Enforcing Least Privilege with Android Permissions in Mobile App Development

Emmanuel Bello-Ogunu, Dr. Mohamed Shehab
University of North Carolina at Charlotte
9201 University Blvd
Charlotte, NC 28223
{ebelloog,mshehab}@uncc.edu

1. INTRODUCTION

Though there is evidence that presenting Android app permission information to the user in a clear, more context-dependent way can influence mobile phone users in choosing apps that request fewer permissions [4], ultimately users still tend to make poor privacy and security decisions, especially when warnings are unclear or inhibitive [1]. As a result, we believe that code developers should take some responsibility in safeguarding users' privacy and preventing data leakage. One way to do this is by enforcing the concept of "least privilege" [5] in application development. Within this context, we are addressing the permission model in Android applications. Fewer permissions means a more effective permission system, so developers should apply this concept to the permission model. We propose PermitMe, which is a tool built as a plugin for the Eclipse IDE for static analysis on Android applications. It enforces "least privilege" by providing feedback to developers on missing or extraneous Android permissions.

2. RELATED WORK

Au et al. developed a version-independent permission plugin called PScout–short for Permission Scout–that performs static analysis to retrieve the permission specification from Android applications [2]. Evaluation was done with a sample of 1,260 applications on the basis of completeness and soundness in determining overprivileged apps. It was compared against another research tool called Stowaway [3], and PScout proved to be significantly more complete and sound in permission mappings, but ultimately there was no significant difference in overprivileging. PScout is not without limitations however. It can't handle some API calls that are invoked through reflection (admittedly neither can ours, though this is an open problem in Android development). Furthermore, PScout is very slow, taking 33 hours to extract the permission specification from the Android 4.0 framework.

Vidas et al. also developed an Eclipse plugin called Permission Check Tool, that extracts the Android permission specification from an app and aid developers in utilizing least privilege during permissioning [6]. The functionality of the Permission Check Tool is similar to that of PermitMe. However, the tool has some limitations. Its permission map only covers Android 2.2. Furthermore, it is built by parsing the Android API documentation, which is known to be incomplete. Lastly, the plugin strictly analyzes source code, and there is no ready library or dataset of Android application source code readily available, so they did not present extensive empirical analysis of the plugin. The PermitMe plugin described in the next section is an improvement over the Permission Check Tool, with empirical data collected to support this claim.

3. OUR PROPOSED APPROACH

PermitMe is compromised of an API-permission mapping database and a static analysis engine. The API-permission mapping database was composed by combining the mappings generated by both Stowaway and PScout. We generated separate mappings for each of the Android versions. The permissions requested by the current application are extracted from the `AndroidManifest.xml` file. The static analysis engine uses the Abstract Syntax Tree (AST) to scan Java source files and ASM for bytecode to find API references associated with the app. Each API reference is compared against the generated API-permission mapping to determine whether the permissions present are *missing*, used (*required*) or unused (*extraneous*).

The current Android APIs supported by the plugin are from versions 2.6 to 4.0, and the current IDE supported is the Eclipse platform, versions Indigo and Juno.

4. METHODOLOGY

We recruited 20 participants from a Mobile Application Development course being taught during the Fall 2013 semester. Of them, 60% reported an Advanced level or programming knowledge/expertise, 35% reported Moderate, and 5% reported Novice. We performed the study in an on-campus lab; participants were given debugging-related tasks for two Android applications, one requiring one permission, and the other requiring four, and both having extraneous ones. Participants were randomly assigned to one of the two conditions, either with or without use of the plugin, and upon completion of the tasks and online survey, they were paid $5 in the form of a Starbucks gift card. They were observed unobtrusively, and made aware at start of the survey that results were anonymous. Time to complete, along with Number of Missing Permissions and Number of Extra Permissions, are the measures upon which the study results were evaluated. Additionally, the survey results provided an evaluation of the Usability of the PermitMe plugin, in the areas of Ease of Use, Trustworthiness of results, Effectiveness in accomplishing what was intended, and Readability of output. We hypothesize that the PermitMe plugin is more *efficient*, *effective*, and *usable* in helping a developer reduce the number of extraneous permissions and incorporate any missing permissions than other resources.

5. RESULTS

5.1 User Study Results

For task 1, the mean completion time for participants who did not use the PermitMe plugin was 722.5 seconds (about 12 minutes), and 234.8 seconds (4 minutes) for those who used the plugin. Since not all of our participant data was normally distributed, we used the Mann-Whitney-Wilcoxon Test to compare the two groups, show statistically significant difference (p-value < 0.001). With Task 2, we found the same. Time to complete Task 2 for non-plugin users was a mean of 792.3 seconds (13 minutes), compared to 406.6 seconds (6.7 minutes) for plugin users, with a Wilcoxon test producing a p-value < 0.001. Given that completion time is our measure of efficiency, these results support our hypothesis. Table 1 shows the mean time comparisons for tasks 1 and 2.

There was no significant difference between the Missing Permissions and Extra Permissions for Task 1. This was expected given that there was only one Missing permission and one Extraneous permission. For Task 2, the mean Number of Missing Permissions for non-plugin users was 2.7, and 0 for plugin users. Using a Wilcoxon test, we found statistically significant difference between the two groups ($p < 0.001$). The mean Number of Extra Permissions for non-plugin users was 1.1, and 0.4 for plugin users. Using a Wilcoxon test, we found statistical significance between the two groups ($p < 0.05$). Table 1 shows the difference in performance between the two groups. This too supports our Hypothesis' claim of effectiveness.

Table 1: Comparison Results for Tasks 1 and 2

Measure	**Without Plugin** (μ, σ)	**With Plugin** (μ, σ)	**p-value**
Task 1			
Time (seconds)	(722.5, 194.75)	(234.8, 35.14)	< 0.001
No. Missing Perms	(0.1, 0.32)	(0.0, 0.0)	0.3434
No. Extra Perms	(0.1, 0.32)	(0.1, 0.32)	1
Task 2			
Time (seconds)	(792.3, 239.51)	(406.6, 115.41)	< 0.001
No. Missing Perms	(2.7, 1.16)	(0.0, 0.0)	< 0.001
No. Extra Perms	(1.1, 0.88)	(0.4, 0.52)	0.046

5.2 Survey Results

Analyzing the 5-point Likert-scale exit survey questions, there was no significant difference between our plugin and online forum resources with regard to Trustworthiness, Effectiveness, and Readability. However, there was a significant difference (p-value$<$0.05) for Ease of Use (4.5 compared to 3.86). This can be attributed to the difficulty involved in navigating responses sites like Stack Overflow. Overall, the plugin is as good as available online resources with regards to those first three measures, and significantly better as far as Ease of Use.

Considering the comparison between PermitMe and LogCat, the measures that proved significantly different in favor of the plugin were Ease of Use (p-value=0.009), Effectiveness (p-value=0.0258), and Readability (p-value=0.0024). This is an expected result as it is very difficult to read and use the Android LogCat output. Since Trustworthiness of LogCat and the plugin had means of 4.1 and 4.3 respectively, this meant the suggestions our plugin offered were as trusted by the users as that of the Android logging platform. Table 2 shows the usability comparisons between assistance from PermitMe plugin and LogCat. Overall, the feedback provided by the plugin was received as well as or better than many of the common resources that developers rely for assistance, so this supports the usability claim of our Hypothesis.

Table 2: Comparison with other Help Resources

Measure (5 Point Likert-Scale)	**Online Resources** (μ, σ)	**Our Plugin** (μ, σ)	**p-value**
Ease of Use	(3.86, 0.51)	(4.50, 0.71)	0.0239
Trustworthiness	(4.38, 0.72)	(4.30, 0.82)	0.8154
Effectiveness	(4.38, 0.81)	(4.60, 0.69)	0.4602
Readability	(4.50, 0.73)	(4.40, 0.84)	0.7607
Measure (5 Point Likert-Scale)	**LogCat** (μ, σ)	**Our Plugin** (μ, σ)	**p-value**
Ease of Use	(3.10, 1.29)	(4.50, 0.71)	0.0090
Trustworthiness	(4.10, 0.74)	(4.30, 0.82)	0.5744
Effectiveness	(3.60, 1.07)	(4.60, 0.67)	0.0258
Readability	(2.80, 1.14)	(4.40, 0.84)	0.0024

6. CONCLUSION

In this paper, we presented PermitMe, a tool developed for integration within the Eclipse IDE for the purpose of developing more privacy aware Android applications. This is accomplished by enforcing the principle of "least privilege" through minimization of permissions requested by an app. We hypothesized that the PermitMe plugin would prove more efficient, effective, and usable for developers than other resources. Through an empirical analysis of 20 Android developers, we found that the plugin was indeed successful in this endeavor.

7. ACKNOWLEDGMENTS

We thank Swapnil Thorat for his assistance in developing the PermitMe plugin. We would also like to thank Dr. Zachary Wartell and Osarieme Omokaro for reviewing and providing feedback on this paper.

8. REFERENCES

[1] A. Acquisti and J. Grossklags. Privacy and rationality in individual decision making. *Security & Privacy, IEEE*, 3(1):26–33, 2005.

[2] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. Pscout: analyzing the android permission specification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 217–228. ACM, 2012.

[3] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[4] N. Sadeh, L. F. Cranor, and P. G. Kelley. Privacy as part of the app decision-making process, 2013.

[5] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[6] T. Vidas, N. Christin, and L. Cranor. Curbing android permission creep. In *Proceedings of the Web*, volume 2, 2011.