Universidad Nacional de Rosario

Tesina de grado
para la obtención del grado de
Licenciado en Ciencias de la Computación

Hacia un prototipo certificado del sistema de seguridad de Android 9

Alumno:
Guido De Luca

4 de septiembre de 2019

Índice general

Estado del arte

En esta sección se describen algunos trabajos sobre el modelo de seguridad de Android. *A priori*, podemos distinguir dos grandes grupos: aquellos que reportan alguna falla puntual del sistema y los que buscan dar una descripción más general del mismo, ya sea de manera formal o informal[1].

Fallas puntuales

Dentro del primer grupo mencionado podemos encontrar bla bla bla. TODO: Buscar trabajos que apunten a vulnerabilidades específicas. El paper de Bagheri [6] tiene muchas referencias a este tipo de trabajos.

Enfoque informal

Por otro lado, encontramos trabajos como el de William Enck *et al.* [13], uno de los primeros artículos académicos en describir el modelo de seguridad de Android. A través de esta descripción, los autores buscaban *desenmascarar* la complejidad a la que debían enfrentarse los desarrolladores a la hora de construir aplicaciones seguras.

Dentro de esta línea también se encuentra el trabajo de Yasemin Acar *et al.* [1], quienes presentan una visión sistémica e integradora de las distintas líneas de investigación en seguridad de Android. Los autores realizan un análisis de los distintos enfoques desde los cuales se aborda la seguridad de este tipo de sistemas, comparándolos y estableciendo, según ellos, las bases para que la investigación futura pueda ser unificada.

Leer con más detalle [1] y ver si vale la pena profundizar

Recientemente, René Mayrhofer *et al.* [19] publicaron un trabajo de características similares a los anteriores basado en la versión 9.0 de Android. El mismo, además de dar un descripción detallada del modelo, incluye una discusión de sus implicaciones y un posterior análisis sobre las medidas que se tomaron a lo largo del tiempo para mitigar distintas amenazas.

Este tipo de trabajos constituyen un complemento importante a la documentación oficial de Android, brindándole nuevas herramientas y referencias más claras a los desarrolladores. Un ejemplo de esto fue el trabajo de Felt *et al.* [14], quienes estudiaron un grupo de aplicaciones disponibles para la versión 2.2 de Android y detectaron que muchas de ellas pedían más permisos de los que realmente necesitaban. Los autores investigaron las causas de sobreprivilegio de estas aplicaciones y encontraron que muchas veces, los desarrolladores

[1] Entendemos por enfoque informal a aquellos enfoques que no utilizan métodos formales para el estudio de la plataforma.

intentaban otorgar la menor cantidad de privilegios necesarios pero en reiteradas ocasiones fallaban por falta de una documentación precisa. En consecuencia, el grupo desarrolló Stowaway, una de las primeras herramientas dedicadas a la detección de permisos innecesarios.

Aclarar finalidad de las herramientas

Actualmente existe una gran cantidad de herramientas de análisis estático, siendo las más recientes: M-Perm [12]; IC3 [21], que incorpora el concepto de *propagación de constantes compuestas multi-valuadas* para lograr una mayor eficiencia; Covert [3] y Separ [5], que combinan análisis estático con métodos formales para inferir automáticamente propiedades sobre un conjunto de aplicaciones, a partir de las cuales se derivarán políticas de seguridad; y Droidtector [25], que a diferencia del resto no necesita el código fuente de las aplicaciones o de Android. Las herramientas de este tipo se focalizan en estudiar una aplicación en particular (o en algunos casos, un conjunto de aplicaicones) y extraer propiedades de seguridad que solo conciernen a ella. En cambio, en esta tesina estudiamos el sistema de permisos subyacente, extrayendo propiedades relevantes para todas las aplicaciones y para el sistema en general.

Enfoque formal

Uno de los primeros trabajos en abordar la seguridad de Android desde los métodos formales fue el de Chaudhuri [11]. En el mismo, se desarrolló un lenguaje que permite describir un subconjunto de aplicaciones de Android y razonar sobre ellas. Adicionalmente, se presentó un sistema de tipos para este lenguaje y se demostró un teorema que garantiza que las aplicaciones bien tipadas preservan la confidencialidad de los datos que manejan. Parcialmente inspirado en este trabajo, Bugliesi *et al.* desarrollaron π-Perm [10], un sistema de tipos y efectos que tiene como finalidad detectar problemas de *privilege escalation*. De manera análoga al trabajo de Chaudhuri, una expresión bien tipada en π-Perm garantiza que la aplicación real a la cual está representando, no es vulnerable al ataque mencionado. Similarmente, Armando *et al.* definen un lenguaje [2], junto a su semántica operacional, que permite describir interacciones entre aplicaciones. También definen un sistema de tipos y efectos basado en un formalismo del estilo del álgebra de procesos, conocido como *history expressions* [7]. Intuitivamente, una *history expression* sirve para representar los efectos laterales vinculados a la seguridad del dispositivo que se producen en una computación. Finalmente, los autores prueban que cualquier comportamiento que la plataforma pueda tener en *runtime* está contenido en este modelo, y por lo tanto, puede analizarse estáticamente.

traducción de privilege escalation

Recientemente, Wilayar Khan *et al.* retomaron el trabajo de Chaudhuri y modelaron el lenguaje en él definido dentro del *framework* lógico-matemático Coq [24]. De esta forma, pudieron no solo estudiar la corrección y seguridad de las aplicaciones de manera mecánica y rigurosa, sino que también utilizaron este asistente para probar la corrección -o *soundness*- del lenguaje en sí. En otro trabajo actual encabezado por Khan [16], se definió en Coq un modelo para estudiar el sistema de comunicación entre componentes. El principal objetivo de este trabajo es analizar la robustez de la plataforma cuando una aplicación detiene su ejecución a causa de un fallo en la resolución de un *intent*. A diferencia del resto de los trabajos citados, éste se concentra en estudiar propiedades de *safety* y no de *security*, a pesar de que este sistema puede ser explotado para filtrar información sensible de los usuarios [17].

Por otra parte, Bagheri *et al.* [4] proponen una formalización del sistema de permisos de Android escrita en Alloy [20], un lenguaje basado en la lógica relacional de primer orden, análisis capaz de realizar *bounded verification* de los modelos que en este lenguaje se describan. Con la ayuda de este modelo, los autores identificaron distintos tipos de vulnerabilidades que permiten esquivar por completo el chequeo de permisos. Particularmente, estudiaron la vulnerabilidad de permisos personalizados, mediante la cual una aplicación maliciosa puede acceder a todos los recursos de otra que estén protegidos por permisos personalizados. Esta falla surge de que el sistema no impone restricciones con respecto al nombre de los nuevos permisos que definen y, como consecuencia, dos permisos distintos podrían tener el mismo nombre. Este trabajo luego se extendió para una nueva versión de Android [6]. La falla por permisos personalizados ya había sido reportada por Shin *et al.* en [23]. Una diferencia fundamental entre este enfoque y el de esta tesina es el tipo de análisis que se realizó. A pesar de que Alloy es capaz de producir contraejemplos de manera automática, algo realmente útil a la hora de buscar potenciales fallas; no es posible demostrar propiedades de una manera rigurosa y formal.

traducción de bounded verifiaction?

Carlos: debería profundizar más en la diferencia?

En trabajos previos encabezados por Gustavo Betarte y Carlos Luna [8, 9, 18], se utilizó el asistente de pruebas Coq para modelar un sistema de transición de estados que representa, principalmente, los distintos estados que atraviesa la plataforma cuando se realizan operaciones sobre ella (por ejemplo, al instalar o desinstalar una aplicación). A partir de esta especificación, no solo se probaron propiedades relevantes a la seguridad del modelo, sino que también se extrajo una implementación certificada del mismo. Los autores explican cómo esta implementación puede utilizarse para generar casos de pruebas abstractos dentro del *testing* basado en modelos, o bien, cómo puede usarse para monitorear las acciones realizadas en un sistema real y evaluar si las propiedades deseadas efectivamente se cumplen. Estos trabajos presentan el modelo que será actualizado y extendido en esta tesina.

explicar de qué se trata un sist. de trans. de estados?

Un enfoque similar a este, es el de Wook Shin *et al.*, quienes también utilizaron Coq para representar el modelo [22]. Sin embargo, esta formalización no considera aspectos de la plataforma que sí son considerados por los trabajos anteriores (y por ende, por esta tesina); como por ejemplo, los distintos tipos de componentes, la interacción entre instancias de aplicaciones en ejecución y el sistema, la operación de escritura/lectura en un *content provider* y la semántica del sistema de delegación de permisos. Al mismo tiempo, cuando Android incorporó los permisos otorgados en *runtime*, este modelo no fue actualizado. El trabajo de Fragkaki *et al.* tambień presenta un modelo formal basado en transiciones de estado [15], pero el mismo no está desarrolado dentro de un *framework* que permita realizar pruebas asisistidas por computadora. Además, el modelo se corresponde con una de las primeras versiones de Android, por lo que tampoco contempla los cambios en el sistema de permisos introducidos a partir de la versión 6.0.

Bibliografía

[1] Y. Acar y col. "SoK: Lessons Learned from Android Security Research for Appified Software Platforms". En: *2016 IEEE Symposium on Security and Privacy (SP)*. Mayo de 2016, págs. 433-451. DOI: 10.1109/SP.2016.33.

[2] Alessandro Armando, Gabriele Costa y Alessio Merlo. "Formal Modeling and Reasoning about the Android Security Framework". En: *Trustworthy Global Computing*. Ed. por Catuscia Palamidessi y Mark D. Ryan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, págs. 64-81. ISBN: 978-3-642-41157-1.

[3] H. Bagheri y col. "COVERT: Compositional Analysis of Android Inter-App Permission Leakage". En: *IEEE Transactions on Software Engineering* 41.9 (sep. de 2015), págs. 866-886. ISSN: 0098-5589. DOI: 10.1109/TSE.2015.2419611.

[4] H. Bagheri y col. "Detection of design flaws in the Android permission protocol through bounded verification". En: *Proceedings of the 2015 International Symposium on Formal Methods* volume 9019 of Lecture Notes in Computer Science (2015), págs. 73-89.

[5] H. Bagheri y col. "Practical, Formal Synthesis and Automatic Enforcement of Security Policies for Android". En: *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. Jun. de 2016, págs. 514-525. DOI: 10.1109/DSN.2016.53.

[6] Hamid Bagheri y col. "A Formal Approach for Detection of Security Flaws in the Android Permission System". En: *Springer Journal on Formal Aspects of Computing* (2017). DOI: 10.1007/s00165-017-0445-z.

[7] Massimo Bartoletti y col. "Types and Effects for Resource Usage Analysis". En: *Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures*. FOSSACS'07. Braga, Portugal: Springer-Verlag, 2007, págs. 32-47. URL: http://dl.acm.org/citation.cfm?id=1760037.1760043.

[8] Gustavo Betarte y col. "A certified reference validation mechanism for the permission model of Android". En: *CoRR* abs/1709.03652 (2017). arXiv: 1709.03652. URL: http://arxiv.org/abs/1709.03652.

[9] Gustavo Betarte y col. "Formal Analysis of Android's Permission-Based Security Model". En: *Scientific Annals of Computer Science* 26 (jun. de 2016), págs. 27-68. DOI: 10.7561/SACS.2016.1.27.

[10] Michele Bugliesi, Stefano Calzavara y Alvise Spanò. "Lintent: Towards Security Type-Checking of Android Applications". En: *Formal Techniques for Distributed Systems*. Ed. por Dirk Beyer y Michele Boreale. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, págs. 289-304. ISBN: 978-3-642-38592-6.

[11] Avik Chaudhuri. "Language-based Security on Android". En: *Proceedings of the ACM SIGPLAN Fourth Workshop on Programming Languages and Analysis for Security*. PLAS '09. Dublin, Ireland: ACM, 2009, págs. 1-7. ISBN: 978-1-60558-645-8. DOI: 10.1145/1554339.1554341. URL: http://doi.acm.org/10.1145/1554339.1554341.

[12] P. Chester y col. "M-Perm: A Lightweight Detector for Android Permission Gaps". En: *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*. Mayo de 2017, págs. 217-218. DOI: 10.1109/MOBILESoft.2017.23.

[13] W. Enck, M. Ongtang y P. McDaniel. "Understanding Android Security". En: *IEEE Security Privacy* 7.1 (ene. de 2009), págs. 50-57. ISSN: 1540-7993. DOI: 10.1109/MSP.2009.26.

[14] Adrienne Porter Felt y col. "Android Permissions Demystified". En: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS '11. Chicago, Illinois, USA: ACM, 2011, págs. 627-638. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046779. URL: http://doi.acm.org/10.1145/2046707.2046779.

[15] Elli Fragkaki y col. "Modeling and Enhancing Android's Permission System". En: *Computer Security – ESORICS 2012*. Ed. por Sara Foresti, Moti Yung y Fabio Martinelli. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, págs. 1-18. ISBN: 978-3-642-33167-1.

[16] Wilayat Khan y col. "CrashSafe: A Formal Model for Proving Crash-safety of Android Applications". En: *Hum.-centric Comput. Inf. Sci.* 8.1 (dic. de 2018), 144:1-144:24. ISSN: 2192-1962. DOI: 10.1186/s13673-018-0144-7. URL: https://doi.org/10.1186/s13673-018-0144-7.

[17] L. Li y col. "IccTA: Detecting Inter-Component Privacy Leaks in Android Apps". En: *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*. Vol. 1. Mayo de 2015, págs. 280-291. DOI: 10.1109/ICSE.2015.48.

[18] Carlos Luna y col. "A formal approach for the verification of the permission-based security model of Android". En: *CLEI Electronic Journal* 21 (ago. de 2018), 3:1-3:22. DOI: 10.19153/cleiej.21.2.3.

[19] R. Mayrhofer y col. *The Android Platform Security Model*. 2019. URL: https://arxiv.org/abs/1904.05572.

[20] Software Design Group at MIT. *Alloy*. URL: http://alloytools.org/ (visitado 20-08-2019).

[21] Damien Octeau y col. "Composite Constant Propagation: Application to Android Inter-component Communication Analysis". En: *Proceedings of the 37th International Conference on Software Engineering - Volume 1*. ICSE '15. Florence, Italy: IEEE Press, 2015, págs. 77-88. ISBN: 978-1-4799-1934-5. URL: http://dl.acm.org/citation.cfm?id=2818754.2818767.

[22] W. Shin y col. "A Formal Model to Analyze the Permission Authorization and Enforcement in the Android Framework". En: *2010 IEEE Second International Conference on Social Computing*. Ago. de 2010, págs. 944-951. DOI: 10.1109/SocialCom.2010.140.

[23] W. Shin y col. "A Small But Non-negligible Flaw in the Android Permission Scheme". En: *2010 IEEE International Symposium on Policies for Distributed Systems and Networks*. Jul. de 2010, págs. 107-110. DOI: 10.1109/POLICY.2010.11.

[24] *The Coq Proof Assistant*. https://coq.inria.fr/.

[25] S. Wu y J. Liu. "Overprivileged Permission Detection for Android Applications". En: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. Mayo de 2019, págs. 1-6. DOI: 10.1109/ICC.2019.8761572.