JURNAL TEKNIK INFORMATIKA DAN MULTIMEDIA Vol 3, No.1, Mei 2023, pp. 104 - 123
p-ISSN : 2808-909X (print)
e-ISSN : 2798-2505 (online)
http://journal.politeknik-pratama.ac.id/index.php/JTIM · page 104

ANALISIS KREDENSIAL UNTUK KONFIGURASI KEAMANANAN ROM ANDROID CUSTOM

Pieter Santoso Hadi[1], Edy Siswanto[2]

[1] Program Studi Komputer Grafis Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: pieterzzz8@gmail.com
[2] Program Studi Komputerisasi Akuntansi Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: edy@stekom.ac.id

ARTICLE INFO

Article history:

Received 10 April 2023
Received in revised form 24 April 2023
Accepted 8 Mei 2023
Available online 15 Mei 2023

ABSTRACT

Android is a layered operating system, where each layer has its own duties and responsibilities. Various parties in the customization chain such as device vendors such as Samsung, Xiaomi, Oppo, Huawei and others, operators such as Telkomsel, Smartfren, XL, etc. and hardware manufacturers can customize one or more layers to adapt devices for different purposes, such as supporting specific hardware and providing different interfaces and services. The purpose of this study was to investigate systematically for any inconsistencies that arose as a result of the processes involved in this research and to assess their various security implications. This research runs DroidDiff to perform large-scale differential analysis on images collected using the analysis methodology. DroidDiff found a lot of differences when it comes to the selection features. The method used in this study is the method of five differential analysis algorithms. As a result, by comparing the security configurations of similar figures, important security changes that could be accidentally introduced during customization can be found. The results show that DroidDiff can be used by vendors to check the configuration of various security features in a given image. DroidDiff will extract those features from the image, and compare them to other image configuration sets, then DroidDiff will flag the inconsistent ones for further investigation by vendors who have the source code and tools to check their effect. For future work, improvements to DroidDiff to more accurately detect risky inconsistencies are highly recommended. Improving DroidDiff will help reduce the number of false positives and determine risky configurations more accurately.

Keywords: Analisis Kredensial, Konfigurasi Keamanan, Android, Kustomisasi Android

Received April 10, 2023; Accepted Mei 8, 2022; Available online Mei 15, 2023

p-ISSN : 2808-909X e-ISSN : 2798-2505

Abstrak

Android merupakan sistem operasi berlapis, dimana setiap lapisan memiliki tugas dan tanggung jawabnya masing-masing. Berbagai pihak dalam rantai penyesuaian seperti vendor perangkat misal saja Samsung, Xiaomi, Oppo, Huawei dan lain sebagainya, operator seperti terlkomsel, smartfren, XL, dan lain sebagainya dan produsen perangkat keras dapat menyesuaikan satu atau beberapa lapisan untuk menyesuaikan perangkat untuk tujuan yang berbeda, seperti mendukung perangkat keras khusus dan menyediakan antarmuka dan layanan yang berbeda. Tujuan penelitian ini adalah untuk menyelidiki secara sistematis untuk setiap ketidakkonsistenan yang muncul sebagai hasil dari proses yang dilakukan dalam penelitian ini serta untuk menilai berbagai implikasi keamanannya. Penelitian ini menjalankan DroidDiff untuk melakukan analisis diferensial skala besar pada gambar yang di kumpulkan menggunakan metodologi analisis. DroidDiff menemukan banyak perbedaan terkait dengan fitur pilihan. Metode yang diguanakan dalam penelitian ini adalah metode lima algoritme analisis diferensial. Hasilnya, dengan membandingkan konfigurasi keamanan dari gambar serupa, perubahan keamanan penting yang bisa jadi secara tidak sengaja diperkenalkan selama penyesuaian dapat ditemukan. Hasil menunjukkan bahwa, DroidDiff dapat diguanakan vendor guna memeriksa konfigurasi berbagai fitur keamanan pada image yang diberikan. DroidDiff akan mengekstrak fitur tersebut dari gambar, dan membandingkannya dengan kumpulan konfigurasi gambar lain, selanjutnya DroidDiff akan menandai yang tidak konsisten untuk diselidiki lebih lanjut oleh vendor yang memiliki kode sumber dan perangkat untuk memeriksa efeknya. Untuk pekerjaan dimasa depan, penyempurnaan DroidDiff dalam mendeteksi ketidakkonsistenan yang berisiko secara lebih akurat sangat disarankan. Dengan meningkatkan DroidDiff akan membantu mengurangi jumlah positif palsu dan menentukan konfigurasi berisiko secara lebih akurat.

Keywords: Analisis Kredensial, Konfigurasi Keamanan, Android, Kustomisasi Android

1. PENDAHULUAN

Layaknya daun yang bersemi dimusim hujan, pasar smartphone berkembang dengan cepat kaarena berbagai alasan. Smartphone dengan OS Android ini memiliki popularitas yang begitu diberbagai kalangan tanpa batas usia. Android mendorong inovasi dan menghadirkan lebih banyak fitur untuk penggunanya, ini membuatnya menjadi perangkat yang kaya fitur dan aplikasi seluler yang menarik. Sifat terbuka ekosistem Android yang diadopsi oleh Google secara alami memberikan dasar bagi sistem operasi yang sangat terfragmentasi. Versi resmi android telah disesuaikan secara agresif menjadi ribuan gambar sistem oleh semua orang dalam rantai penyesuaian dengan pabrikan hardware, vendor device, dan operator bebas membuat dasar untuk menyesuaikannya dengan berbagai fitur dan model dalam upaya membedakan produk mereka dari pesaing. Karena vendor perangkat lebih mahir dalam mengubah kerangka kerja Android dan aplikasi sistem default, membuat berbagai fitur seperti kamera yang akhirnya memiliki resolusi semakin tinggi dan semakin inovatif membuat kustomisasi Android menjadi lebih lazim dari waktu ke waktu. Akan tetapi, kustomisasi tersebut justru mendatangkan masalah, dan memberikan kerentanan keamanan yang serius dan pada akhirnya memberikan kerusakan parah. Mengingat hal tersebut, serta keamanan kustomisasi Android konsumen yang luas dipertaruhkan maka penting untuk mempelajari dan menyelidiki konsekuensi keamanan Android.

Proses Kustomisasi Android

Lapisan aplikasi, terletak di bagian atas model arsitektur Android dan menyediakan aplikasi inti seperti Beranda, Kontak, Telepon, dan Peramban, sering disesuaikan oleh vendor dan operator perangkat. Penelitian oleh Jiang et al., (2013) telah menunjukkan bahwa lapisan ini selalu menjadi fokus kustomisasi vendor. Vendor dan operator memodifikasi aplikasi AOSP yang dimuat ulang secara default untuk menyediakan fungsionalitas dan UI yang lebih canggih. Lapisan Kerangka dan Perpustakaan terletak tepat di bawah lapisan Aplikasi dan memberikan dukungan bagi pengembang untuk mengakses berbagai sumber daya, layanan, dan fungsionalitas istimewa. Di bagian bawah lapisan adalah kernel Linux, menyediakan tingkat abstraksi antara perangkat keras perangkat dan berisi semua driver perangkat keras penting seperti tampilan, kamera, dll.

Bahaya Kustomisasi Android

Kustomisasi perangkat Android memberikan peluang dalam meningkatkan fungsi dan personalisasi secara sempurna untuk pelanggannya, akan tetapi, ini juga memberikan peluang peningkatan risiko keamanan, mengingat proses fragmentasi Android sangat tidak diatur. Untuk mengendalikannya,

p-ISSN : 2808-909X e-ISSN : 2798-2505

Google telah meluncurkan Program Kompatibilitas Android untuk memandu proses penyesuaian. Pada lapisan kernel Linux, studi Wang et al., (2014) menyelidiki konfigurasi keamanan driver perangkat Linux Android, dan menemukan bahwa banyak dari perangkat ini belum terlindungi dengan baik, menyebabkan eksposur mereka ke pihak yang seharusnya tidak mengaksesnyadapat langsung memerintahkan driver kamera yang terbuka untuk mengambil gambar. Penelitian oleh Jiang et al., (2013) mengungkapkan bahwa lapisan Aplikasi juga penuh dengan kekurangan yang diperkenalkan selama proses kustomisasi. Demikian pula, pekerjaan Jiang et al., (2012) secara anekdot menunjukkan bahwa aplikasi pramuat vendor Android memiliki kelemahan keamanan yang dikirimkan pada beberapa perangkat khusus.

LITERATUR REVIEW

Kustomisasi Android - Vendor menyesuaikan kerangka kerja Android untuk mendukung fitur yang lebih canggih dan menawarkan pengalaman pengguna yang unik. Semakin memperumit proses, garis dasar AOSP telah banyak dimodifikasi menjadi versi Android yang berbeda. Dari 682.000 perangkat yang disurvei, lebih dari 24.000 perangkat berbeda dan 1294 merek berbeda telah diidentifikasi, di mana porsi terbesar (37,8%) diproduksi oleh Samsung. **Komponen Perangkat Keras Khusus -** Setiap saat, perangkat Android mungkin menjalankan komponen perangkat keras yang berbeda. Pabrikan perangkat keras dan chipset menyesuaikan perangkat Android untuk memberikan tingkat kinerja yang lebih baik dan perangkat keras yang lebih canggih. Vendor mengadaptasi tumpukan perangkat lunak Android dasar ke platform perangkat keras khusus vendor baru dengan mengintegrasikan beberapa tambahan ke OS Android. **Antarmuka Pengguna Khusus Vendor Perangkat -** Beberapa vendor mendesain UI mereka sendiri, setiap UI kustom menggunakan palet warna dan elemen UI-nya sendiri, misalnya perbedaan UI pada samsung dan Xiaomi Redmi, masing-masing menghadirkan pengaturan yang berbeda secara estetika. Vendor perangkat mengintegrasikan UI kustom ini ke dalam perangkat mereka sendiri dengan menambahkan dan mengubah elemen UI Stock Android yang ada. **Fitur Khusus Operator -** Berdasarkan persyaratan jaringan operator (Tsel, XL, 3 dan lain sebagainya), vendor perangkat memodifikasi layanan telepon untuk memungkinkan pengintegrasian berbagai pita LTE dan GSM, bahkan jaringan 5G yang saat ini sudah rilis. Vendor perangkat juga harus melakukan serangkaian perubahan lapisan Android, untuk mendukung pembatasan operator tertentu. **Pembaruan Android -** Proses penyesuaian Android dengan langkah cepat AOSP memperbarui versi OS-nya, yang sebagian besar dari sudah sangat disesuaikan dan memunculkan ribuan cabang Android yang disesuaikan hidup berdampingan di miliaran ponsel di seluruh dunia dengan berbagai tingkat versi dan hidup berdampingan di pasar saat ini.

Efek Kustomisasi

Masalah Kompatibilitas dan Portabilitas

Setiap level API Android baru memperkenalkan fitur dan menghapus bug. Bahkan jika setiap rilis bertujuan untuk mengintegrasikan perubahan baru tanpa merusak aplikasi pra-instal yang ada di versi lama, sering kali kompatibilitas yang sempurna tidak dapat dicapai. Karena pengujian lintas platform dan lini produk yang tidak mencukupi untuk fitur dan perubahan baru, aplikasi Android mungkin tidak berperilaku secara konsisten di berbagai versi. Untuk memastikan perilaku aplikasi Android yang konsisten dan benar di berbagai level API dan konfigurasi perangkat keras, Google memperkenalkan Firebase Test Lab (FTL) for Android, infrastruktur berbasis cloud untuk pengujian komprehensif aplikasi Android sebelum dirilis, melalui FTL untuk android ini developer dapat mengakses berbagai perangkat fisik Android yang diinstal di pusat data Google dan menguji aplikasi mereka di berbagai merek dan model, di berbagai API Android, konfigurasi perangkat, dan orientasi layar serta untuk mengatasi fragmentasi dan memastikan kompatibilitas di seluruh perangkat Android, Google meluncurkan Program Kompatibilitas sebelumnya.

Arsitektur Berlapis Android

Untuk memungkinkan pengembang aplikasi mengakses berbagai sumber daya dan fungsionalitas, lapisan Kerangka Kerja Android menyediakan banyak layanan tingkat tinggi seperti PackageManager, ActivityManager, NotificationManager, dan banyak lainnya. Layanan ini memediasi akses ke sumber daya sistem dan menegakkan kontrol akses yang tepat berdasarkan beberapa kriteria seperti id pengguna aplikasi, izin Android yang diperoleh, Signature, dll. Tepat di bawah lapisan kerangka kerja terdapat lapisan Perpustakaan, yang merupakan kumpulan pustaka khusus Android dan inti pustaka lainnya seperti libc, basis data SQLite, pustaka media, dll. Sama seperti layanan kerangka kerja, pustaka khusus Android tertentu melakukan berbagai pemeriksaan kontrol akses berdasarkan kriteria serupa (misalnya, id pengguna pemanggil, izin, dll). Di bagian bawah lapisan Android terdapat kernel Linux yang menyediakan tingkat abstraksi antara perangkat keras perangkat dan lapisan atasnya. Ini berisi semua fungsionalitas sistem inti

p-ISSN : 2808-909X e-ISSN : 2798-2505

tingkat rendah seperti manajemen memori, proses dan manajemen daya dan menyediakan driver perangkat keras penting seperti kamera, tampilan perangkat, Wifi, dll. Lapisan kernel Linux memediasi akses ke driver perangkat keras dan sumber daya mentah berdasarkan Akses Diskresioner standar Kontrol (DAC).

Bahaya Keamanan Kustomisasi Android - Penelitian Jiang et al., (2012) yang secara sistematis mempelajari 8 smartphone Android populer dari berbagai produsen mengungkapkan bahwa gambar ponsel ini tidak menegakkan model keamanan berbasis izin dengan benar. Beberapa izin istimewa atau berbahaya yang melindungi data sensitif pengguna dalam aplikasi yang dimuat sebelumnya akan diekspos ke aplikasi yang tidak memiliki hak istimewa. Penelitian Wang et al., (2014) menganalisis konfigurasi keamanan driver perangkat Linux Android dalam upaya untuk menemukan bahaya fragmentasi pada lapisan Kernel, Wang et al., (2014) melakukan studi sistematis tentang bahaya keamanan kustomisasi perangkat Android melalui identifikasi otomatis file Linux yang terkait dengan operasi pada driver perangkat tertentu kemudian membandingkan tingkat perlindungannya (Linux bit izin file untuk setiap file individu) pada versi vendor dengan versi AOSP yang sesuai. Setiap perlindungan lemah yang terdeteksi pada driver perangkat vendor menyiratkan potensi bahaya keamanan. Adanya ketidaksesuaian izin file Linux di dua OS serupa, bersama dengan hubungannya dengan izin Android yang berbahaya cukup mengkhawatirkan. Tian et al., (2014) mengaudit ponsel Android pihak ketiga dengan membandingkannya secara berdampingan dengan sistem operasi Android resmi untuk menemukan potensi kerentanan keamanan dan kelemahan desain yang menjalar melalui kustomisasi vendor. Tian et al., (2014) mengekstrak aplikasi dan pustaka prainstal dari image Android kustom, membangun sistem yang cocok dari AOSP, lalu membandingkan aplikasi dan pustaka prainstal untuk menemukan modifikasi dan menilai keamanannya. Penelitian oleh Gallo et al (2015), menyoroti masalah keamanan dalam model izin Android berkaitan dengan kustomisasi Android.

Peningkatan Keamanan Android. Wallach et al., (2011) mengusulkan mekanisme keamanan untuk mengatasi masalah deputi yang membingungkan melalui pelacakan rantai panggilan IPC dan mengizinkan aplikasi pilihan untuk beroperasi dengan hak istimewa penelepon yang dikurangi atau menggunakan hak istimewa penuhnya. Demikian pula, penelitian Wetherall et al., (2011) menghadirkan kontrol privasi baru untuk melindungi data sensitif pengguna melalui penyediaan data bayangan sebagai pengganti data pribadi, dan melalui pemblokiran eksfiltrasi. Penelitian Zhang et et al., (2009) mengembangkan kerangka kerja Perpanjangan Izin Android (Apex), mekanisme penegakan kebijakan komprehensif untuk platform Android yang memungkinkan pengguna untuk secara selektif memberikan izin ke aplikasi serta memberlakukan batasan pada penggunaan sumber daya. Untuk itu, penulis mendefinisikan semantik dan model kebijakan yang digunakan untuk menggambarkan kendala tersebut. Kerangka kerja lain seperti XManDroid (Sadeghi et al., (2011)) dan TrustDroid (Shastry et al., (2011)) fokus pada mediasi komunikasi antar komponen dalam aplikasi yang berbeda. FlaskDroid (Sadeghi et al., (2013)) dan proyek SEAndroid (Craig et al., (20130) juga memediasi interaksi komponen sebagai bagian dari penerapannya. SEAndroid memecahkan tantangan rumit secara teknis untuk mem-porting kontrol akses wajib berbasis SELinux dari domain desktop ke Android.

Di jalur yang berbeda, revDroid (Chen et al., (2016)) bertujuan untuk menilai efek samping (seperti crash aplikasi) dari pemberian dan pencabutan izin selektif yang diusulkan oleh karya-karya ini (Wetherall et al., (2011); Zhang et et al., (2009)), melalui analisis statis otomatis yang menghitung pengecualian keamanan yang tidak tertangani yang disebabkan dengan pencabutan izin. Wetherall et al., (2011) menyediakan mekanisme tambahan untuk membayangi data sensitif dan untuk memblokir kebocoran yang tidak sah melalui jaringan melalui pelacakan noda yang halus. YAASE (Zhauniarovich et al., (2011)) mencakup tainting untuk mencegah serangan eskalasi wakil dan hak istimewa yang membingungkan. Penelitian Feth&Jung et al., (2012) di sisi lain, menggunakan pelacakan noda untuk menegakkan kontrol penggunaan berbasis data di lingkungan bisnis. AdDroid (Nunez et al., (2012)) dan AdSplit (Wallach et al., (2012)) menyediakan kontrol akses yang sangat halus di tingkat komponen. Keduanya berupaya membatasi komponen pihak ketiga yang tidak tepercaya (yaitu, iklan) di dalam aplikasi. AdDroid (Nunez et al., (2012)) merangkum pustaka periklanan ke dalam kerangka kerja Android untuk meningkatkan tingkat kepercayaan mereka, yang hanya dapat terwujud jika vendor perangkat mencapai kesepakatan dengan semua perusahaan periklanan. AdSplit (Wallach et al., (2012)), di sisi lain, mengisolasi iklan ke dalam aktivitas terpisah dari aplikasi, sehingga aktivitas iklan ditempatkan di bawah aktivitas aplikasi. Namun, pendekatan ini sepertinya tidak akan diadopsi dalam praktiknya karena menimbulkan risiko atas transparansi aplikasi, sehingga mungkin mengarah ke serangan Click Jacking dan variasinya. AFrame (Du et al., (2013)) menyajikan pendekatan isolasi berbeda dengan mengandalkan id proses. Solusinya menempatkan iklan dan aplikasi yang memuatnya pada permukaan gambar yang sama

p-ISSN : 2808-909X e-ISSN : 2798-2505

tetapi dalam proses yang berbeda. Dengan demikian, pendekatan ini tidak mengalami keterbatasan yang diwarisi dari penggunaan teknik transparansi.

Pembaruan Android. Beberapa peneliti mencoba mengidentifikasi kerentanan yang disebabkan oleh siklus hidup aplikasi Android yang sangat cepat dan pembaruan yang sering dirilis oleh Google. Thomas et al., (2015) mengumpulkan korpus dari 20400 perangkat Android khusus dan menunjukkan bahwa ada variabilitas yang signifikan dalam pengiriman pembaruan keamanan ke perangkat Android yang diproduksi oleh vendor dan operator yang berbeda; mengarah ke kerentanan keamanan yang diketahui belum ditambal. Faktanya, penelitian tersebut mengungkapkan bahwa 87,7% dari perangkat Android yang dikumpulkan memiliki kerentanan keamanan yang besar dan terpapar setidaknya pada 1 ancaman utama. Pekerjaan penelitian (Yuan et al., (2014)) mengungkapkan kelas serangan lain yang disebabkan selama pembaruan OS Android di mana penyerang dapat secara strategis meminta izin dan atribut lainnya, yang tersedia di versi OS mendatang, untuk meningkatkan hak istimewanya setelah pembaruan dilakukan.

Serangan Pembajakan Komponen di Android.

Efek keamanan dari mengekspor penyedia konten telah dipelajari oleh Zhou et al., (2013) termasuk kebocoran konten dan polusi konten. Serangan pendelegasian ulang izin (Chin et al., (2011))menggambarkan konsekuensi lain dari ekspor antarmuka publik yang tidak disengaja, di mana aplikasi dengan izin melakukan tugas istimewa atas nama aplikasi tanpa izin itu; dengan demikian, penyerang dapat menggunakan kemampuan istimewa tanpa memperoleh izin Android yang sesuai.

Malware Android.

Potharaju et al., (2012) mengusulkan sinyal risiko yang berbeda berdasarkan izin yang diminta, kategori, serta izin yang diminta dari aplikasi yang termasuk dalam kategori yang sama. Dalam penelitian Molloy et al., (2012) menggunakan model generatif probabilistik untuk menghitung skor risiko nyata dari aplikasi Android berdasarkan izin yang mereka minta. Pekerjaan lain untuk mendeteksi malware melalui informasi level bytecode termasuk (AASandbox oleh Albayrak et al., (2010)) yang mengandalkan pendekatan coba-coba untuk mengidentifikasi pola yang mencurigakan dalam kode sumber, dan DroidRanger (Jiang et al., (2012)) yang mendeteksi malware Android berdasarkan kesamaan izin yang diminta dan jejak perilaku untuk berbagai keluarga malware yang dikenal, dirumuskan melalui pemfilteran berbasis heuristik. DroidAPIMiner melakukan analisis frekuensi panggilan API secara menyeluruh dalam aplikasi jinak dan berbahaya untuk mengekstrak fitur malware dan menggunakan pembelajaran mesin untuk mendapatkan fitur yang paling relevan. Camtepe et al., (2009) mengekstrak pustaka dan panggilan fungsi sistem dari executable Android dan membandingkannya dengan executable malware untuk mengklasifikasikan aplikasi. Zurutuza et al., (2011) mengumpulkan jejak panggilan sistem dari aplikasi yang sedang berjalan pada perangkat Android yang berbeda dan menerapkan algoritme pengelompokan untuk mendeteksi perilaku berbahaya. AsDroid Liang et al., (2014) mendeteksi perilaku aplikasi tersembunyi dengan mengidentifikasi ketidaksesuaian antara pemanggilan API dan teks yang ditampilkan di GUI. Rieck et al., (2014) mengekstraksi beberapa fitur dari aplikasi Android dan menerapkan teknik pembelajaran mesin untuk melakukan klasifikasi. Zhao et al., (2014) mengekstrak fitur klasifikasi yang lebih canggih untuk melawan varian malware dan malware zero-day. Lebih khusus lagi, mereka mengekstraksi grafik dependensi API kontekstual berbobot sebagai semantik program untuk membuat kumpulan fitur dan memperkenalkan metrik kesamaan grafik untuk mengungkap perilaku aplikasi yang homogen sembari menoleransi perbedaan implementasi kecil.

Andromaly (Weiss et al., (2012)) terus memantau berbagai metrik sistem untuk mendeteksi aktivitas mencurigakan melalui penerapan teknik deteksi anomali yang diawasi. Sheth et al., (2014) melakukan analisis noda dinamis untuk melacak aliran data pribadi dan sensitif melalui aplikasi pihak ketiga, dan mendeteksi kebocoran apa pun ke server jarak jauh. Bos et al., (2010) mengusulkan model keamanan untuk melindungi perangkat seluler yang melakukan beberapa teknik deteksi serangan secara bersamaan di server jarak jauh yang menghosting replika perangkat yang tepat. Penelitian FlowDroid (Traon et al., (2014)) dan DroidSafe (Rinard et al., (2015)) yang mengusulkan analisis noda statis yang tepat untuk mendeteksi aliran data yang berpotensi berbahaya. Demikian pula, AppSealer (Zhang&Yin et al., (2014)), Capper (Zhang&Yin, (2014)), PEG (Song et al., (2013)) melakukan analisis aliran data statis untuk mengidentifikasi kode berbahaya di aplikasi Android. AppContext (Andow et al., (2015)) adalah sistem yang memanfaatkan pembelajaran mesin yang diawasi untuk mengklasifikasikan perilaku yang berpotensi berbahaya dengan mempertimbangkan konteks di mana perilaku tersebut dijalankan. Pendekatan yang lebih mirip (Barbara et al., (2016)) juga berbagi pengamatan yang sama dengan AppContext; yang

p-ISSN : 2808-909X e-ISSN : 2798-2505

hanya melihat perilaku saja tidak cukup untuk melakukan klasifikasi yang tepat, dan mengusulkan solusi untuk mendeteksi bom logika (yaitu, perilaku jahat yang dipicu dalam kondisi khusus).

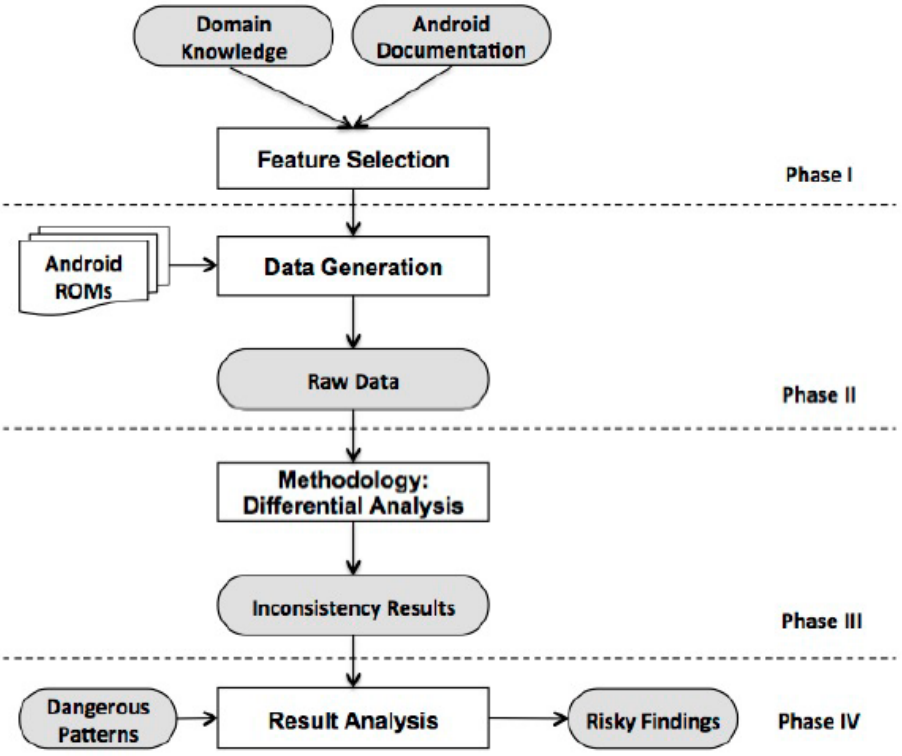
Yan&Yin, (2012) menghadirkan DroidScope dan membangun kembali sistem operasi dan semantik tingkat Java, dan memungkinkan instrumentasi Dalvik dan instruksi asli. Akibatnya, DroidScope dapat digunakan untuk memahami perilaku malware baik pada tingkat kode asli maupun interaksi dengan sistem. Pada baris yang sama, AppFence (Wetherall et al., (2011)) adalah kerangka kerja dinamis yang diimplementasikan sebagai modifikasi kerangka kerja Android yang mencegah serangan terhadap privasi pengguna melalui bayangan data. Karya penelitian lain, seperti Mobile Sandbox (Hoffmann et al.,(2013)), CopperDroid(Cavallaro et al., (2015)), Andrubis (Platzer et al., (2014)), VetDroid (Zang et al., (2013)) mengembangkan alat dan teknik untuk menganalisis secara dinamis aplikasi Android yang tidak dikenal untuk potensi perilaku jahat. Garis penelitian lain mengusulkan solusi berdasarkan analisis dinamis untuk melakukan eksekusi multi-jalur dan eksekusi simbolis dinamis dari aplikasi Android yang tidak dikenal (Mahmood et al., (2012); Wong&Lie, (2016) dan Miltenberger et al., (2016)).

Ekosistem Android yang terfragmentasi membawa beberapa kerentanan keamanan ketika vendor mengubah fungsionalitas dan konfigurasi tanpa pemahaman yang komprehensif tentang implikasinya. Wu et al., (2013) menganalisis beberapa stok image Android dari vendor yang berbeda, dan menilai masalah keamanan yang mungkin ditimbulkan oleh kustomisasi vendor. Hasil mereka menunjukkan bahwa kustomisasi bertanggung jawab atas sejumlah masalah keamanan mulai dari aplikasi sistem yang terlalu diistimewakan hingga buggy yang dapat dieksploitasi untuk melakukan pendelegasian ulang izin atau serangan kebocoran izin.. Aplikasi berbahaya dapat "meniru" komponen yang hilang untuk meluncurkan eskalasi hak istimewa, kebocoran informasi, dan serangan phishing. ADDICTED (Wang et al., (2014)) menemukan bahwa banyak perangkat Android kustom tidak melindungi driver perangkat Linux dengan benar, mengekspos mereka ke pihak yang tidak sah. Penelitian ini mengusulkan DroidDiff, alat yang mendeteksi konfigurasi keamanan yang tidak konsisten dalam skala besar, dan yang dapat digunakan oleh vendor untuk menemukan konfigurasi berisiko yang dibuat secara tidak sengaja. Ini dimulai dengan arsitektur berlapis Android dan mencantumkan pemeriksaan kontrol akses yang digunakan di setiap lapisan. Kemudian, untuk setiap pemeriksaan kontrol akses mengandalkan dokumentasi Android dan pengetahuan domain untuk menentukan fitur keamanan yang sesuai. Konfigurasi yang berbeda dari fitur ini juga dianalisis di seluruh gambar kustom dapat menyebabkan ketidakkonsistenan dan dengan demikian memengaruhi semantik pemeriksaan kontrol akses kemudian mengekstrak fitur-fitur ini dari 591 ROM Android kustom yang telah dikumpulkan dari berbagai sumber. Langkah ini menghasilkan data mentah yang akan digunakan untuk analisisa. Hasil DroidDiff mengungkapkan bahwa proses penyesuaian menyebabkan inkonsistensi di antara fitur keamanan, mulai dari mengubah tingkat perlindungan izin, menghapus definisi broadcast yang dilindungi, mengubah persyaratan untuk memperoleh GID penting, dan mengubah konfigurasi perlindungan komponen aplikasi.

Metodologi

Ekstraksi Fitur

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 1. Alur Investigasi

Di lapisan atas adalah aplikasi yang dimuat sebelumnya yang disediakan oleh vendor perangkat dan pihak ketiga lainnya (operator seluler). Untuk memungkinkan pengembang aplikasi mengakses berbagai sumber daya dan fungsionalitas, lapisan Android Framework menyediakan banyak layanan tingkat tinggi. Layanan ini memediasi akses ke sumber daya sistem dan menegakkan kontrol akses yang tepat berdasarkan id pengguna aplikasi dan izin Android yang diperolehnya. Selain itu, layanan tertentu mungkin memberlakukan kontrol akses berdasarkan nama paket atau sertifikat pemanggil. Tepat di bawah lapisan kerangka kerja terdapat lapisan Perpustakaan, yang merupakan sekumpulan pustaka khusus Android dan pustaka lain yang diperlukan seperti libc, basis data SQLite, pustaka media, dll. Sama seperti layanan kerangka kerja, pustaka khusus Android tertentu melakukan berbagai pemeriksaan kontrol akses berdasarkan pada id pengguna penelepon dan izinnya juga. Di bagian bawah lapisan adalah kernel Linux yang menyediakan tingkat abstraksi antara perangkat keras perangkat dan berisi semua driver perangkat keras penting seperti tampilan, kamera, dll. Lapisan kernel Linux memediasi akses ke driver perangkat keras dan sumber daya mentah berdasarkan Akses Diskresioner standar Kontrol (DAC). Untuk mendorong kolaborasi, aplikasi Android dihubungkan bersama oleh Inter-Component Communication (ICC). Aplikasi bisa memanggil komponen aplikasi lain (misalnya aktivitas dan layanan) melalui mekanisme maksud. Selanjutnya dapat mengonfigurasi beberapa parameter keamanan untuk melindungi sumber daya dan fungsionalitasnya.

Izin

Izin Android default dan kustom digunakan untuk melindungi komponen dalam, data, dan fungsionalitas. Tingkat perlindungan izin harus dipilih dengan hati-hati tergantung pada sumber daya yang akan dilindungi. Izin level Signature dan SystemOrSignature digunakan untuk melindungi sumber daya yang paling diistimewakan dan hanya akan diberikan ke aplikasi yang ditandatangani dengan sertifikat yang sama dengan aplikasi yang menentukan. Izin berbahaya melindungi data pribadi dan sumber daya atau operasi yang memengaruhi data yang disimpan pengguna atau aplikasi lain seperti membaca kontak atau mengirim pesan SMS. Meminta izin level Berbahaya memerlukan konfirmasi pengguna secara eksplisit sebelum memberikannya. Level normal di sisi lain, diberikan untuk izin yang melindungi sumber daya yang paling tidak diistimewakan dan tidak memerlukan persetujuan pengguna. Nilai yang tidak ditentukan mengacu pada izin yang telah ditentukan tanpa tingkat perlindungan, sedangkan 0 mengacu pada izin yang tidak ditentukan pada gambar.

p-ISSN : 2808-909X e-ISSN : 2798-2505

GID

Grup ID Linux (GID) tingkat rendah tertentu dipetakan ke izin Android. Setelah proses aplikasi memperoleh izin maka GID dipetakan dan digunakan untuk kontrol akses di kernel. Android menyatakan bahwa setiap perubahan yang dilakukan secara tidak hati-hati pada platform.xml akan memberikan kerentanan serius. Untuk memungkinkan penemuan GID yang rentan terhadap pemetaan izin, Persyaratan izin minimum yang diperlukan untuk memperoleh GID tertentu dipetakan pada gambar yang diberikan. Jika GID yang sama memiliki perbedaan persyaratan minimum pada 2 gambar, ini berpotensi rentan.

Broadcast yang Dilindungi

Broadcast yang dilindungi adalah broadcast yang hanya dapat dikirim oleh proses tingkat sistem. Aplikasi menggunakan broadcast yang dilindungi untuk memastikan bahwa tidak ada proses tingkat sistem dapat memicu penerima broadcast tertentu. Aplikasi sistem dapat menentukan broadcast yang dilindungi sebagai berikut:

<protected-broadcast android:name="broadcast.name"/>

Aplikasi lain dapat menggunakan broadcast terproteksi yang ditentukan di atas melalui berikut ini:

<receiver android:name="ReceiverA">

<intent-filter>

<action = "broadcast.name"/> <intent-filter/>

<receiver/>

Receiver di atas hanya dapat dipicu oleh proses sistem yang menyiarkan broadcast.name protected broadcast. Aplikasi ini juga dapat menggunakan broadcast yang dilindungi melalui penerima broadcast yang terdaftar secara dinamis. Selama proses kustomisasi, paket-paket tertentu dihapus dan diubah. Definisi broadcast terproteksi tertentu akan dihapus juga. Tujuannya adalah untuk mengungkap apakah broadcast yang tidak dilindungi secara tidak konsisten ini masih digunakan, sebagai filter tindakan di dalam penerima. Ini mungkin membuka kerentanan serius, karena penerima yang dianggap pengembang hanya dapat dipanggil oleh proses sistem sekarang akan dapat dipanggil oleh aplikasi pihak ketiga mana pun dan akibatnya mengekspos fungsinya. Secara formal, untuk setiap Broadcast Terproteksi $e \in E_{PB}$ didefinisikan sebagai:

$$fn_e = DefineUse(e),$$

Di mana,

$$DefineUse(e) = \begin{cases} 1 \; jika \; e \; digunakan \; pada \; gambar \; tapi \; tidak \; didefinisikan \\ 0 \; jika \; kasus \; lain \end{cases}$$

Visibilitas Komponen

Android mengizinkan pengembang untuk menentukan apakah komponen yang dideklarasikan dapat dipanggil secara eksternal dari aplikasi lain. Visibilitas dapat diatur melalui flag yang diekspor dalam deklarasi komponen di dalam file manifes aplikasi. Jika flag ini tidak ditentukan, visibilitas akan diatur secara implisit berdasarkan apakah komponen penentu filter, jika ada maka komponen akan diekspordan jika tidak maka akan muncul:

// Service1 is private to the app

<service android:name="Service1"/> // Service2 is not private to the app <service android:name="Service2">

<intent-filter> ... <intent-filter/> </service>

Perlindungan Komponen

Aplikasi dapat menggunakan izin untuk membatasi pemanggilan komponennya. Dalam cuplikan kode berikutnya, ServiceA dapat dipanggil jika pemanggil mendapatkan vendor.permissionA. Selain itu, aplikasi dapat menggunakan izin untuk membatasi membaca dan menulis ke penyedia kontennya, serta ke jalur tertentu di dalamnya.

p-ISSN : 2808-909X e-ISSN : 2798-2505

android:readPermission dan android:writePermission diutamakan daripada android:permission jika ditentukan, seperti yang ditampilkan dalam cuplikan kode. Komponen mewarisi izin orang tuanya jika mereka tidak menentukannya.

<service android:name="ServiceA" android:permission="vendor.permissionA"/> <provider android:authorities="providerId" android:name="providerB"

android:Permission="vendor.permissionB"

android:readPermission="vendor.read" android:writePermission="vendor.write">

Ketidaksesuaian perlindungan mungkin tidak selalu menunjukkan cacat jika komponen tidak terbuka. Itu sebabnya, jika ada komponen yang diekspor ketidakcocokan perlindungan akan dipertimbangkan.

Pembuatan Data

Untuk mengungkap apakah pihak kustomisasi mengubah konfigurasi fitur keamanan yang disebutkan, analisis diferensial skala besar dilakukan. sejumlah 591 ROM Android dikumpulkan dariPembaruan Samsung dan perangkat fisik. Gambar-gambar ini disesuaikan oleh 11 vendor, untuk sekitar 135 model, dan 8 operator. Mereka mengoperasikan versi Android dari 4.1.1 hingga 5.1.1. Secara total, gambar ini mencakup rata-rata 157 aplikasi per gambar dan 93169 semua aplikasi secara bersamaan. Untuk mengekstrak nilai fitur keamanan yang dipilih pada setiap gambar, alat bernama DroidDiff dikembangkan dalam penelitian ini. Untuk setiap gambar, pertama-tama DroidDiff mengumpulkan Apks sumber daya kerangka kerjanya dan Apks yang dimuat sebelumnya, kemudian menjalankan Apktool untuk mengekstrak file manifes yang sesuai. Kedua, ia mengumpulkan file konfigurasi di bawah /etc/permission/. Kemudian, DroidDiff mencari manifes yang diekstraksi dan file konfigurasi untuk definisi entitas yang ditargetkan (EP , EPB, EGID dan EC). Terakhir, DroidDiff menjalankan nilai yang dihasilkan melalui metodologi analisis diferensial.

Analisis Diferensial

Setiap ketidakkonsistenan yang terdeteksi menunjukkan kemungkinan perubahan konfigurasi yang tidak disengaja yang diperkenalkan oleh pihak kustomisasi dan memerlukan analisis keamanan lebih lanjut untuk menilai kemungkinan kerusakan yang diakibatkannya. Misalkan fv(fne, img) mewakili nilai fitur fne pada gambar img yang diberikan. Untuk mengilustrasikan pemetaan fn_e ke fv(fn_e, img), fitur keamanan diekstrak dan nilai yang sesuai hanya dari 2 gambar Xiaomi. Untuk izin khusus e = MIPUSH_RECEIVE, langkah myfeatureextraction menghasilkan nilai berikut fv(fn_e,I1) = Signature, dan fv(fn_e,I2) = Unspecified. Diketahui IMG menunjukkan satu set gambar kandidat untuk dibandingkan, fitur fne didefinisikan sebagai ketidakkonsistenan jika:

$$C(fn_e)=\exists x\ \exists y\ [\ x \in IMG \wedge y \in IMG$$

$$\wedge x \neq y \wedge fv\ (fn_e, x)\ 6=\ fv\ (fn_e, y)]$$

Pernyataan di atas berarti bahwa fitur fne dianggap tidak konsisten di seluruh set IMG jika ada setidaknya dua gambar berbeda di mana nilai fne tidak sama.

Pemilihan Sampel. Untuk menemukan ketidakkonsistenan yang bermakna melalui analisis diferensial, gambar yang dikumpulkan harus dikelompokkan berdasarkan kriteria umum. Inkonsistensi yang berarti akan memberi wawasan tentang pihak yang bertanggung jawab yang memperkenalkannya. Untuk mengungkap jika vendor tertentu menyebabkan ketidakkonsistenan dalam model baru, tidak logis untuk membandingkannya dengan model dari vendor lain. Untuk menghindari pendeteksian perubahan yang disebabkan oleh ketidaksesuaian versi OS, model baru harus dibandingkan dengan model yang menjalankan versi OS yang sama. Lima algoritme yang berbeda dirancang untuk mengungkap ketidakkonsistenan yang bermakna. Secara khusus, dengan hati-hati menelusuri setiap pihak dalam rantai penyesuaian, algoritme yang akan mengungkap ketidakkonsistenan yang disebabkan oleh masing-masing pihak juga dirancang. Selanjutnya, untuk setiap algoritme, gambar kandidat dipilih berdasarkan kriteria spesifik yang sesuai dengan tujuan algoritme.

A1: Analisis Lintas Versi. Analisis ini bertujuan untuk mengungkap fitur keamanan yang tidak konsisten yang disebabkan oleh peningkatan versi OS. Kumpulan gambar kandidat yang menjalankan model perangkat serupa diilih untuk memastikan bahwa ketidakkonsistenan murni karena peningkatan OS. Misalnya, 2 perangkat Samsung S4 yang menjalankan 4.4.4 dan 5.0.1 sebagai kumpulan gambar kandidat,

p-ISSN : 2808-909X e-ISSN : 2798-2505

akan dipilih dan akan mengungkapkan jika memutakhirkan model ini dari 4.4.4 ke 5.0.1 menyebabkan perubahan konfigurasi keamanan. Secara formal, IMG_{MODEL} menunjukkan kumpulan gambar kandidat sebagai berikut:

$$IMG_{MODEL} = \{img_1, img_2, ..., img_n\}$$

Berdasarkan gambar yang dikumpulkan, algoritme ini menghasilkan 135 kumpulan gambar kandidat (hitungan model berbeda). Diketahui $f_v(f_{ne}, img)$ menunjukkan nilai untuk fitur fn_e di $img \in IMG_{MODEL}$. Tentukan kondisi inkonsistensi di bawah algoritma analisis Cross-Version sebagai berikut,

$$C_{version}(fn_e) = EX\ EY\ [X \in I\ MG_{MODEL} \wedge Y \in I\ MG_{MODEL}$$

$$\wedge x \neq y \wedge fv\ (fne, x) \neq fv\ (fn_e, y)$$

$$\wedge version(x) \neq version(y)]$$

Kondisi di atas menyiratkan bahwa fn_e tidak konsisten jika ada dua gambar dari model yang sama yang menjalankan versi yang berbeda, dan nilai fne tidak sama. DroidDiff menjalankan analisis untuk masing-masing dari 135 set kandidat dan menghasilkan jumlah ketidakkonsistenan yang terdeteksi. ***A2: Analisis Lintas Vendor***. Analisis ini bertujuan untuk mengungkap fitur fn_e yang tidak konsisten di seluruh vendor. Untuk memastikan bahwa perbandingan gambar dengan kriteria serupa di berbagai vendor, set gambar kandidat yang menjalankan versi OS yang sama dipilih. Jika ditemukan ketidakkonsistenan, maka vendor adalah pihak yang bertanggung jawab. Kumpulan gambar kandidat sebagai:

$$IMG_{VERSION} = \{img_1, img_2, ..., img_n\}$$

Algoritme ini menghasilkan 12 kumpulan gambar kandidat. ***A3: Analisis Lintas Model***. Untuk memastikan bahwa ketidakkonsistenan apa pun murni karena perubahan model dalam vendor yang sama, rangkaian gambar kandidat yang menjalankan versi OS yang sama dipilih. Diketahui fv(fn_e, img) menunjukkan nilai untuk fne di img E IMGvERSION· Kondisi ketidakkonsistenan di bawah analisis Cross-Model didefinisikan kembalai sebagai:

$$C_{Model}(fn_e) = Ex\ Ey\ [\ x\ E\ IMG_{VERSION} \wedge y \in IMG_{VERSION}$$

$$\wedge x \neq y \wedge fv\ (fn_e, x) \neq fv\ (fn_e, y)$$

$$\wedge vendor(x) \neq vendor(y) \wedge model(y) \neq model(x)$$

Kondisi terakhir menyiratkan bahwa fn_e tidak konsisten jika ada dua image dari vendor yang sama, menjalankan versi OS yang sama, tetapi dikustomisasi untuk model yang berbeda, di mana nilainya tidak sama. ***A4: Analisis Lintas Operator***. Untuk memastikan perbandingan gambar yang menjalankan versi OS yang sama, kumpulan gambar kandidat dari IMGvERSION dipilih. Selanjutnya gambar yang menjalankan model yang sama dibandingkan dengan kondisi ketidakkonsistenan berikut:

$$C_{Carier}(fn_e) = Ex\ Ey\ [\ x\ E\ IMG_{VERSION} \wedge y \in IMG_{VERSION}$$

$$\wedge x \neq y \wedge fv\ (fn_e, x) \neq fv\ (fn_e, y)$$

$$\wedge carrier(x) \neq carrier(y) \wedge model(y) \neq model(x)$$

Kondisi terakhir dalam definisi Ccarrier di atas menyiratkan bahwa fne tidak konsisten jika ada dua image yang menjalankan model dan versi OS yang sama, tetapi dari operator yang berbeda di mana nilainya tidak sama.

HASIL DAN TEMUAN

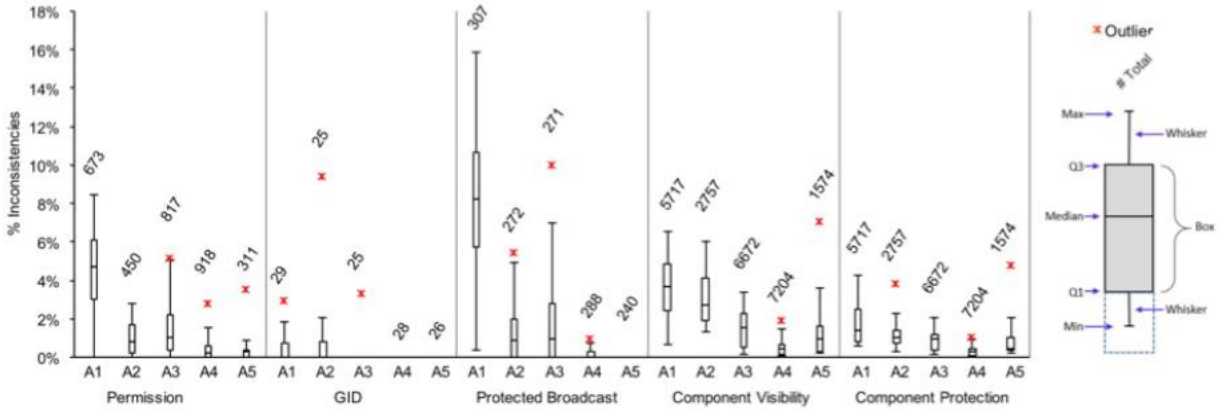
Penelitian ini menjalankan DroidDiff untuk melakukan analisis diferensial skala besar pada gambar yang di kumpulkan menggunakan metodologi yang disebutkan di atas. DroidDiff menemukan banyak perbedaan terkait dengan fitur pilihan.

Hasil keseluruhan

Gambar 2 menunjukkan keseluruhan perubahan yang terdeteksi dari analisis penelitian ini. Persentase rata-rata ketidakkonsistenan yang terdeteksi diplot untuk setiap kategori fitur (Izin, GID, Broadcast Terproteksi, Visibilitas Komponen, dan Perlindungan Komponen) menggunakan lima algoritme analisis diferensial. Untuk memberikan perkiraan jumlah ketidakkonsistenan, setiap petak kotak menunjukkan jumlah rata-rata entitas umum total (muncul pada setidaknya 2 gambar) dalam kumpulan

p-ISSN : 2808-909X e-ISSN : 2798-2505

gambar yang dipelajari. Dengan petak kotak pertama sebagai contoh untuk mengilustrasikan arti data: di bawah analisis Cross-Version (A1), DroidDiff menghasilkan rata-rata 673 izin umum per setiap set kandidat yang dipelajari. 50% dari kumpulan gambar kandidat berisi setidaknya 4,8% dari total izin (sekitar 32 dari 673) yang memiliki tingkat perlindungan yang tidak konsisten; mereka yang berada di persentil 25 teratas (ditampilkan di kumis teratas) memiliki setidaknya 6% (40) izin yang tidak konsisten. Gambar 2 juga menggambarkan kumpulan gambar yang outlier, yaitu, mereka memiliki jumlah inkonsistensi yang lebih tinggi dibandingkan dengan kumpulan gambar lain dalam kelompok yang sama. Misalnya, kumpulan gambar kandidat IMGVersion=4.4.2 dalam analisis Lintas-Vendor (A2) berisi sekitar 10% GID yang perlindungannya tidak konsisten.



Gambar 2. Keseluruhan Inkonsistensi Terdeteksi. Al: Lintas Versi, A2: Lintas Vendor, A3: Lintas Model, A4: Lintas Operator, AS: Lintas Wilayah

Seperti yang digambarkan dalam Gambar 2, analisis Cross-Version (A1) mendeteksi persentase ketidakkonsistenan tertinggi di semua 5 kategori, yang berarti bahwa memutakhirkan model perangkat yang sama ke versi OS yang berbeda memperkenalkan perubahan konfigurasi keamanan tertinggi. Alasan intuitif di balik ini adalah bahwa melalui rilis OS baru, Android mungkin menerapkan perlindungan yang lebih tinggi pada entitas terkait untuk memperbaiki beberapa bug yang ditemukan (misalnya menambahkan persyaratan izin ke layanan istimewa). Namun, melalui rilis OS yang lebih baru, ditemukan fitur keamanan tertentu sebenarnya diturunkan versinya, yang menyebabkan potensi risiko jika dilakukan secara tidak sengaja. Melalui analisis Cross-Vendor (A2), DroidDiff mendeteksi bahwa beberapa fitur keamanan tidak konsisten di antara vendor, meskipun mereka memiliki versi OS yang sama. Analisis lebih lanjut dilakukan pada vendor yang menyebabkan jumlah ketidakkonsistenan tertinggi. Pengamatan yang menarik adalah vendor yang lebih kecil, seperti BLU, Xiaomi dan Digiland menyebabkan beberapa inkonsistensi yang berisiko. Semua GID yang tidak konsisten (berpotensi sangat parah) sebenarnya disebabkan oleh 3 perusahaan ini. Mungkin, vendor kecil mungkin tidak memiliki keahlian yang cukup untuk sepenuhnya mengevaluasi implikasi keamanan dari tindakan mereka. Analisis Cross-Model (A3) juga mendeteksi sejumlah ketidakkonsistenan, yang berarti bahwa model perangkat yang berbeda dari vendor dan versi OS yang sama, mungkin memiliki konfigurasi keamanan yang berbeda. Meskipun analisis Cross-Carrier (A4) dan Cross-Region (A5) mendeteksi persentase ketidakkonsistenan yang lebih kecil, masih signifikan untuk mengetahui bahwa model perangkat yang sama yang menjalankan versi OS yang sama mungkin memiliki beberapa konfigurasi yang berbeda jika disesuaikan untuk perangkat yang berbeda. operator atau daerah.

Izin Mengubah Pola

Ketidakcocokan tingkat perlindungan. Hasil analisis diferensial DroidDiff memastikan bahwa izin Android mungkin memiliki tingkat perlindungan yang berbeda pada gambar yang serupa. Seperti yang diilustrasikan oleh Gambar 2, lebih dari 50% kumpulan kandidat gambar berisi setidaknya 32 (dari 673), 9 (dari 817) izin yang memiliki tingkat perlindungan yang tidak konsisten di Cross-Version (A1) dan Cross-Model (A3) analisis, masing-masing. Untuk mengungkapkan lebih banyak wawasan, kombinasi perubahan tingkat perlindungan yang paling umum juga diperiksa, kombinasi mana dari 3 kemungkinan kombinasi berikut yang paling umum (Normal, Berbahaya), (Normal, Signature) atau (Berbahaya, Signature). Kemunculan setiap pola juga dihitung, dan hasilnya disajikan pada Gambar 3. Kombinasi (Normal, Signature) adalah pola yang paling umum. Ini cukup serius karena beberapa izin yang memiliki tingkat

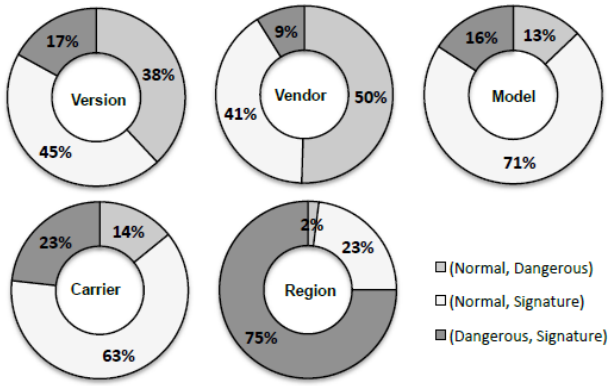
p-ISSN : 2808-909X e-ISSN : 2798-2505

perlindungan Signature pada beberapa gambar ditetapkan dengan tingkat perlindungan Normal pada gambar lainnya. Di sini disajikan dua izin yang memiliki tingkat perlindungan yang tidak konsisten:

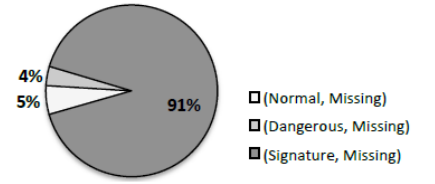
- com.orange.permission.SIMCARD_AUTHENTICATION holds Signature and Normal protection on Samsung S4(4.2.2) and Sony Experia C2105 (5.0.1), respectively.
- com.sec.android.app.sysscope.permission.RUN_ SYSSCOPE holds Dangerous and Signature protection on Samsung Note4 and S4(5.0.1), respectively.

Penggunaan tingkat perlindungan yang tidak ditentukan. Android mengizinkan pengembang untuk menentukan izin tanpa menetapkan tingkat perlindungan, dalam hal ini, tingkat perlindungan default adalah Normal. Dalam penelitian ini ditemukan bahwa tidak jelas apakah pengembang benar-benar bermaksud menggunakan Normal sebagai tingkat perlindungan. Pada gambaran lain ditmukan bahwa sebagian besar izin dengan tingkat perlindungan yang tidak ditentukan memiliki perlindungan konflik. Secara keseluruhan, 2% dari izin yang dipelajari ditentukan tanpa tingkat perlindungan yang ditentukan dalam setidaknya satu gambar. Untuk memeriksa apakah pengembang bermaksud menggunakan Normal sebagai tingkat perlindungan, untuk setiap izin yang telah ditentukan tanpa tingkat perlindungan, definisi yang sesuai pada gambar lain juga diperiksa untuk melihat apakah tingkat perlindungan telah ditentukan. Selanjutnya pesifikasi lainnya juga dibandingkan untuk melihat normal atau tidak. Seperti yang diilustrasikan oleh Gambar 4(a), rata-rata, 91% dari izin ini yang memiliki tingkat perlindungan yang tidak ditentukan memiliki perlindungan Signature pada setidaknya 1 gambar lainnya, yang menunjukkan bahwa pengembang mungkin bermaksud menggunakan tingkat perlindungan Signature. Temuan ini diilustrasikan menggunakan 2 izin:

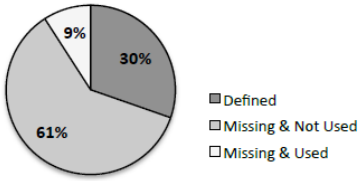
- com.sec.android.phone.permission.UPDATE_ MUTE_STATUS holds Unspecified and Signature protections on Samsung E7 (5.1.1) and S6 Edge(5.1.1), respectively.
- com.android.chrome.PRERENDER_URL holds Unspecified and Signature protections on LG Vista (4.4.2) and Nexus7 (4.4.2), respectively.



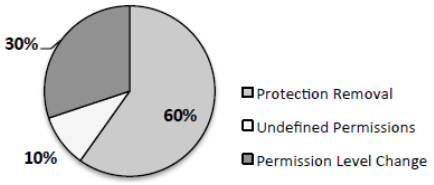
Gambar 3 Pola Perubahan Tingkat Perlindungan



(a) Unspecified Protection Level Change Patterns



(b) Protected Broadcast Inconsistencies



(c) Causes of Components Protection Mismatch

p-ISSN : 2808-909X e-ISSN : 2798-2505

Gambar 4 Perincian Inkonsistensi

Pemetaan Izin-GID

Dengan menganalisis hasil analisis diferensial dari pemetaan antara GID dan izin, dapat dipastikan bahwa penyesuaian menimbulkan masalah Pemetaan GID-ke-izin yang dapat menyebabkan kerentanan serius pada gambar korban. Melalui analisis Lintas-Vendor (A2), DroidDiff mendeteksi 3 kasus yang tidak konsisten (dari 25 GID umum), di mana vendor memetakan izin kurang istimewa ke GID istimewa. Pola berbahaya ini mengarah pada penurunan tingkat perlindungan GID ini. Pada gambar AOSP dan beberapa gambar yang disesuaikan (menjalankan 4.4.4 dan yang lebih lama), GID kamera dipetakan ke izin level Berbahaya (android.permission.CAMERA). Namun, pada Neo 4.5 (BLU), GID yang sama dipetakan ke izin tingkat Normal: android.permission.ACCESS_MTK_MMHW. Kasus ini menunjukkan bahwa BLU telah menurunkan persyaratan aplikasi untuk mendapatkan GID kamera. Analisis penelitian ini menunjukkan bahwa persyaratan untuk dua GID lainnya, GID sistem dan GID media, telah diturunkan. Kedua GID ini, dilindungi oleh izin Signature di sebagian besar perangkat, dapat diperoleh dengan izin Normal di perangkat korban.

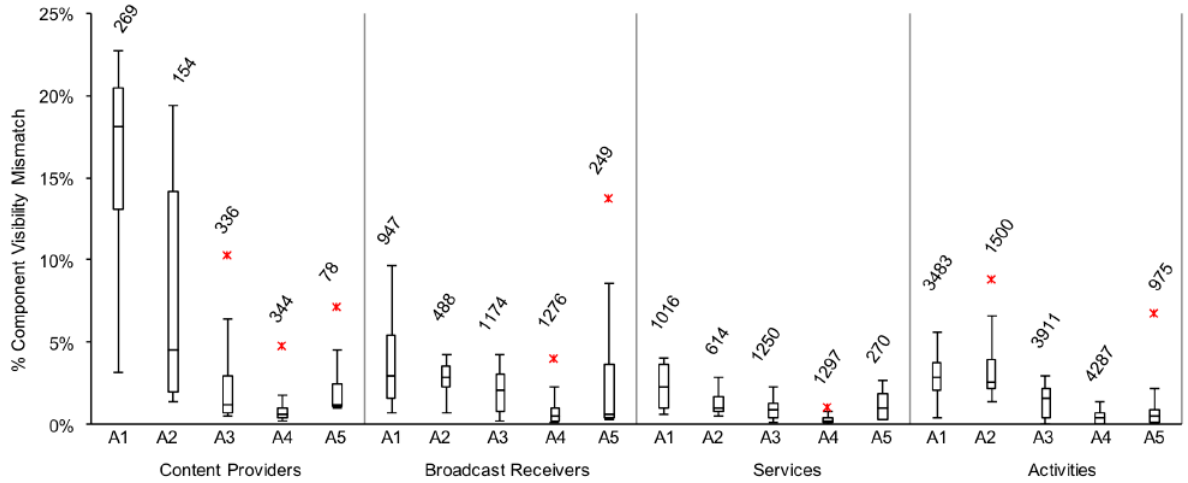
Broadcast yang Dilindungi Mengubah Pola

DroidDiff selanjutnya mengungkapkan bahwa definisi broadcast yang dilindungi mungkin dihapus dari beberapa gambar selama proses penyesuaian. Seperti yang diilustrasikan pada Gambar 5.5(b), melalui analisis Cross-Version (A1), ditemukan bahwa 70% broadcast yang dilindungi tidak ditentukan pada setidaknya satu vendor. Ini belum tentu bermasalah jika broadcast tidak digunakan. Namun, penelitian ini menunjukkan bahwa sekitar 9% dari broadcast yang tidak dilindungi secara tidak konsisten ini (rata-rata 28 per kumpulan gambar) digunakan sebagai tindakan filter maksud untuk penerima broadcast. Ketidakkonsistenan antar versi ini cukup mengkhawatirkan karena penerima dengan hak istimewa yang seharusnya dipanggil oleh proses sistem dapat dipanggil oleh aplikasi apa pun yang tidak memiliki hak istimewa pada versi tertentu. Seperti yang diilustrasikan lebih lanjut oleh Gambar 5.3, analisis Cross-Vendor (A2) dan Cross-Model (A3) mengungkapkan bahwa lebih dari 25% kumpulan gambar kandidat berisi setidaknya 2% broadcast yang dilindungi secara tidak konsisten, tetapi masih digunakan sebagai tindakan filter maksud.

Pola Perubahan Keamanan Komponen

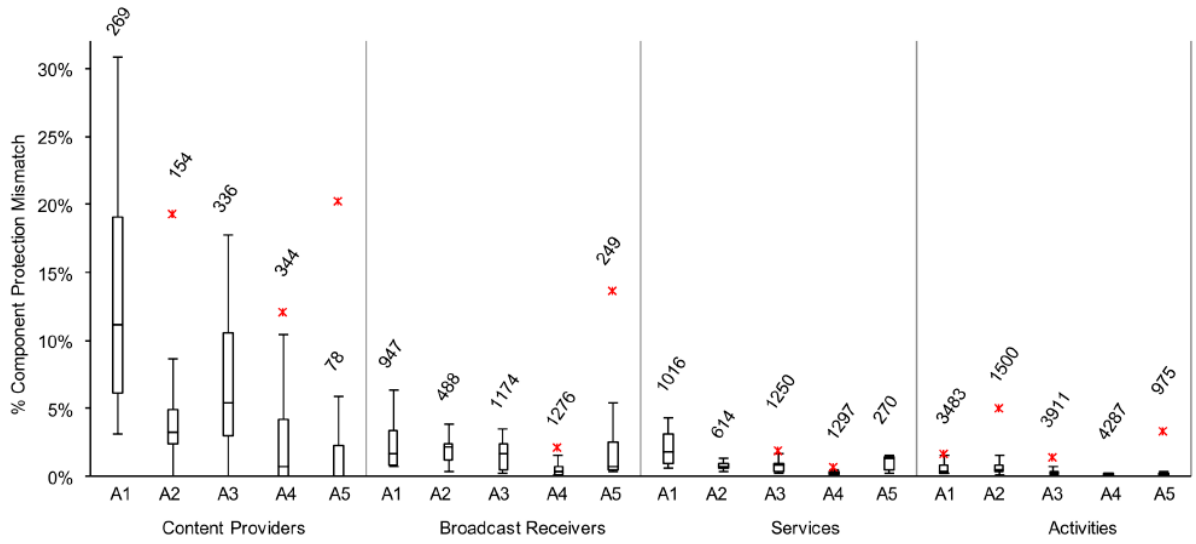
Ketidakcocokan visibilitas. Hasil DroidDiff mengonfirmasi bahwa komponen aplikasi mungkin memiliki visibilitas yang bertentangan. Artinya, komponen diekspos pada satu gambar tetapi tidak pada gambar lainnya. Seperti yang diilustrasikan oleh Gambar 2, 50% dari kumpulan kandidat gambar mengandung setidaknya 3,9% komponen (sekitar 222) dan 2% (133) memegang visibilitas yang tidak konsisten melalui berbagai versi (A1) dan model (A3). Untuk memberikan wawasan tentang komponen mana yang memiliki lebih banyak ketidakkonsistenan visibilitas, temuan ini dibagi menjadi aktivitas, layanan, penerima, dan penyedia konten. Hasilnya diplot pada Gambar 5. Seperti yang digambarkan, penyedia konten dan aktivitas memiliki ketidakcocokan visibilitas tertinggi. 25% dari kumpulan gambar kandidat berisi setidaknya 20% (53) dan 14% (21) penyedia konten yang masing-masing memiliki visibilitas berbeda dalam versi yang berbeda (A1) dan vendor (A2). Demikian pula, 4% (139) dan 3% (45) aktivitas memiliki visibilitas yang bertentangan di 50% dari set yang dipelajari berdasarkan A1 dan A2, masing-masing.

p-ISSN : 2808-909X e-ISSN : 2798-2505



Gambar 5 Kerusakan Komponen: Visibility Mismatch

Ketidakcocokan izin. DroidDiff mengungkapkan bahwa komponen mungkin memiliki perlindungan yang tidak konsisten di seluruh gambar. Hasil menunjukkan bahwa penyedia konten menunjukkan jumlah ketidakkonsistenan perlindungan tertinggi. Faktanya, lebih dari 25% kumpulan gambar kandidat mencakup setidaknya 19%(51) dan 10% (33) penyedia konten yang memiliki perlindungan berbeda dalam analisis Cross-Version (A1) dan Cross-Model (A3). Analisis lebih lanjut dilakukan pada komponen-komponen yang tidak konsisten. Seperti yang diilustrasikan oleh Gambar 4(c), pada sebagian besar kasus (60%), ketidaksesuaian disebabkan oleh komponen yang sama dilindungi dengan izin pada satu gambar, tetapi tidak dilindungi sama sekali pada gambar lainnya. Alasan umum kedua (30%) adalah komponen yang sama dilindungi dengan izin yang memiliki tingkat perlindungan berbeda di seluruh gambar yang dipelajari. Menggunakan izin yang tidak ditentukan untuk melindungi komponen adalah alasan umum ketiga (10%).



Gambar 6 Kerusakan Komponen: Ketidaksesuaian Perlindungan Izin

Deklarasi komponen duplikat. Berdasarkan analisis penerima broadcast yang tidak konsisten, ditemukan bahwa sebagian besar disebabkan oleh praktik tidak aman yang diikuti pengembang. Pengembang mendeklarasikan nama penerima broadcast duplikat di aplikasi yang sama, tetapi menetapkan perlindungan yang berbeda. Setelah penyelidikan lebih lanjut, ditemukan bahwa ini bukanlah praktik yang aman untuk dilakukan karena akan memungkinkan untuk memintas batasan apa pun yang diberikan pada

p-ISSN : 2808-909X e-ISSN : 2798-2505

receiver pertama yang ditetapkan. Sebagai ilustrasi, ini adalah receiver yang ditentukan dalam aplikasi PhoneErrorService bawaan Samsung:

<receiver android:name="PhoneErrorReceiver"

android:permission="android.permission.REBOOT">

<intent-filter>

<action android:name="REFRESH_RESET_FAIL"/>

...

</intent-filter>

</receiver>

<receiver android:name="PhoneErrorReceiver">

<intent-filter>

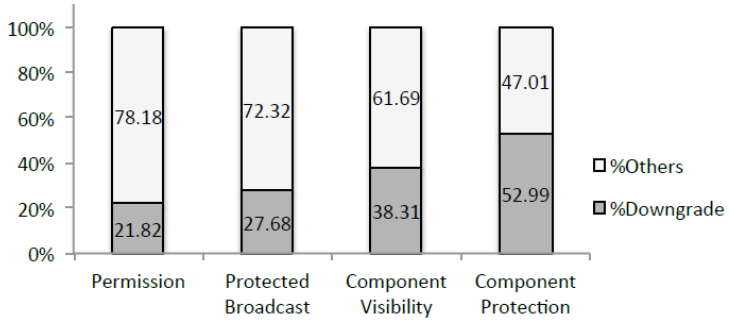
<action android:name="DATA_ROUTER_DISPLAY"/> </intent-filter>

</receiver>

Dalam kode di atas, pengembang memutuskan untuk melindungi fungsionalitas yang dipicu saat menerima tindakan REFRESH_RESET_FAIL dengan izin REBOOT (Level Signature). Dalam kasus lain, dia memutuskan untuk tidak memerlukan izin apa pun saat menjalankan fungsionalitas yang dipicu oleh tindakan DATA_ROUTER_DISPLAY. Sepintas, deklarasi komponen duplikat di atas mungkin terlihat baik-baik saja. Namun, PackageManagerService ditemukan tidak hati-hati dalam menangani pendaftaran penerima duplikat. Di satu sisi, ini menangani pemetaan setiap filter ke izin yang diperlukan dengan benar, disisi lain, memetakan setiap nama komponen ke izin yang diperlukan untuk perutean maksud eksplisit bukanlah hal benar. Dengan demikian, persyaratan perlindungan apa pun pada penerima kedua akan menggantikan persyaratan izin penerima pertama jika terjadi pemanggilan eksplisit. Akibatnya, PhoneErrorReceiver secara eksplisit tidak memerlukan izin apa pun. Maksud eksplisit selanjutnya dapat mengatur tindakan REFRESH_RESET_FAIL dan dengan demikian memicu fungsionalitas istimewa (me-reboot ponsel) tanpa izin REBOOT yang diperlukan.

Downgrade Melalui Analisis Versi

Seperti yang diilustrasikan oleh Gambar 5.8, sejumlah besar konfigurasi benar-benar diturunkan. Misalnya, 52% ketidaksesuaian perlindungan komponen yang tidak konsisten sebenarnya disebabkan oleh penurunan versi perlindungan.



Gambar 7 Persentase Penurunan Versi Fitur Keamanan

Serangan

Ada beberapa serangan aktual yang diantaranya dikonfirmasikan pada beberapa perangkat.

1. **Mencuri email**. SecEmailSync.apk adalah aplikasi bawaan di sebagian besar perangkat Samsung. Ini termasuk penyedia konten, yang disebut "com.samsung.android.email.otherprovider", yang menyimpan salinan email pengguna yang diterima melalui aplikasi email default Samsung. Analisis Lintas Model mengungkapkan perlindungan izin yang tidak konsisten pada penyedia ini di antara

p-ISSN : 2808-909X e-ISSN : 2798-2505

beberapa citra Samsung. Akses Baca dan Tulis ke penyedia ini dilindungi dengan izin Signature "com.samsung.android.email.permission.

2. **Memalsukan pesan SMS premium**. Paket TeleService (com.android.phone) sudah dimuat sebelumnya di banyak perangkat Samsung, dan menyediakan beberapa layanan untuk pengelolaan telepon dan panggilan. Layanan penting adalah .TPhoneService, yang melakukan beberapa fungsi telepon utama seperti menerima panggilan suara dan video, menghubungi nomor telepon baru, mengirim pesan (misalnya untuk menginformasikan mengapa panggilan tidak dapat diterima), serta merekam panggilan suara dan video. Analisis Cross-Model dan Cross-Version mengungkapkan ketidaksesuaian izin pada layanan penting ini. Pada beberapa perangkat, seperti Samsung S5 LTE-A (4.4.2, Korea), akses ke layanan ini dilindungi dengan izin Signature com.skt.prod. permission.OEM_PHONE_SERVICE, yang membuat layanan tidak dapat diakses oleh aplikasi pihak ketiga.
3. **Reset pabrik tidak sah**. Aplikasi Samsung ServiceModeApp_FB.apk yang dimuat sebelumnya menjalankan berbagai fungsi yang terkait dengan pengaturan ponsel yang sensitif. Ini termasuk ServiceModeAppBroadcastReceiver penerima broadcast yang mendengarkan beberapa filter maksud termasuk filter tindakan com.samsung.intent.action.SEC_FACTORY_RESET_ WITHOUT_FACTORY_UI yang memungkinkan untuk mengatur ulang ponsel dan menghapus semua data tanpa konfirmasi pengguna. Analisis Cross-Version mengungkapkan ketidakcocokan perlindungan untuk penerima broadcast penting ini. Di sebagian besar perangkat yang menjalankan Kitkat dan yang lebih lama, penerima ini dilindungi dengan izin Signature com.sec.android.app.servicemodeapp.permission.KEYSTRING. Namun, pada beberapa gambar Lollipop, tidak dilindungi dengan benar.
4. **Mengakses driver penting dengan izin normal**. Analisis Lintas-Vendor mengungkap penurunan versi perlindungan kritis dari GID sistem. Pada beberapa gambar, seperti Samsung S5 (4.4.2), GID ini dipetakan ke izin Signature com.qualcomm. izin.IZAT.Namun demikian, pada gambar lain, GID ini dipetakan ke tingkat normal izin android.permission. ACCESS_MTK_MMHW, menunjukkan bahwa aplikasi pihak ketiga mana pun dapat dengan mudah mendapatkan GID sistem.
5. **Memicu broadcast darurat tanpa izin**. CellBroadcastReceiver adalah aplikasi Google bawaan yang menjalankan fungsi penting berdasarkan broadcast seluler yang diterima. Ini mendaftarkan penerima broadcast PrivilegedCellBroadcastReceiver yang memungkinkan menerima broadcast darurat dari penyedia sel (misalnya peringatan evakuasi, peringatan kepresidenan, peringatan kuning, dll.) dan menampilkan peringatan yang sesuai. Fungsi penting ini dapat dipicu jika tindakan android.provider.Telephony.SMS_EMERGENCY_CB_RECEIVED diterima. Analisis Lintas-Vendor dan Lintas-Versi menemukan ketidakcocokan perlindungan pada receiver ini di antara beberapa perangkat namun dilindungi dengan izin Berbahaya android.permission. READ_PHONE_STATE. Investigasi mengungkapkan bahwa ini juga disebabkan oleh pola penerima duplikat yang berisiko. Pada perangkat korban, PrivilegedCellBroadcastReceiver telah dideklarasikan dua kali sehingga deklarasi pertamanya memerlukan izin Signature dan menangani tindakan android.provider.Telephony.SMS_EMERGENCY_CB _RECEIVED, sedangkan deklarasi kedua menangani tindakan yang kurang istimewa dan memerlukan izin Berbahaya. Aplikasi pihak ketiga mana pun dapat mengabaikan persyaratan izin pada penerima pertama melalui pemanggilan eksplisit.
6. **Merusak pengaturan seluruh sistem.** SystemUI adalah aplikasi bawaan yang mengontrol jendela sistem. Ini menangani dan menggambar banyak UI sistem seperti bilah status teratas, pemberitahuan sistem, dan dialog. Untuk mengelola bilah status teratas, Samsung SystemUI khusus menyertakan layanan com.android.systemui.PhoneSettingService, yang menangani permintaan masuk untuk menghidupkan/mematikan berbagai setelan seluruh sistem yang muncul di bilah status atas. Pengaturan ini termasuk menyalakan/mematikan wifi, bluetooth, lokasi, data seluler, tathering, mode mengemudi, dll; yang biasanya dilakukan dengan persetujuan pengguna. Analisis yang dilakukan menunjukkan ketidakcocokan perlindungan untuk layanan ini. Pada S5(4.4.2) dan Note8(4.4.2), layanan ini dilindungi dengan izin Signature com.sec.phonesettingservice.permission.PHONE_ SETTING, sedangkan pada Catatan 2, 4.4.2, layanan tidak dilindungi dengan izin.
7. **Kasus Dipilih Secara Acak Lainnya**. Dampak dari konfigurasi keamanan yang tidak konsisten adalah signifikan. Selain serangan end-to-end, sejumlah 40 sampel ketidakkonsistenan dipilih secara acak dan dianalisis secara manual untuk mengetahui apa yang bisa terjadi setelah dieksploitasi.

Tabel 1 Dampak Konfigurasi Keamanan yang Tidak Konsisten

p-ISSN : 2808-909X e-ISSN : 2798-2505

Inconsistent Configuration Category	Impact	Specific Examples
Permission Protection Change	Change System / App Wide Settings	Xiaomi Cloud Settings, Activate SIM
Removed Protected Broadcasts	Trigger Dangerous Operations and events	Trigger data sync, SMS received Airplane mode active, SIM is full
Non-Protected Content Providers	Data Pollution	Write to system logs, Add contacts Change instant messaging configurations
Non-Protected Content Providers	Data Leaks	Read emails, Read contacts Read blocked contact lists
Non-Protected Services	Trigger Dangerous Operations	Access Location, Bind to printing services Kill specific apps, Trigger backup
Non-Protected Activities	Change System wide Settings	Change Telephony settings, Access hidden activities
Non-Protected Receivers	Trigger Dangerous Operations	Send SMS messages, Trigger fake alerts Alter telephony settings , Issue SIM commands

Keterbatasan

- **Perubahan implementasi komponen.** Perubahan statis konfigurasi keamanan komponen (visibilitas atau perlindungan izin) mungkin tidak selalu menunjukkan risiko keamanan sepanjang waktu.
- **Penggantian nama komponen.** Pendekatan yang diguanakan akan melewatkan pendeteksian konfigurasi komponen yang tidak konsisten yang telah diganti namanya selama kustomisasi.

KESIMPULAN DAN SARAN

Penelitian ini melakukan kajian tentang kustomisasi Android yang berkaitan dengan aspek keamanan. Tujuan penelitian ini adalah untuk menyelidiki secara sistematis setiap ketidakkonsistenan yang tercipta sebagai hasil dari proses ini dan untuk menilai berbagai implikasi keamanannya. Penyelidikan pertama mengarah pada penemuan kelemahan keamanan Android serius yang belum pernah dipelajari sebelumnya. Penelitian ini menyoroti pentingnya risiko keamanandan mengungkapkan kerusakan pada berbagai perangkat Android, serta menunjukkan bahwa perangkat android penuh dengan kekurangan tersebut. Kedua, penelitain ini melakukan upaya pertama untuk secara sistematis mendeteksi perubahan konfigurasi keamanan yang diperkenalkan oleh kustomisasi Android. Penelitian ini membuat daftar fitur keamanan yang diterapkan di berbagai lapisan Android dan memanfaatkan analisis diferensial di antara sejumlah besar ROM khusus untuk mengetahui apakah mereka konsisten di semua lapisan tersebut. Dengan membandingkan konfigurasi keamanan dari gambar serupa, perubahan keamanan penting yang bisa jadi secara tidak sengaja diperkenalkan selama penyesuaian dapat ditemukan. Hasil meunjukkan bahwa, DroidDiff dapat diguanakn vendor guna memeriksa konfigurasi berbagai fitur keamanan pada image yang diberikan. DroidDiff akan mengekstrak fitur tersebut dari gambar, dan membandingkannya dengan kumpulan konfigurasi gambar lain, selanjutnya DroidDiff akan menandai yang tidak konsisten untuk diselidiki lebih lanjut oleh vendor yang memiliki kode sumber dan perangkat untuk memeriksa efeknya.

Untuk pekerjaan dimasa depan, penelitian ini menyarankan untuk menyempurnakan DroidDiff dalam mendeteksi ketidakkonsistenan yang berisiko secara lebih akurat. Selain itu, kemiripan antar komponen juga harus dihitung dan konfigurasi keamanannya harus dperiksa lebih teliti lagi untuk mendeteksi kasus yang mungkin terlewatkan. Dengan meningkatkan DroidDiff akan membantu mengurangi jumlah positif palsu dan menentukan konfigurasi berisiko secara lebih akurat. Sehingga hasil analisis diferensial DroidDiferensial dapat digunakan untuk memprediksi konfigurasi keamanan yang benar dari fitur yang salah dikonfigurasi. Jika sebagian besar fitur keamanan memiliki konfigurasi yang sama, maka komponen yang tidak konsisten kemungkinan besar akan dikonfigurasi dengan cara yang sama pada citra korban.

DAFTAR PUSTAKA

M. C. Grace, Y. Zhou, Z. Wang, and X. Jiang, “Systematic detection of capability leaks in stock android smartphones,” in 19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012, 2012.

L. Wu, M. Grace, Y. Zhou, C. Wu, and X. Jiang, “The impact of vendor customizations on android security,” in Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS ’13, (New York, NY, USA), pp. 623–634, ACM, 2013.

X. Zhou, Y. Lee, N. Zhang, M. Naveed, and X. Wang, “The peril of fragmentation: Security hazards in android device driver customizations,” in 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA.

M. Mitchell, G. Tian, and Z. Wang, “Systematic audit of third-party android phones,” in Proceedings of the 4th ACM Conference on Data and Application Security and Privacy, CODASPY ’14, (New York, NY, USA), pp. 175–186, ACM, 2014.

D. R. Thomas, A. R. Beresford, and A. Rice, “Security metrics for the android ecosystem,” in Proceedings of the 5th Annual ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices, SPSM ’15, (New York, NY, USA), pp. 87–98, ACM, 2015.

A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin, “Permission re-delegation: attacks and defenses,” in Proceedings of the 20th USENIX conference on Security symposium,2011.

Y. Zhou and X. Jiang, “Detecting passive content leaks and pollution in android applications,” in 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013,2013.

R. Gallo, P. Hongo, R. Dahab, L. C. Navarro, H. Kawakami, K. Galvão, G. Junqueira, and L. Ribeiro, “Security and system architecture: Comparison of android customizations,” in Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks,WiSec ’15,(NewYork,NY,USA), pp. 12:1–12:6, ACM, 2015.

S. Jana and V. Shmatikov, “Memento: Learning secrets from process footprints,” in Proceedings of the 2012 IEEE Symposium on Security and Privacy,SP ’12, (Washington, DC, USA), pp. 143–157, IEEE Computer Society, 2012.

X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and

M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach, “Quire: Lightweight provenance for smart phone operating systems,” in 20th USENIX Security Symposium, (San Francisco, CA), Aug. 2011.

Y. Zhou, X. Zhang, X. Jiang, and V. W. Freeh, “Taming information-stealing smartphone applications (on android),” in Proceedings of the 4th International Conference on Trust and Trustworthy Computing, TRUST’11, (Berlin, Heidelberg), pp. 93–107, Springer-Verlag, 2011.

P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall, “These aren’t the droids you’re looking for: Retrofitting android to protect data from imperious applications,” in Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS ’11, (New York, NY, USA), pp. 639–652, ACM, 2011.

M. Nauman, S. Khan, and X. Zhang, “Apex: extending android permission model and enforcement with user-defined runtime constraints,” in Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security,2010.

S. Bugiel, L. Davi, A. Dmitrienko, T. Fischer, and A.-R. Sadeghi, “Xmandroid: A new android evolution to mitigate privilege escalation attacks,” tech. rep., Technische UniversitÃÂČÂâĆňt Darmstadt, 2011.

S. Bugiel, L. Davi, A. Dmitrienko, S. Heuser, A.-R. Sadeghi, and B. Shastry,

“Practical and lightweight domain isolation on android,” in Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices,2011.

S. Bugiel, S. Heuser, and A.-R. Sadeghi, “Flexible and fine-grained mandatory access control on android for diverse security and privacy policies,” in Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13), (Washington, D.C.), pp. 131–146, USENIX, 2013.

S. Smalley and R. Craig, “Security enhanced (SE) android: Bringing flexible MAC to android,” in 20th Annual Network and Distributed System Security Symposium, NDSS 2013, San Diego, California, USA, February 24-27, 2013,2013.

Z. Fang, W. Han, D. Li, Z. Guo, D. Guo, X. S. Wang, Z. Qian, and H. Chen, “revdroid: Code analysis of the side effects after dynamic permission revocation of android apps,” in Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, ASIA CCS ’16, (New York, NY, USA), pp. 747–758, ACM, 2016.

p-ISSN : 2808-909X e-ISSN : 2798-2505

G. Russello, B. Crispo, E. Fernandes, and Y. Zhauniarovich, "Yaase: Yet another android security extension," in Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on,pp. 1033–1040, 2011.

D. Feth and C. Jung, Context-Aware, Data-Driven Policy Enforcement for Smart Mobile Devices in Business Environments,pp. 69–80. Berlin,Heidelberg: Springer Berlin Heidelberg, 2012.

P. Pearce, A. P. Felt, G. Nunez, and D. Wagner, "AdDroid: Privilege Separation for Applications and Advertisers in Android," in Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security,2012.

S. Shekhar, M. Dietz, and D. S. Wallach, "Adsplit: Separating smartphone advertising from applications," in Proceedings of the 21st USENIX Conference on Security Symposium, Security'12, (Berkeley, CA, USA), pp. 28–28, USENIX Association, 2012.

X. Zhang, A. Ahlawat, and W. Du, "AFrame: Isolating Advertisements from Mobile Applications in Android," in Proceedings of the 29th Annual Computer Security Applications Conference (

58] L. Xing, X. Pan, R. Wang, K. Yuan, and X. Wang, "Upgrading your android, elevating my malware: Privilege escalation through mobile os updating," in Proceedings of the 2014 IEEE Symposium on Security and Privacy,SP '14, (Washington, DC, USA), pp. 393–408, IEEE Computer Society, 2014.

B. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Android Permissions: A Perspective Combining Risks and Benefits," SACMAT,2012.

H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy, "Using probabilistic generative models for ranking risks of android apps," in Proceedings of the 2012 ACM conference on Computer and communications security, 2012.

T. Blasing, L. Batyuk, A.-D. Schmidt, S. A. Camtepe, and S. Albayrak, "An Android Application Sandbox System for Suspicious Software Detection," MALWARE, 2010.

Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets," NDSS,2012.

A.-D. Schmidt, R. Bye, H.-G. Schmidt, J.Clausen, O. Kiraz, K. A. Yuksel, S. A. Camtepe, and S. Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android," ICC,2009.

I. Burguera, U. Zurutuza, and S.Nadijm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android.," SPSM,2011.

J. Huang, X. Zhang, L. Tan, P. Wang, and B. Liang, "Asdroid: Detecting stealthy behaviors in android applications by user interface and program behavior contradiction," in Proceedings of the 36th International Conference on Software Engineering, ICSE 2014, (New York, NY, USA), pp. 1036–1046, ACM, 2014.

D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket.," in NDSS, The Internet Society, 2014.

M. Zhang, Y. Duan, H. Yin, and Z. Zhao, "Semantics-aware android malware classification using weighted contextual api dependency graphs," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, (New York, NY, USA), pp. 1105–1116, ACM, 2014.

A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, "Andromaly: a Behavioral Malware Detection Framework for Android Devices," Journal of Intelligent Information Systems archive Volume 38 Issue 1,2012.

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 Trans. Comput. Syst.,vol.32, pp. 5:1–5:29, June 2014.

G. Portokalidis, P. Homburg, K. Anagnostakis, and H. Bos, "Paranoid Android: Versatile Protection for Smartphones," ACSAC,2010.

S. Arzt, S. Rasthofer, C. Fritz, E. Bodden, A. Bartel, J. Klein, Y. Le Traon, D. Octeau, and P. McDaniel, "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, PLDI '14, (New York, NY, USA), pp. 259–269, ACM, 2014.

M. I. Gordon, D. Kim, J. Perkins, L. Gilham, N. Nguyen, and M. Rinard, "Information-flow analysis of android applications in droidsafe," 2015.

M. Zhang and H. Yin, "Appsealer: Automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications," in NDSS,2014.

M. Zhang and H. Yin, "Efficient, context-aware privacy leakage confinement for android applications without firmware modding," in Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security, ASIA CCS '14, (New York, NY, USA), pp. 259–270, ACM, 2014.

K. Z. Chen, N. M. Johnson, V. D'Silva, S. Dai, K. MacNamara, T. R. Magrino, E. X. Wu, M. Rinard, and D. X. Song, "Contextual policy enforcement in android applications with permission event graphs.," in NDSS, The Internet Society, 2013.

W. Yang, X. Xiao, B. Andow, S. Li, T. Xie, and W. Enck, "Appcontext: Differentiating malicious and benign mobile app behaviors using context," in Proceedings of the 37th International Conference on Software Engineering - Volume 1, ICSE '15, (Piscataway, NJ, USA), pp. 303–313, IEEE Press, 2015.

Y. Fratantonio, A. Bianchi, W. Robertson, E. Kirda, C. Kruegel, G. Vigna, S. Uc, and Barbara, "Triggerscope: Towards detecting logic bombs in android applications," in S&P,2016.

L. K. Yan and H. Yin, "Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis," in Proceedings of the 21st USENIX conference on Security symposium,2012.

M. Spreitzenbarth, F. Freiling, F. Echtler, T. Schreck, and J. Hoffmann, "Mobile-sandbox: Having a deeper look into android applications," in Proceedings of the 28th Annual ACM Symposium on Applied Computing,SAC '13, (NewYork, NY, USA), pp. 1808–1815, ACM, 2013.

K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "Copperdroid: Automatic reconstruction of android malware behaviors.," in NDSS, The Internet Society, 2015.

M. Lindorfer, M. Neugschw, L. Weichselbaum, Y. Fratantonio, V. V. D. Veen, and C. Platzer, "Andrubis- 1,000,000 apps later: A view on current android malware behaviors," in International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security,2014.

Y. Zhang, M. Yang, B. Xu, Z. Yang, G. Gu, P. Ning, X. S. Wang, and B. Zang, "Vetting undesirable behaviors in android apps with permission use analysis," in CCS, (New York, NY, USA), ACM, 2013.

N. Mirzaei, S. Malek, C. S. Păsăreanu, N. Esfahani, and R. Mahmood, "Testing android apps through symbolic execution," SIGSOFT Softw. Eng. Notes,vol.37, pp. 1–5, Nov. 2012.

M. Y. Wong and D. Lie, "Intellidroid: A targeted input generator for the dynamic analysis of android malware," in NDSS,2016.

S. Rasthofer, S. Arzt, M. Miltenberger, and E. Bodden, "Harvesting runtime values in android applications that feature anti-analysis techniques," in NDSS,2016.