by intentionally replacing its actual label 'Tracking/Maps' with 'Games'. We choose a tracking app because it has a UI showing GPS coordinates update, which makes it convenient to compare the data when different rules are enforced. An event path identified for *My Tracks* involving location data is written in an abbreviated manner, as follows:

GPS updated → *getLocation* → *Socket.getOutputStream* → *Socket.connect*

The recommended rule for this path is 0.4, which means a security action corresponding to 0.4 needs to be applied when it shows up in a game app. When we replace the actual label 'Tracking/Maps' back, the rule recommended is 1, meaning such an event path in a 'Tracking/Maps' app should be left intact. The tracks with rules 1 and 0.4 enforced are shown in Figure 3. Malware like *nickispy* [2] eavesdropping users' location exhibit similar event paths, and after rule enforcement the location data sent should be similar to the right one in Figure 3.

– *Love Chat [3]:* This malware doesn't show an UI, but has an event path as follows, in which data is accessed, stored locally and sent out by network socket.

getLine1Number → *getDeviceId* → *query(contacts)* → *io.FileOutputStream* → *Socket.getOutputStream*

Based on this sample's declared functionality, we attach 'Communication' as its label. The recommended rule for this path is 0.2. With examination of the

[1] In our design, such labels are assigned by our system based on apps' functionality descriptions, and they cannot be manipulated by the apps.

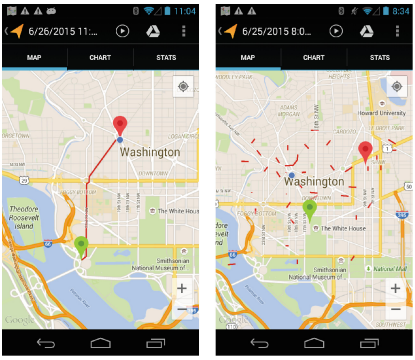

Fig. 3. Intact track (left) when rule '1' is enforced, and track with noise (right) when rule '0.4' is enforced

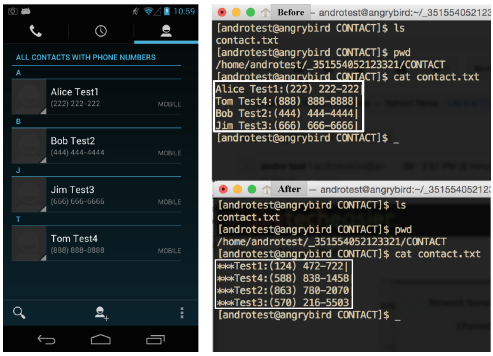

Fig. 4. Contacts on the phone (left) and data sent by *Love Chat* when rule '1' (upper right) or '0.2' (lower right) is enforced

Reference Rule DB, we see that although this malware disguises as a 'Communication' app, it doesn't get a rule with large number, because paths similar to the above path are popular among privacy-stealing malware, but not 'Communication' apps. This makes sense and shows the necessity to use event path as reference for rule recommendation. To see the enforcement effect, we redirect packets sent by this sample to an external server. We input some made-up contacts data on the phone as shown in Figure. 4 (left). The contents in the files that are sent to the external server before and after rule enforcement are also shown in the figure (right). We can see the effectiveness of the rule enforcement by hiding contacts' first names and scrabbling some digits in the phone numbers.

Acknowledgments. This work is supported, in part, by the Office of Naval Research under grant No. N00014-15-1-2210. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.

References

1. Android still triggers the most mobile malware. http://goo.gl/FXTGsi
2. Malware with package name *com.nicky.lyyws.xmall*. http://goo.gl/U7D2FW
3. Malware with package name *com.yxx.jiejie*. http://goo.gl/CpkioI
4. Mobile malware sharing website. http://goo.gl/YNIOLg
5. Chakraborty, S., Shen, C., Raghavan, K.R., Shoukry, Y., Millar, M., Srivastava, M.: ipShield: a framework for enforcing context-aware privacy. In: NSDI (2014). USENIX

6. Demetriou, S., Zhou, X., Naveed, M., Lee, Y., Yuan, K., Wang, X., Gunter, C.A.: What's in your dongle and bank account? mandatory and discretionary protection of android external resources. In: NDSS (2015)
7. Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B.-G., Cox, L.P., Jung, J., McDaniel, P., Sheth, A.N.: Taintdroid: an information-flow tracking system for real-time privacy monitoring on smartphones. In: ACM TOCS (2014)
8. Zhou, Y., Zhang, X., Jiang, X., Freeh, V.W.: Taming information-stealing smartphone applications (on android). In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 93–107. Springer, Heidelberg (2011)