JURNAL TEKNIK INFORMATIKA DAN MULTIMEDIA Vol 3, No.1, Mei 2023, pp. 124 - 143
p-ISSN : 2808-909X (print)
e-ISSN : 2798-2505 (online)
http://journal.politeknik-pratama.ac.id/index.php/JTIM · page 124

ATRIBUT ANCAMAN HANGING IN THE WILD ANDROID

Neilin Nikhlis [1], Edy Siswanto[2]

[1] Program Studi Sistem Komputer Universitas Sains dan Teknologi Komputer
Jl. Majapahit 605 Semarang, telp : (024)-6723456, e-mail: neilin.nikh@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 complex system, whose components and applications are meant to work together, giving rise to highly complex interdependence relationships among them. Meanwhile, the Android ecosystem is known for being highly diverse and decentralized: each OS version is customized and re-customized by various parties almost independently and used by anyone who can build an app for that version. Android protects its information assets through an app sandbox and permissions model, where each app runs in its own compartment and can only access sensitive global resources and other app components (content providers, services, activities, broadcast receivers) with the proper permissions. This study uses Harehunter measurement with the aim of automatically detecting Hare vulnerabilities in Android system applications. Harehunter and HareGuard performance evaluations were carried out in this study, both of which proved to be highly effective. The approach used here is differential analysis, by searching all extracted, decompiled code, and manifest files for targeted attribute definitions as an initial step, and running an XML parser. The results of this study indicate that the impact of Hares is very significant. The application of HareGuard in this study proved to be effective in detecting all attack applications that were made. Further evaluation of the performance impact on the minimum system host. For future research, to make Harehunter more effective, it is suggested to use a more qualified analyzer. So that this direction can be explored in more depth.

Keywords: Harehunter, HareGuard, Android, Deteksi Malware Android

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

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

Abstrak

Android adalah sistem yang kompleks, yang komponen dan aplikasinya dimaksudkan untuk bekerja bersama, menimbulkan hubungan saling ketergantungan yang sangat rumit di antara. Sementara itu, ekosistem Android dikenal sangat beragam dan terdesentralisasi: setiap versi OS dikustomisasi dan dikustomisasi ulang oleh berbagai pihak hampir secara independen dan digunakan oleh siapa saja yang dapat membuat aplikasi untuk versi tersebut. Android melindungi aset informasinya melalui kotak pasir aplikasi dan model izin, di mana setiap aplikasi berjalan dalam kompartemennya sendiri dan hanya dapat mengakses sumber daya global yang sensitif dan komponen aplikasi lainnya (penyedia konten, layanan, aktivitas, penerima broadcast) dengan izin yang tepat. Penelitian ini menggunakan pengurkuran Harehunter dengan tujuan untuk mendeteksi kerentanan Hare dalam aplikasi sistem Android secara otomatis. Eveluasi kinerja Harehunter dan HareGuard dilakukan dalam penelitian ini, yang keduanya terbukti sangat efektif. Pendekatan yang diguanakan disini adalah analisis diferensial, dengan mencari semua kode yang diekstrak, didekompilasi, dan file manifes untuk definisi atribut yang ditargetkan sebagai langkap awal, dan menjalankan parser XML. Hasil penelitian ini menunjukkan bahwa dampak Hares sangat signifikan. Penerapan HareGuard dalam penelitian ini terbukti efektif dalam mendeteksi semua aplikasi serangan yang dibuat. Evaluasi lebih lanjut terhadap kinerja berdampak pada inang sistem minimum. Untuk penelitian dimasa depan, agar Harehunter lebih efektif, disarankan untuk menggunakan pengaalisa yang lebih mumpuni. Sehingga arah ini dapat dieksplorasi lebih dalam lagi.

Keywords: Harehunter, HareGuard, Android, Deteksi Malware Android

1. PENDAHULUAN

Pada dasarnya, apa yang menyebabkan masalah di sini adalah hubungan saling ketergantungan intrinsik antara komponen Android yang berbeda (aplikasi dan layanan kerangka kerja), yang menghubungkan satu pihak ke pihak lain melalui referensi ke atribut yang terakhir seperti paket, aktivitas, nama layanan, otoritas penyedia konten, dan izin. Penyesuaian yang dilakukan pada komponen tersebut, jika tidak dipikirkan dengan baik, dapat dengan mudah merusak beberapa hubungan tersebut, yang mengakibatkan referensi ke atribut yang tidak ada (misalnya, otoritas penyedia SMS/MMS tidak ada di tablet), dalam penelitian ini disebut sebagai referensi atribut gantung, atau Hares. Sebagai efek samping dari fragmentasi Android, Hares juga dapat dibawa oleh pengembang pihak ketiga yang mendesain aplikasinya untuk berjalan di berbagai versi Android, dengan atau tanpa komponen layanan tertentu yang digunakannya. Misalnya, referensi ke penyedia konten perpesanan yang tidak ada juga dapat disematkan di aplikasi pihak ketiga yang berfungsi baik di smartphone maupun tablet. Dibandingkan dengan kelemahan kustomisasi yang ditemukan dalam penelitian sebelumnya, yaitu tentang kesalahan konfigurasi driver perangkat lapisan Linux), referensi gantung adalah masalah lapisan kerangka kerja dan berpotensi lebih menyebar, mengingat fakta bahwa aplikasi sistem pada lapisan itu selalu menjadi fokus dari kustomisasi (Nappa et al., (2012)). Namun, masalah seperti itu belum pernah dipelajari, yang implikasi keamanannya, ruang lingkup dan besarnya, oleh karena itu, tidak jelas sama sekali.

LITERATUR REVIEW

Demistifikasi Keamanan Android

Survei Keamanan Android

Pandangan tingkat tinggi keamanan Android disajikan dalam study Chaudhuri et al., (2011) bertujuan untuk lebih memahami vektor serangan aplikasi Android melalui karakterisasi sistematis keamanan aplikasi Android populer. Para penulis mempertimbangkan berbagai masalah termasuk fungsionalitas berbahaya dan kerentanan dan mengusulkan pengurai Dalvik untuk analisis. Kedua karya menyoroti kekhawatiran pada aplikasi yang memiliki hak istimewa dan mempertanyakan arsitektur keamanan berbasis izin Android. Studi serupa Christin et al., (2011) memberikan survei lain tentang keamanan Android secara umum. Pekerjaan pertama menyediakan taksonomi kelas serangan platform seluler dengan contoh spesifik (seperti serangan pengemasan ulang Android, eksekusi muatan jarak jauh, dll.) karena setiap kelas berlaku untuk lingkungan Android, kemudian mengusulkan mitigasi jika memungkinkan. Baru-baru ini, SoK oleh Smith et al., (2016) mensistematisasikan pekerjaan penelitian tentang keamanan dan privasi Android di platform aplikasi. Untuk mengevaluasi dan membandingkan pendekatan berbeda yang ada secara objektif, penulis pertama-tama memiliki pemahaman umum tentang

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

beberapa tantangan dan model serangan yang mengancam ekosistem Android, kemudian menciptakan pemahaman terpadu tentang kemampuan penyerang.

Izin Android

Izin Android dimaksudkan untuk melindungi sumber daya Android penting pada lapisan kerangka kerja. Secara opsional, aplikasi dapat menggunakan izin bawaan atau khusus untuk melindungi sumber dayanya sendiri (penyedia konten, layanan, aktivitas, atau penerima broadcast). Penelitian oleh Wagner et al., (2011) memanfaatkan analisis dinamis untuk mendemistifikasi penggunaan izin android, melalui pemetaan API ke persyaratan izinnya. Untuk mengatasi beberapa keterbatasan Stowaway (Wagner et al., (2011)), PScout (Lie et al., (2012)) mengusulkan alat analisis statis yang bertujuan untuk menghasilkan spesifikasi lengkap untuk sistem izin Android yang mencantumkan persyaratan izin untuk setiap panggilan API. Alat ini melakukan analisis keterjangkauan antara panggilan API dan pemeriksaan izin di seluruh kode sumber Android. Studi yang dilakukan oleh Cowan et al., (2012) mengusulkan realisasi nyata dari UI tepercaya dalam bentuk gadget kontrol akses yang memungkinkan pendelegasian izin berbasis pengguna ke aplikasi kapan pun widget tersebut dapat diintegrasikan secara efektif ke dalam alur kerja aplikasi. Penelitian yang dilakukan oleh Beznosov et al., (2015) bertujuan untuk meningkatkan efektivitas izin Android dengan menggunakan gagasan privasi sebagai integritas kontekstual. Lebih khusus lagi, penulis mengusulkan model permintaan izin baru, yang hanya akan meminta pengguna saat aplikasi mengakses data sensitif dengan cara yang bertentangan dengan harapan pengguna. Sadeh et al., (2014) mengusulkan untuk mengurangi daftar izin yang dihadapi pengguna pada waktu penginstalan aplikasi dan menggantinya dengan daftar ringkas yang mencerminkan profil privasi. Penelitian lain yang dilakukan oleh Xie et al., (2013) dan Chen et al., (2014) tentang izin Android menggunakan teknik NLP untuk menganalisis deskripsi aplikasi Android, memperoleh izin yang diperlukan, dan memeriksa apakah izin tersebut sesuai dengan izin efektif yang diminta dalam file manifes Android. Yin et al., (2015) mengambil pendekatan yang berbeda dan menghasilkan deskripsi aplikasi yang berpusat pada keamanan dari analisis kode aplikasi dalam upaya mengedukasi pengguna Android tentang pemahaman fungsionalitas aplikasi tertentu pada waktu penginstalan.

Android dan Web

Peneliti lain berfokus untuk mengungkap kerentanan dalam aplikasi Android tertentu di lanskap web. Yinet al., (2011) menyajikan studi sistematis pertama tentang masalah keamanan WebView dan menemukan beberapa serangan yang mengungkapkan masalah mendasar dalam TCB dan kotak pasir yang melemah dari Webview Android infrastruktur. Chen et al., (2013) melakukan studi sistematis lainnya untuk memahami penyeberangan asal yang tidak sah pada OS seluler dan menyoroti keberadaan kerentanan semacam itu di aplikasi profil tinggi. Penelitin yang dilakukan oleh Peri et al., (2014) telah memperluas cakupan untuk mencakup serangan injeksi kode pada semua aplikasi seluler berbasis HTML5. Serangan yang dilakukan terutama menargetkan aplikasi rentan yang memanfaatkan fitur WebView yang disediakan oleh Android. Penelitian sebelumnya oleh Wagner et al., (2011) telah mempelajari penerimaan niat tidak sah di mana penyerang dapat membajak aktivitas dan layanan jika ada niat implisit. Karya tersebut tidak secara langsung menyentuh kelemahan Hare karena tidak mensyaratkan tidak adanya aktivitas/layanan sah yang dirujuk. Sebaliknya, ini membahas kasus di mana banyak penerima hadir di perangkat. Pekerjaan lain termasuk mengevaluasi risiko keamanan yang dihasilkan dari cacat desain dalam perpesanan push-cloud (Han et al., (2014)), mengidentifikasi risiko proses uninstall aplikasi Android (Qiu et al., (2016)) dan risiko komponen Clipboard Android dan mekanisme berbagi (Smith et al., (2013)). Dua studi terbaru meneliti lebih lanjut penyalahgunaan crypto di aplikasi Android (Fratantonio et al., (2013); Lee et al., (2013)). Pekerjaan lainnya termasuk menemukan kerentanan pada desain Android yang cacat, seperti penelitian yang dilakukan oleh Huang et al., (2015) mengeksploitasi kelemahan di server sistem Android untuk memasang beberapa serangan DoS dan penelitian Qian et al., (2015) mengungkap penyedia root Android dan menunjukkan bahwa eksploitasi yang direkayasa dengan baik ini tidak baik. dilindungi, dan bisa sangat berbahaya jika dieksploitasi.

Keamanan GUI

Keamanan GUI telah dipelajari secara ekstensif dalam konteks OS Android dengan hubungannya dengan desain unik dan subsistem GUI. Telah dibuktikan bahwa kerahasiaan GUI Android dapat dilanggar dengan menyematkan komponen UI dari sumber berbahaya (Yinet al., (2011); Roesner&Kohno et al., (2013)), melalui perintah adb untuk mengambil cuplikan layar tanpa sepengetahuan pengguna(Wang et al., (2014)), melalui saluran samping lain seperti memori bersama (Mao et al., (2014)) , atau membaca sensor perangkat (Balakrishnan et al., (2012)). Baru-baru ini, penelitian yang dilakukan oleh Liu et al., (2015)

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

melakukan evaluasi keamanan sistematis multitasking Android dan desain ActivityManagerService secara mendalam dan menemukan permukaan serangan terbuka lebar yang memungkinkan membingungkan pengguna tentang UI yang ditampilkan dan mengancam kerahasiaannya.

Saluran sampingan: Memanfaatkan Sumber Daya yang Tidak Dilindungi

Beberapa penelitian menunjukkan kebocoran informasi yang tidak diinginkan menggunakan sensor gerak yang disediakan oleh perangkat seluler. Studi lain yang ndilakukan oleh Nordin et al., (2013) menunjukkan manfaat mengeksploitasi giroskop dibandingkan dengan sensor gerak lainnya pada perangkat. untuk menyimpulkan penekanan tombol. Studiyang dilakukan oleh Nakibly et al., (2014) menunjukkan bahwa giroskop pada telepon pintar dapat digunakan untuk menguping percakapan di sekitar telepon dan mengidentifikasi pembicara. Nahrstedt et al., (2013) mengungkapkan bahwa status nyala/mati audio adalah saluran samping untuk pelacakan lokasi tanpa izin. Penelitian oleh Boneh et al., (2015) menunjukkan bahwa dengan hanya membaca konsumsi daya agregat telepon selama beberapa menit, aplikasi dapat mempelajari informasi tentang lokasi pengguna. Penulis menggunakan algoritme machine learning untuk menyimpulkan lokasi pengguna secara akurat.

Penelitian ini melaporkan jenis kerentanan yang tunjukkan memang kritis terhadap keamanan dan ekstensif. Penelitian ini menunjukkan bahwa perangkat Android populer dipenuhi dengan kelemahan seperti itu, yang seringkali memiliki implikasi keamanan yang serius: ketika atribut digunakan pada perangkat tetapi pihak yang menentukannya telah dihapus, aplikasi berbahaya dapat mengisi celah untuk memperoleh kapabilitas sistem kritis, hanya dengan menyamar sebagai pemilik atribut. Lebih khusus lagi, ditemukan bahwa Hare pada Note 8.0 dapat dieksploitasi untuk mencuri catatan suara pengguna dan kelemahan lain pada Tab S 8.4 memungkinkan aplikasi jahat untuk menyamar sebagai pelindung Facelock untuk mendapatkan kendali atas autentikasi login pengguna. Aplikasi Tango yang populer berisi referensi yang tidak dilindungi ke sms yang hilang, yang dapat dimanfaatkan untuk mencuri pesan pengguna. Juga melalui pembajakan berbagai paket, aktivitas, atau penyedia konten yang hilang, musuh dapat mengganti antarmuka pengaturan akun internal Google Email, menyuntikkan aktivitas ke LG FileManager dan LG CloudHub untuk mencuri kata sandi pengguna, dan mengelabui S-Voice agar meluncurkan program jahat kapan saja pengguna perlu menggunakan perekam suara pra-instal. Selain itu, pada Note 3 (ponsel) dan Note 8.0 (tablet), Rabbit terkait dengan izin yang tidak ada dapat dieksploitasi untuk mencuri semua informasi kontak (misalnya, email, nomor telepon, dll.) dari pengguna perangkat dan bahkan mengutak-atik kontennya (mis., mengubah nomor telepon, email, dan URL teman menjadi yang berada di bawah kendali musuh), ketika aplikasi jahat tidak memiliki hak istimewa untuk melakukannya.

Untuk memahami ruang lingkup dan besarnya bahaya keamanan yang diperkenalkan oleh Hares, sehingga penelitian ini mengusulkan dan menjalankan alat baru untuk secara otomatis mengevaluasi lebih dari 97 citra OS untuk perangkat Google, Samsung, LG, HTC, dan Motorola. Studi pengukuran ini menunjukkan bahwa Hares yang tidak terlindungi ada di setiap perangkat yang diuji dan sepenuhnya terbuka untuk dieksploitasi. Meskipun kelemahan tersebut dapat disebabkan oleh operator dan pihak lain, tampaknya kelemahan tersebut terutama diperkenalkan oleh produsen saat menyesuaikan OS yang sama ke model perangkat yang berbeda. Selain itu, masalah masih menyebar bahkan pada versi OS dan model ponsel terbaru, di berbagai pabrikan, menunjukkan bahwa risiko keamanan ini belum menjadi perhatian mereka. Temuan ini menunjukkan betapa seriusnya bahaya keamanan tersebut dan kebutuhan mendesak untuk mengembangkan solusi yang efektif untuk mengatasinya.

Studi pengukuran penelitian ini dimungkinkan oleh Harehunter, alat baru untuk deteksi otomatis kerentanan Hare dalam aplikasi sistem. Untuk tujuan ini, Harehunter pertama-tama melakukan analisis diferensial, membandingkan semua atribut yang ditentukan oleh aplikasi sistem pada gambar Android dengan atribut yang dirujuk oleh mereka. Perbedaan apa pun antara definisi dan referensi mengungkapkan risiko Hare. Instance ini dievaluasi lebih lanjut melalui analisis program otomatis untuk mengetahui apakah ia benar-benar dilindungi: misalnya, apakah tanda tangan sebuah paket telah diverifikasi sebelum aktivitasnya dijalankan. Jika tidak, maka masalahnya dilaporkan sebagai kemungkinan kasus Hare (LHare). Menjalankan Harehunter di 97 image perangkat populer, ditemukan 21557 kemungkinan Hares dalam 3450 aplikasi sistem yang rentan, yang telah didokumentasikan dalam database. Basis data ini digunakan oleh aplikasi pelindung yang dikembangkan, yang disebut HareGuard, untuk memeriksa setiap aplikasi yang baru dipasang di perangkat ini, mengidentifikasi aplikasi mencurigakan yang berupaya mengeksploitasi Hares di sana, sehingga mengamankan perangkat bahkan sebelum pabrikannya dapat memperbaiki masalah. Studi ini selanjutnya mengevaluasi kemanjuran dan kinerja Harehunter dan HareGuard, yang keduanya terbukti sangat efektif.

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

Referensi atribut dan model keamanan Android

Berbagai komponen Android (aplikasi atau aktivitas internalnya, layanan, penyedia konten, penerima, dll.) dihubungkan bersama oleh Inter-Component Communication (ICC), seperti pesan Intent. Intent adalah pesan yang menjelaskan operasi yang akan dilakukan oleh penerima: misalnya, startActivity yang memicu aktivitas (serangkaian operasi terkait antarmuka pengguna) yang terkait dengan aplikasi. Nama paket aplikasi dan nama aktivitas dapat ditentukan melalui Intent, menggunakan metode setPackage, setClassName, setComponent, dll. Di sini referensi dari satu komponen ke komponen lain terjadi melalui atribut yang terakhir, yaitu nama paket dan nama aktivitas. Saat atribut ini belum disetel untuk komunikasi, Intent bersifat implisit dan harus diselesaikan oleh OS untuk menemukan penerima yang mampu menanganinya. Dalam hal ini, pengirim perlu memberikan tindakan dan parameter lain (seperti data), dan penerima harus mendeklarasikan filter Intent untuk komponennya (aktivitas, layanan, penerima) yang cocok dengan parameter ini untuk mendapatkan Maksud. Komponen Android penting lainnya adalah penyedia konten, yang mengelola akses ke database aplikasi (set data terstruktur). Untuk beroperasi pada penyedia konten aplikasi lain, seseorang harus mendapatkan URI "content://authorityname/path", di mana tabel database yang sesuai dengan jalur dapat dibaca (kueri) dan ditulis di bawah persetujuan dari pemiliknya. Dalam semua komunikasi ICC tersebut, setelah target referensi (misalnya, nama paket, nama aktivitas, nama tindakan, dan nama otoritas) tidak ada di sistem yang sama, referensi menjadi menggantung, yang dapat menimbulkan implikasi keamanan yang serius.

Lebih khusus lagi, aplikasi dapat menentukan izin untuk setiap komponennya dan hanya memproses pesan atau permintaan layanan dari pihak yang memiliki izin. Misalnya, penyedia konten dapat dilindungi dengan readPermission dan writePermission; penerima broadcast dapat dikonfigurasi untuk mendapatkan pesan hanya dari mereka yang memiliki izin khusus. Perlindungan izin semacam itu sebagian besar disetel secara statis dalam file manifes aplikasi, tetapi juga dapat ditentukan secara terprogram, menggunakan API seperti checkPermission. Aplikasi yang ingin mendapatkan izin tersebut harus meminta persetujuan pengguna. Namun, ketika pihak yang menentukan izin tersebut tidak ada pada versi khusus, perlindungan izin menjadi menggantung: siapa pun yang menetapkan izin dapat memperoleh hak istimewa secara diam-diam untuk mengakses komponen aplikasi yang dilindungi.

Model lawan

Skenario di mana aplikasi berbahaya telah dipasang di perangkat target dalam penilitian ini dipertimbangkan. Namun, aplikasi tidak perlu memiliki izin yang mencurigakan. Sebenarnya, dalam kasus perlindungan izin yang ditangguhkan, itu dapat menentukan sendiri izin yang hilang untuk meluncurkan semua jenis serangan. Untuk mengirimkan informasi yang dicuri dari perangkat, aplikasi membutuhkan kemampuan komunikasi. Ini dapat dilakukan secara eksplisit dengan meminta izin jaringan, yang telah diminta oleh hampir semua aplikasi. Alternatifnya, aplikasi jahat dapat memanfaatkan saluran lain, seperti browser, untuk mengirimkan data (Brodley et al., (2013)).

Mengeksploitasi Hares

Seperti disebutkan sebelumnya, referensi atribut gantung dapat berupa panggilan ICC ke paket, aktivitas, layanan yang tidak ada (yang dapat ditentukan secara implisit oleh tindakan atau filter data) atau otoritas penyedia konten, atau penggunaan izin yang hilang untuk melindungi komponen aplikasi (layanan, aktivitas, penerima broadcast, dan penyedia konten). Di hadapan referensi semacam itu, aplikasi jahat yang mengklaim atribut targetnya dapat memperoleh akses ke aset informasi yang diekspos oleh ICC atau dilindungi oleh izin. Lebih khusus lagi, ketika referensi tidak dijaga di sepanjang jalur eksekusi yang melibatkan Hare, yaitu, tidak ada validasi keberadaan dan keabsahan atribut sebelum menggunakannya, malware yang memperoleh atribut (misalnya nama paket/otoritas/izin) secara otomatis memperoleh hak istimewa yang terkait dengan atribut dan berhak menerima pesan sensitif dari pengirim, memanfaatkan komponennya, dll.

Penting untuk dicatat bahwa tidak semua referensi gantung dapat dieksploitasi. Itu dapat dilindungi dengan memverifikasi keberadaan paket yang seharusnya mendefinisikannya dan kemudian memverifikasi tanda tangannya (diekstraksi melalui getPackageInfo dengan flag GET_SIGNATURE S), atau info aplikasinya FLAG_SYSTEM, atau dengan memeriksa model perangkat saat ini, kode negara, atau properti lainnya (misalnya getProperty). Kehadiran perlindungan tersebut diidentifikasi dalam penelitian ini melalui analisis kode otomatis. Di sisi lain, jika pemeriksaan keamanan tidak dilakukan, Rabbit menjadi rentan untuk dieksploitasi, meskipun masih sulit menemukan kondisi untuk memicu kode. Dalam penelitian ini sejumlah 97 gambar pabrik Android dari produsen perangkat utama (Google, Samsung, LG, HTC,

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

Motorola) dianalisis secara sitematis dan ditemukan 21557 referensi atribut gantung yang cenderung rentan ntuk memahami risiko keamanan yang mungkin ditimbulkannya, serangan end-to-end dibuat pada beberapa instans Hare. Kecuali sebagian kecil dari mereka yang ditemukan secara manual, yang memotivasi seluruh penelitian, sebagian besar Hares, terutama yang berada dalam aplikasi pra-instal, terdeteksi secara otomatis menggunakan Harehunter.

Pembajakan Paket, Tindakan, dan Aktivitas

Di antara semua Hares yang ditemukan dalam penelitian ini, referensi yang menggantung sering menunjuk ke nama dan tindakan paket. Atribut ini memainkan peran penting dalam eksploitasi Hare, meskipun target utamanya adalah atribut lain. Ini karena paket yang hilang bisa menjadi pemilik dari aktivitas yang tidak ada, dan tindakan seringkali perlu ditentukan untuk menerima Intent yang disebabkan oleh referensi yang rentan. Selain itu, referensi untuk kegiatan yang tidak ada juga ditemukan meresap. Dengan mengeksploitasi kerentanan ini, malware dapat membiarkan sumber tepercaya (layanan sistem atau aplikasi) menjalankan aktivitas jahat, membuatnya terlihat cukup tepercaya bagi pengguna. Ini memungkinkan berbagai serangan phishing yang sangat realistis yang dapat menyebabkan pengungkapan data sensitif, seperti kata sandi. ***Mencuri catatan suara***. S-Voice adalah asisten pribadi dan aplikasi layanan navigator pengetahuan yang telah diinstal sebelumnya pada perangkat tertentu (mis. Note 8.0). Salah satu fiturnya adalah memo suara: pengguna cukup mengatakan "ambil memo" atau "catat" untuk mengaktifkan fungsionalitas dan mengikuti instruksi ("harap ucapkan catatan Anda") untuk merekam catatannya. Setelah catatan dibuat, aplikasi terlebih dahulu memeriksa apakah aplikasi sistem lain com.vendor.android.app.memo (disingkat memo) ada, dan jika demikian, menghubungkan dirinya sendiri ke layanan yang terakhir dengan memanggil bindService menggunakan nama tindakan yang ditentukan olehnya Filter niat. Ini menyerahkan catatan ke aplikasi memo. Jika aplikasi tidak ada, S-Voice mencari layanan sistem lain untuk menangani catatan suara. Ditemukan bahwa S-Voice gagal memverifikasi tanda tangan memo saat merujuknya. Akibatnya, pada perangkat yang aplikasinya hilang, referensi ke nama paket dan tindakannya (melalui bindService) menjadi hang. Aplikasi jahat kemudian dapat menyamar sebagai memo menggunakan nama paket/tindakannya untuk mencuri catatan suara pengguna.

Kunci pengaman AOSP curang. Sebelum 5.0, semua versi AOSP setelah 2.3 mendukung buka kunci layar berbasis wajah, yang dilakukan melalui aplikasi sistem yang disebut Facelock (com.android .facelock). Setelah opsi autentikasi biometrik ini dipilih oleh pengguna, layanan Android Keyguard akan mengikat dirinya ke layanan Facelock, memungkinkan pengguna untuk menggunakan wajahnya dan kamera depan untuk membuka kunci perangkatnya. Lebih khusus lagi, setiap kali fragmen pengaturan keamanan dalam aplikasi Pengaturan dibuat, aplikasi Pengaturan akan memanggil isBiometricWeakInstalled di kelas kerangka kerja LockPatternUtils untuk memeriksa apakah aplikasi Facelock diinstal. Jika demikian, itu akan menambahkan Facelock sebagai opsi kunci layar yang tersedia. Nanti saat pengguna mengklik opsi tersebut, Pengaturan mengirimkan Intent ke Facelock untuk konfigurasi. Setelah langkah ini selesai (yang juga termasuk mengonfigurasi PIN atau Pola cadangan), FaceUnlock ditetapkan sebagai opsi layar kunci. Di bawah opsi, setiap kali pengguna mengklik ponsel yang terkunci, Keyguard akan mengikat dirinya sendiri ke layanan buka kunci wajah dengan mengirimkan Intent yang menentukan tindakan com.android.internal.policy.IfaceLockInterface ke aplikasi Facelock. Layar tidak terkunci setelah Facelock memberi tahu Keyguard bahwa pengguna diautentikasi.

Masalahnya di sini adalah bahwa pada semua versi AOSP sebelum 5.0 yang mendukung opsi FaceUnlock, kelas kerangka kerja Android LockPatternUtils gagal memverifikasi tanda tangan aplikasi Facelock. Akibatnya, pada model perangkat yang aplikasinya tidak ada, referensi ke nama paketnya menjadi hang dan dapat dieksploitasi oleh aplikasi jahat. Dalam penelitian ini, Tab S 8.4 dipasangi sebuah aplikasi serangan yang menyamar sebagai com.android .facelock bersama dengan aktivitas penyiapan yang diperlukan dan layanan membuka kunci, dan berhasil mengaktifkan opsi FaceUnlock. Saat opsi dipilih, aplikasi penyerang dipanggil dan akibatnya ditetapkan sebagai kunci telepon. Saat pengguna ingin membuka kunci layar, aplikasi penyerang menggunakan tindakan com.android.internal.policy.IfaceLockInterface untuk menipu Keyguard agar mengikat ke layanannya. Akibatnya, malware memperoleh kendali penuh atas proses membuka kunci layar dan mampu mengekspos perangkat ke siapa pun yang diinginkannya. Serangan ini menimbulkan ancaman yang sangat serius terhadap kerangka multipengguna yang disediakan oleh Google dari Android 4.2, di mana penyerang dengan sengaja memasang aplikasi Facelock berbahaya sebagai pintu belakang ke akun pengguna lain. Faktanya, setelah dipasang di akun pengguna jahat, aplikasi tersebut akan segera diaktifkan di akun pengguna lain seperti yang dibahas dalam penelitian sebelumnya (Du et al., (2014)). Perhatikan bahwa

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

meskipun Lollipop dan versi yang lebih baru tidak lagi menawarkan FaceUnlock, dan sebagai gantinya mendorong dukungan untuk fungsionalitas tersebut ke produsen perangkat, kelemahan keamanan ini masih berdampak signifikan, mengingat fakta bahwa sekitar 90% perangkat di pasar menjalankan versi di bawah 5.0.

Hantu di Galaksi. Samsung Task Manager adalah aplikasi sistem yang menawarkan manajemen memori yang nyaman bagi pengguna. Melalui layanan ini, seseorang dapat memantau aplikasi mana yang sedang berjalan di perangkat. Dalam penelitian ini, Task Manager dianalisis dan ditemukan bahwa itu tidak menampilkan aplikasi pada daftar putih. Menariknya, aplikasi khusus tersebut diidentifikasi dari nama paketnya dan tidak ada verifikasi tanda tangan untuk memeriksa keasliannya. Juga, banyak dari mereka yang hilang di berbagai perangkat. Konsekuensinya di sini adalah bahwa musuh dapat membangun malware yang mengeksploitasi referensi gantung tersebut, menggunakan nama aplikasi resmi untuk memastikan bahwa aplikasinya tidak akan menarik perhatian saat beroperasi, misalnya, merekam percakapan telepon di latar belakang. Beberapa aplikasi hantu diterapkan di Samsung Note 8.0 dan menghilangkannya dari Task manager. Sekali lagi aplikasi serangan yang dipasang secara sengaja ini menyamar sebagai pencatat, melewati pemeriksaan keamanan Appstore.

Memalsukan Dropbox di LG. LG FileManager adalah aplikasi sistem pada perangkat LG yang membantu pengguna mengelola sistem filenya. Ini juga mendukung penggunaan Dropbox, yang dapat dibuka dengan mengklik tombol dengan ikon Dropbox. Menariknya, pada gambar pabrik LG G3, penganalisa pada penelitian ini menemukan bahwa tombol sebenarnya pertama kali mencoba meluncurkan aktivitas dalam com.vcast.manager, aplikasi cloud Verizon, dan hanya masuk ke halaman login web Dropbox setelah upaya gagal. Logika program ini dapat dirancang untuk perangkat yang didistribusikan oleh Verizon tetapi membiarkan referensi ke layanan tergantung pada perangkat dengan operator lain dan ponsel pengembangan. Dalam penelitian ini, aplikasi serangan dibuat untuk menyamar sebagai com.vcast.manager dan membajak aktivitas yang ditunjukkan oleh referensi gantung. Karena LG FileManager tidak memeriksa tanda tangan aplikasi target sebelum memulai aktivitasnya, LG FileManager secara membabi buta menjalankan aplikasi ini setiap kali pengguna mengklik tombol "Dropbox". Ini memberi aplikasi kesempatan untuk menampilkan aktivitas masuk Dropbox palsu untuk mencuri kredensial pengguna.

Mengganti perekam resmi. S-Voice melakukan perekaman suara menggunakan perekam default. Ada dua perekam seperti itu, com.sec.android.app.voicerecorder dan com.sec.android.app.voicenote. Apa yang terjadi adalah S-Voice pertama kali mencoba menggunakan aktivitas perekam suara dan hanya jika ini gagal (aplikasi tidak ada), ia beralih ke catatan suara. Sekali lagi, proses dua-pilih-satu seperti itu tidak melibatkan otentikasi target yang tepat. Ini memungkinkan pembuatan aplikasi penyerang yang meniru aplikasi perekam suara dengan aktivitas VoiceRecorderMain Activity untuk mengontrol target referensi. Percobaan pada Note 8.0, menunjukkan bahwa aktivitas penyerang selalu dipanggil, bahkan dengan adanya voicenote, yang memungkinkannya merekam percakapan pengguna yang sensitif atau melakukan serangan phishing.

Hulu berjaga-jaga. WatchON adalah aplikasi populer yang memungkinkan penggunanya untuk melihat program TV di TV mereka atau memilih film dari layanan Video-on-Demand yang mengintegrasikan Hulu, Vudu, popcornflix, dll. Setelah pengguna mengklik film Hulu, WatchON mengirim pesan Maksud implisit untuk meluncurkan aktivitas Hulu. Untuk beberapa film yang memerlukan akun HuluPlus, pengguna akan dialihkan ke aktivitas peningkatan di mana dia dapat membayar untuk ditingkatkan ke status HuluPlus. Masalahnya di sini adalah bahwa referensi ke aktivitas Hulu ditemukan menggantung dalam penelitian ini: meskipun WatchON memang memeriksa apakah Hulu ada sebelum mengirimkan Intent implisit, WatchON gagal memverifikasi tanda tangan aplikasi. Oleh karena itu, dalam penelitian ini dapat membuat aplikasi jahat yang menyamar sebagai Hulu dan menyetel filter Intent dengan aksi hulu.intent.action.LAUNCH_VIDEO_ID untuk mendapatkan Intent pemutakhiran. Melalui peluncuran aktivitas jahat, ini dapat menipu pengguna untuk memasukkan kredensial masuknya untuk Hulu. Lebih serius lagi, ketika dia benar-benar mengklik film berbayar, malware tersebut menampilkan aktivitas peningkatan, meminta informasi kartu kreditnya. Karena semua aktivitas ini dipicu oleh WatchON, malware tersebut kemungkinan besar mendapatkan apa yang diinginkannya.

Tangkap Penyedia Konten

Sama seperti tindakan dan aktivitas, penyedia konten juga banyak digunakan untuk interaksi antar aplikasi dan kerangka kerja aplikasi. Secara khusus, aplikasi dapat mengkueri penyedia konten aplikasi lain dengan merujuk langsung ke otoritasnya, satu atau beberapa URI yang diformat dalam konvensi penamaan

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

gaya Java: mis., com.example.provider.imageprovider. Namun, seperti yang terjadi pada atribut lainnya, referensi semacam itu (ke otoritas) juga dapat menjadi hang, saat penyedia terkait tidak ada di perangkat. Ini membuka jalan lain untuk eksploitasi Hare, ketika aplikasi jahat secara strategis menentukan penyedia konten untuk memberikan informasi yang salah kepada penanya. Perhatikan bahwa tidak seperti nama paket, duplikat nama otoritas tidak dilarang di Play Store. Hasilnya, semua aplikasi serangan ini berhasil diunggah ke Google Play.

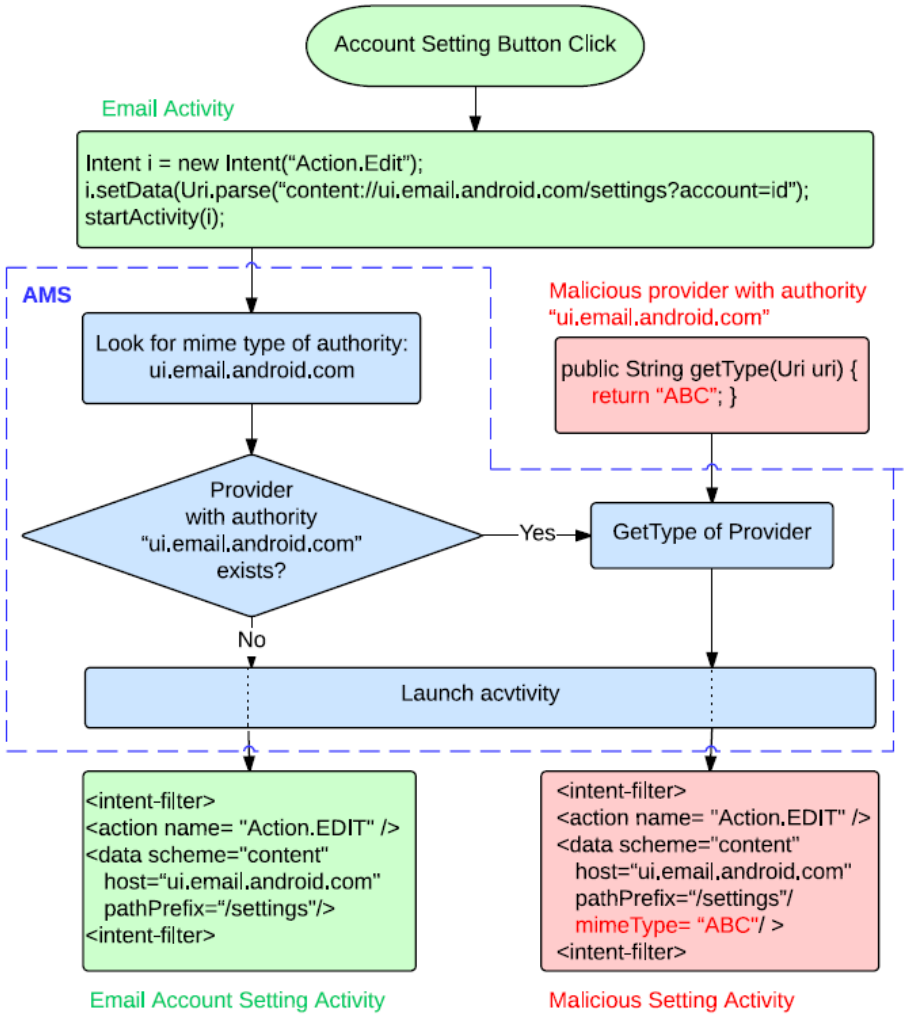
Pemanggilan Intent Pembajakan

Temuan mengejutkan dari penelitian ini bahwa penyedia konten halus Hare di dalam Google Email (versi 6.3-1218562) memungkinkan aplikasi berbahaya untuk sepenuhnya mengganti pengaturan akun internalnya dengan aktivitas berbahaya. Khususnya, Google Email, aplikasi email standar di setiap ponsel Google, memungkinkan pengguna mengonfigurasi berbagai akun email (Gmail, pertukaran, dll.) melalui antarmuka Setelan. Untuk memanggil aktivitas ini, aplikasi mengirimkan Intent implisit dengan tindakan android.intent.action.EDIT dan konten data://ui.email android.com/settings?account=x, di mana x adalah ID akun email yang digunakan untuk menginformasikan aktivitas pengaturan akun pengaturan email mana yang akan diedit. Kedua parameter ini ditentukan dalam filter Intent aktivitas setelan akun, seperti yang diilustrasikan dalam cuplikan kode berikut:

```
<!-- Account Settings Intent Filters-->
<activity
android:name=".activity.setup.AccountSettings" android:exported="true">
<intent-filter>
<action android:name="android.intent.action.EDIT"/>
<category android:name= "android.intent.category.DEFAULT"/>
<data android:scheme="content"
android:host="ui.email.android.com"
android:pathPrefix="/settings"/>
</intent-filter>
```

Intent implisit ini dapat diterima oleh aplikasi apa pun yang menentukan filter Intent di atas untuk aktivitasnya. Namun, saat ini terjadi, Android akan memunculkan jendela yang mencantumkan semua penerima yang memenuhi syarat untuk memungkinkan pengguna memilih. Sehingga perlindungan dapat dihindari dan, menjadikan aplikasi berbahaya sebagai satu-satunya penerima yang memenuhi syarat. Untuk tujuan ini, bagian data filter Intent dalam cuplikan kode di atas dianalisis dan memeriksa bagaimana wakil ActivityManagerSer (singkatnya AMS) menyelesaikan Intent yang dikirim ke filter Intent ini. Gambar 1 menggambarkan langkah-langkah resolusi Intent dalam skenario ini. Jika skema data adalah konten, AMS akan mencoba menyimpulkan MIME (Multi-Purpose Internet Mail Extension) dari data terlampir untuk mengidentifikasi penerima yang dapat menangani tipe ini: tipe data di sini seharusnya diberikan oleh penyedia konten ui .email.android.com. Namun, penyedia ini tidak ada dan sebagai akibatnya, tipe tersebut biasanya diabaikan dan Intent dikirim ke siapa pun yang menentukan tindakan.EDIT dan filter data (dengan schema="content") tanpa tipe MIME yang ditentukan (sebagai Tidak ada cabang di Gambar 1).

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



Gambar 1.: Memanfaatkan Hare Authority untuk Membajak Aktivitas Pengaturan Akun Email

Risiko keamanan di sini adalah referensi ke penyedia konten macet dan dapat dieksploitasi oleh aplikasi jahat yang menentukan penyedia tersebut. Apa yang dapat dilakukan malware adalah menamai otoritas penyedia ui.email.android.com untuk menerima kueri dari AMS (cabang Yes pada Gambar 1), mengembalikan jenis MIME pilihannya sendiri untuk memberikan informasi yang salah, dan sementara itu tentukan jenis ini dalam filter Intent aktivitasnya sendiri, menjadikannya satu-satunya aplikasi yang memenuhi syarat untuk mendapatkan Intent (untuk menjalankan aktivitas pengaturan akun). Dalam penelitian ini, aplikasi serangan yang dibuat mengambil alih penyedia konten dan merespons kueri dari AMS dengan tipe MIME vnd.android.cursor.dir/vnd.example.ABC. Selain itu, penyerang menentukan filter Intent seperti yang diilustrasikan dalam cuplikan kode berikutnya, dengan mengklaim mineType dengan jenis yang diberitahukan kepada AMS.

```
<!-- Malicious Setting Activity Intent Filters-->
<activity android:name=".MaliciousSetting">
<intent-filter>
<action android:name= "android.intent.action.EDIT"/>
<category android:name= "android.intent.category.DEFAULT"/>
<data android:scheme="content"
android:host="ui.email.android.com"
```

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

```
android:pathPrefix="/settings"
android:mimeType= "vnd.android.cursor.dir/vnd.example.ABC"/>
</intent-filter>
```

Dengan cara ini, Intent dari aplikasi hanya mengarah ke malware, mengarahkan pengguna ke aktivitas jahat yang memungkinkannya memasukkan kata sandinya.

Tango dalam kegelapan. Tango adalah aplikasi perpesanan lintas platform yang populer, menawarkan audio, panggilan video melalui jaringan 3G, 4G, dan Wi-Fi. Aplikasi ini telah diinstal lebih dari 100 juta kali dari Google Play. Untuk menampilkan pesan SMS yang diterima, set up Intent filter dengan action android.provider.Telephony.SMS_RECEIVED untuk mendapatkan Intent yang membawa pesan dari Telephony Manager. Saat pengguna mengirim pesan melalui Tango, aplikasi menyimpannya ke sms, penyedia konten telepon. Di perangkat tanpa Telepon, referensi Tango ke penyedia kontennya menjadi menggantung. Oleh karena itu, aplikasi jahat dapat menentukan penyedia konten menggunakan sms otoritas untuk mendapatkan pesan SMS yang dikirim pengguna. Hal ini dapat terjadi saat malware pertama kali mengirim pesan, menyebabkan pengguna yang tidak sengaja membalas. Apa yang dapat dimanfaatkan di sini adalah kerentanan lain di Tango: aplikasi tidak melindungi penerima SMS dengan izin sistem android.permission.broadcast_sms, seperti yang seharusnya dilakukan. Ini memungkinkan pihak mana pun menyiarkan aksi SMS_RECEIVED untuk menyuntikkan pesan singkat palsu ke dalam aplikasi. Dalam penelitian ini, serangan diterapkan pada Tab S 8.4, mengirimkan pesan palsu ke Tango dan menerima respons pengguna menggunakan penyedia konten jahat.

Penipuan LG CloudHub. LG CloudHub adalah aplikasi sistem yang memungkinkan pengelolaan akun cloud, mengunggah data ke cloud, dan mengaksesnya dari berbagai perangkat. Secara default, aplikasi mendukung Dropbox dan Box, dan di berbagai perangkat juga dapat menghubungkan pengguna ke layanan lain, termasuk penyedia cloud LG. Informasi tentang layanan tambahan ini disimpan di penyedia konten com.lge.lgaccount.provider, yang dicari oleh LG CloudHub setiap kali dipanggil. Menariknya, pada beberapa ponsel, penyedia ini tidak ada. Contoh yang menonjol adalah LG G3. Saat ini terjadi, LG CloudHub hanya menampilkan layanan default, Dropbox dan Box. Namun, hal ini menjadikan referensi ke penyedia konten sebagai kasus Hare dan memaparkannya pada manipulasi aplikasi jahat. Secara khusus, aplikasi serangan yang mendefinisikan com.lge.lgaccount.provider diterapkan dan ditempatkan di entri untuk akun LG Cloud di penyedia konten. Akun ini kemudian ditampilkan di daftar akun yang tersedia LG CloudHub. Setelah diklik oleh pengguna, aplikasi mengirimkan Intent implisit dengan aksi com.lge.lgaccount.action.ADD_ACCOUNT. Di perangkat (G3), tidak ada aplikasi pra-instal yang menentukan tindakan, yang memungkinkan malware menentukan tindakan, mengklaim bahwa itu dapat menangani Maksud. Konsekuensinya adalah klik pengguna pada aplikasi sistem (LG CloudHub) memicu aktivitas berbahaya yang menyamar sebagai halaman login untuk akun LG Cloud, yang digunakan untuk menipu pengguna agar mengungkap kata sandi dan kredensial lainnya.

Penyitaan Izin

Cacat Hare juga dapat disebabkan oleh izin, yang ditentukan oleh aplikasi sistem dan digunakan untuk mengontrol akses ke berbagai sistem atau sumber daya yang ditentukan aplikasi (mis., penyedia konten, penerima broadcast, dll. .). Selama proses penyesuaian OS, aplikasi yang menentukan izin ("pemilik" aslinya) dapat dihapus. Sementara itu, jika sumber daya yang dilindungi oleh izin ini masih ada, penggunaan izin (untuk perlindungan) menjadi referensi yang menggantung. Untuk mengeksploitasi kelemahan seperti itu, musuh dapat dengan mudah menentukan izin yang hilang namun masih digunakan untuk mendapatkan akses ke sumber daya yang mereka lindungi. Masalah ini juga ditemukan luas dalam penelitian ini, dan muncul di semua 97 gambar pabrik yang dipindai. Membuat ancaman ini sangat berbahaya adalah kenyataan bahwa Google Play tidak memeriksa izin ganda: semua aplikasi serangan yang dilakukan disini berhasil diunggah di sana.

Mendapatkan kontak dari S-Voice

Aplikasi sistem S-Voice menyertakan penyedia konten (com.vlingo.midas.contacts.penyedia konten) yang menyimpan informasi tentang kontak pengguna, termasuk nama, alamat email, nomor telepon, alamat rumah, dll. Akses ke penyedia adalah dijaga oleh sepasang perizinan com.vlingo.midas .contacts.permission.READ (BACA singkatnya) dan com.vlingo.midas.contacts.permission.WRITE (WRITE). Namun, ditemukan bahwa mereka tidak ditentukan di Galaxy Note 3 (ponsel) dan Note 8.0 (tablet), yang membuka pintu untuk eksploitasi. Secara khusus, aplikasi serangan untuk kedua perangkat

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

dalam penelitian ini dibuat, yang menentukan izin BACA dan TULIS. Aplikasi ditemukan berhasil membaca semua data kontak dari S-Voice dan juga memperbarui datanya yang dikelola oleh penyedia konten sesuka hati, misalnya, mengubah alamat email, URL, dan nomor telepon kontak, yang dapat menyebabkan kebocoran informasi dan konsekuensi lainnya (misalnya, menyebabkan pengguna mengunjungi URL musuh yang ditempatkan di kontak temannya).

Tautan Retak

Tautan adalah aplikasi sistem yang memungkinkan penggunanya menyinkronkan datanya (file, gambar, audio, video, dll.) di berbagai perangkat (ponsel, tablet, laptop, dll.). Untuk tujuan ini, pada perangkat seluler (ponsel atau tablet), aplikasi menggunakan penyedia konten com.mfluent.asp.datamodel.ASPMediaStoreProvider untuk memelihara informasi tentang data tersebut, bersama dengan geolokasi pengguna. Penyedia ini dilindungi oleh com.mfluent.asp.permission.DB_READ _WRITE (disingkat DB_READ_WRITE). Namun, pada banyak image pabrik, disini tidak izin yang telah ditentukan justru tidak ditemukan. Akibatnya, perlindungan di sini menjadi menggantung. Aplikasi serangan dalam penelitian ini dibuat kembali yang pada nantinya akan menjadi penentu izin DB_READ _WRITE. Di Galaxy Note 3 dan Note 8.0, aplikasi ini berhasil memperoleh informasi sensitif dari penyedia konten, termasuk geolokasi pengguna, semua meta-data dokumen, file audio dan video (nama, jalur direktori, artis, genre, dll.). Juga, malware mampu mengubah meta-data.

Deteksi dan Pengukuran

Untuk lebih memahami Hares dan memitigasi risiko keamanan yang ditimbulkannya, serangkaian alat dalam penelitian ini dibangun termasuk Harehunter, penganalisa otomatis yang mendeteksi kelemahan Hare dari aplikasi pra-instal pada citra pabrik, dan HareGuard, aplikasi yang menangkap upaya untuk mengeksploitasi Rabbit yang dikenal pada perangkat. Dengan menggunakan Harehunter, studi pengukuran yang memeriksa 97 gambar OS pabrik juga dilakukan untuk perangkat populer seperti Galaxy S5, S6, Note 3, 4, 8.0, LG G3, Nexus 7, Moto X, dll. Studi ini mengungkap kemungkinan 21557 Hares di seluruh perangkat ini, yang menunjukkan meluasnya kerentanan yang kritis terhadap keamanan tersebut.

Harehunter

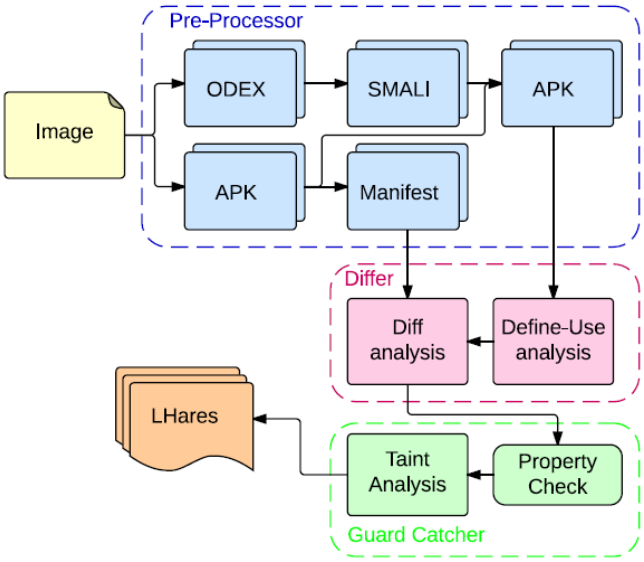
Harehunter dirancang untuk mengidentifikasi referensi yang menggantung di dalam aplikasi sistem dan dapat mencapai akurasi yang tinggi. Penelitian ini fokus pada aplikasi ini karena penelitian sebelumnya menunjukkan bahwa aplikasi pra-instal adalah komponen yang paling banyak disesuaikan secara intensif di berbagai perangkat Android (Jiang et al., (2013)), dan karenanya merupakan sumber kerentanan Hare yang paling mungkin. Analisis manual selanjutnya menunjukkan bahwa sebagian besar Hares memang berasal dari aplikasi sistem. Di sisi lain, layanan kerangka kerja juga dapat menyertakan referensi gantung, demikian juga aplikasi pihak ketiga (mis., Tango). Harehunter dapat langsung diterapkan untuk menemukan masalah di aplikasi pihak ketiga dan diperluas (dengan mengutak-atik langkah pra-pemrosesan) untuk bekerja di layanan Android.

Mendesain

Ide di balik desain ini sangatlah sederhana. Untuk setiap gambar pabrik, pertama-tama analisis diferensial dijalankan, yakni: mengekstraksi semua atribut (nama paket, tindakan, aktivitas, layanan, penyedia konten, dan izin) yang ditentukan oleh aplikasi pra-instalnya dan semua referensi ke atribut dalam kode dan manifesnya, dan kemudian membandingkan referensi dengan definisi. Perbedaan apa pun antara kedua ujung ini menunjukkan kemungkinan adanya Hares. Misalnya, jika nama paket digunakan untuk memulai aktivitas (startActivity) atau mengikat layanan (bindService) tetapi tidak dimiliki oleh aplikasi apa pun yang telah diinstal sebelumnya di perangkat, referensi ke sana kemungkinan akan hang. Di sisi lain, referensi semacam itu ternyata dijaga dengan baik: misalnya, sebelum merujuk ke paket, aplikasi sistem mungkin terlebih dahulu memeriksa keberadaannya, mengumpulkan informasi tanda tangannya (misalnya, getPackageInfo dengan flag GET_SIGNATURE) dan memverifikasinya dengan tanda tangan dari aplikasi asli. Untuk mendeteksi Hare yang benar-benar rentan, kode antara penjaga potensial (mis., fungsi untuk pemeriksaan tanda tangan) harus dianalisis dan referensi gantung yang mungkin (mis., startActivity) untuk mengetahui apakah mereka benar-benar terkait. Hanya referensi yang tidak terlindungi yang akan dilaporkan sebagai Hare. Untuk mengimplementasikan ide ini, sistem dirancang dengan tiga komponen utama, Pre-processor, Differ, dan Guard Catcher, seperti yang diilustrasikan pada Gambar 2: Pre-processor mengekstrak paket aplikasi dari citra OS dan mengubahnya menjadi bentuk yang dapat dianalisis oleh langkah tindak lanjut; Differ melakukan analisis diferensial dan melaporkan kemungkinan referensi

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

gantung; Catcher memeriksa APK yang melibatkan referensi tersebut untuk menentukan apakah referensi tersebut telah dilindungi.



Gambar2.: Desain Harehunter.

Pra-pemrosesan. Dari setiap gambar pabrik, Harehunter pertama-tama mengumpulkan semua aplikasi pra-instalnya, dalam bentuk file APK dan ODEX, dan menjalankan Apktool untuk mengekstrak file manifes setiap aplikasi dan Baksmali untuk mendekompilasi aplikasi menjadi kode Smali. Untuk beberapa perangkat, terutama dengan Samsung, file ODEX aplikasi sistem sering dipisahkan dari file APK-nya, untuk tujuan meningkatkan waktu pemuatannya, sementara Flowdroid, penganalisa statis tempat ini membangun sistem yang ada pada penelitian ini hanya berfungsi pada APK. Untuk mengatasi masalah ini, pra-prosesor diterapkan untuk mengekstrak file ODEX secara otomatis, mendekompilasinya, lalu mengkompilasi ulang dan mengompresnya, bersama dengan file sumber dayanya, menjadi file APK baru. Lebih lanjut memperumit proses ini adalah untuk Android 5.0 Lollipop, file ODEX diganti dengan file OAT, yang menyertakan kode asli. Untuk aplikasi dalam bentuk seperti itu, Harehunter pertama-tama membuka ritsleting file OAT-nya dan kemudian menjalankan oat2dex untuk mengubahnya menjadi format ODEX, memungkinkan proses di atas untuk bergerak maju. ***Analisis diferensial***. Untuk melakukan analisis diferensial, Differ pertama-tama mencari semua kode yang diekstrak, didekompilasi, dan file manifes untuk definisi atribut yang ditargetkan. Menjalankan parser XML, pendekatan ini dapat dengan mudah mengumpulkan paket yang ditentukan, tindakan, serta otoritas dan izin penyedia konten dari file manifes aplikasi individu. Perhatikan bahwa semua atribut ini, kecuali tindakan untuk menerima pesan broadcast, hanya dapat ditentukan dalam manifes. Meskipun tindakan yang digunakan dalam filter Intent untuk penerima broadcast dapat ditentukan secara terprogram, tindakan tersebut hanya berfungsi untuk mendapatkan pesan, bukan memanggil layanan atau aktivitas, dan oleh karena itu ketidakhadirannya tidak akan menyebabkan bahaya Hare.

Sebagian besar referensi untuk atribut ini ada di dalam kode, dalam bentuk berbagai panggilan API. Secara khusus, nama paket dan tindakan digunakan melalui startActivity, startActivityForResult, startService, dll. Nama otoritas penyedia konten muncul dalam berbagai operasi pada penyedia, seperti pembaruan, kueri, hapus, dan lainnya. Izin diklaim dalam manifes atau diverifikasi melalui checkPermission dan API lainnya. Untuk mengidentifikasi referensi ini, Differ pertama-tama mencari situs panggilan untuk semua fungsi terkait dari kode Jimple aplikasi dan kemudian melakukan analisis penggunaan-definisi dari setiap situs panggilan untuk memulihkan nama atribut yang ditargetkan menggunakan grafik aliran kontrol (CFG) yang dibangun oleh Flowdroid. Masalahnya di sini adalah Flowdroid tidak dapat membuat CFG lengkap, kehilangan beberapa titik masuk program seperti onHandleIntent. Dalam implementasi ini, sebanyak mungkin entri yang dapat temukan ditambahkan kembali, tetapi masih ada beberapa pemanggilan fungsi target yang CFG terkaitnya tidak dapat dibuat oleh Flowdroid. Untuk panggilan ini, prototipe saat ini hanya dapat menangani situasi di mana nama atribut di-hardcode dalam fungsi terkait.

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

Deteksi penjaga. Seperti disebutkan sebelumnya, referensi ke atribut yang hilang seringkali dilindungi. Ada dua cara dasar untuk proteksi tersebut, signature guard atau feature guard. Gambar 3 dan 4 menyajikan contoh untuk kedua kasus tersebut. Signature guard mencoba mendapatkan signature dari paket yang akan dipanggil, dan membandingkannya dengan yang diharapkan. Pada contoh (Gambar 3), pemeriksaan ini dilakukan dengan mengekstraksi tanda tangan com.facebook.katana melalui getPackageInfo dengan GET_SIGNATURES sebagai benderanya, lalu meminta compareSignature untuk membandingkannya dengan aplikasi Facebook yang sah, sebelum mengikat ke aplikasi target. layanan (bindService). Kehadiran paket otentik juga dapat memastikan kebenaran nama tindakan dan aktivitas. Cara lain untuk melindungi atribut ini adalah dengan memeriksa model build perangkat saat ini, karena hanya beberapa di antaranya yang memiliki fitur tertentu (dalam hal paket, penyedia konten, dan lainnya): misalnya, metode input, aplikasi email, semuanya bisa berbeda dari bangunan ke bangunan; Penyedia SMS/MMS bahkan mungkin tidak ada di tablet. Sebagai contoh, Gambar 4 menunjukkan bahwa aplikasi pertama kali menjalankan hasSystemFeature untuk memeriksa apakah perangkat saat ini mendukung Google TV (com.google.android.tv): jika demikian, aplikasi youtube.googletv akan dipanggil, dan sebaliknya, hanya YouTube.

Untuk mendeteksi perlindungan tersebut, Guard Catcher melakukan analisis taint melalui aliran data aplikasi dan aliran kontrol, menggunakan fungsionalitas yang disediakan oleh Flowdroid. Secara khusus, pendekatan ini pertama-tama mengidentifikasi serangkaian fungsi penjaga seperti hasSystemFeature dan getPackageInfo dengan parameter GET_SIGNATURES dan kemudian mencoba membuat hubungan di antara keduanya dan referensi gantung ditemukan oleh analisis diferensial, kondisi yang diperlukan untuk referensi ini harus dilindungi. Untuk tujuan ini, keluaran penjaga ini ditetapkan sebagai sumber taint dan referensi (misalnya, startActivity, bindService) diberi label sebagai taint sink. Flowdroid dijalankan untuk menentukan apakah noda dapat disebarkan dari yang pertama ke yang terakhir. Untuk bak cuci yang tidak dapat dinodai, kemungkinan besar dilaporkan sebagai Hares. Menjalankan analisis taint penuh (melalui arus informasi eksplisit dan implisit) untuk setiap pasangan penjaga dan referensi bisa sangat lambat. Untuk membuat deteksi penjaga lebih terukur, Catcher menggunakan strategi hibrid multi-langkah, menggabungkan pemeriksaan properti cepat dengan analisis noda. Secara khusus, pertama-tama memeriksa apakah sumber dan sink yang sesuai berada dalam metode yang sama. Ketika ini terjadi, dalam sebagian besar kasus, mereka terkait dan oleh karena itu referensi tersebut dianggap dilindungi. Jika tidak, pendekatan selanjutnya membandingkan nama paket yang terlibat dalam pemeriksaan tanda tangan dengan yang digunakan untuk referensi. Kecocokan yang ditemukan di antara pasangan hampir selalu menunjukkan hubungan perlindungan. Contohnya adalah com.facebook.katana di dalam cuplikan kode di Gambar 3 yang muncul di dalam getPackageInfo dan setClassName. Hanya ketika kedua pemeriksaan gagal, analisis noda kelas berat akan digunakan. Dalam analisis gambar pabrik skala besar penelitian ini, ditemukan bahwa sebagian besar waktu, pelindung untuk referensi dapat ditemukan dalam dua langkah pertama.

Evaluasi

Evaluasi keefektifan penerapan dalam studi pengukuran ini dilakukan, melibatkan citra OS untuk 97 perangkat populer, semuanya bersama-sama di lebih dari 24.000 aplikasi sistem. Harehunter melaporkan 21557 kemungkinan Hares. Dari semua Hares ini, sampel sejumlah 250 diambil secara acak dan menganalisis kodenya secara manual. Hanya 37, yaitu 14%, ditemukan sebagai deteksi palsu: yaitu, salah memperlakukan referensi yang dijaga sebagai Rabbit. Selanjutnya tingkat negatif palsu dari Guard Catcher diukur dengan memeriksa secara acak kemungkinan referensi gantung yang dilaporkan oleh Differ dan membandingkan temuan dengan apa yang terdeteksi oleh Catcher. Dalam semua 250 sampel, 46 (19%) terlewatkan oleh implementasi ini: yaitu, Hares sejati dianggap dijaga secara salah. Melihat positif dan negatif palsu tersebut, ditemukan bahwa semuanya disebabkan oleh keluaran grafik panggilan yang tidak lengkap oleh FlowDroid. Flowdroid diketahui bermasalah dalam menangani ICC (Octeau et al., (2014)) dan masalah lain seperti titik masuk yang hilang dan grafik panggilan yang tidak lengkap. Ketika ini terjadi, analisis noda tidak dapat dilakukan.

HASIL

Pengukuran Skala Besar

Untuk memahami ruang lingkup dan besarnya bahaya keamanan yang disebabkan oleh Hares, studi pengukuran berskala besar pada 97 citra pabrik dilakukan dalam penelitian ini. Studi tersebut menunjukkan bahwa Hares memang tersebar luas, dengan dampak signifikan pada ekosistem Android:

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

lebih dari 21557 LHares ditemukan dan banyak di antaranya dapat menyebabkan konsekuensi seperti pembajakan aktivitas, kebocoran data, dan polusi.

Koleksi Gambar OS

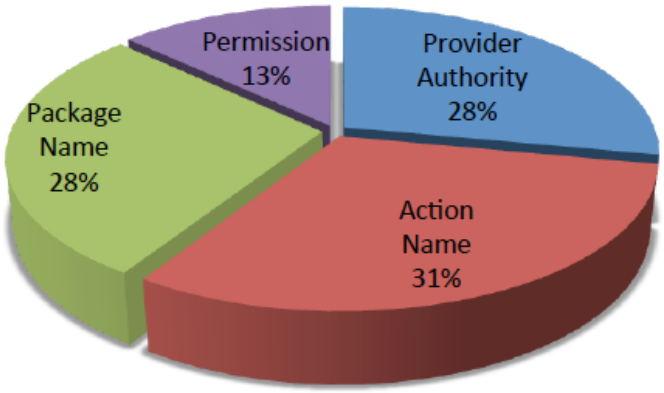
Dalam penelitian ini sejumlah 97 factory image ditemukan dari Samsung Update, Android Revolution dan perangkat fisik, yang mencakup sekitar 183 aplikasi per gambar dan 24185 aplikasi secara keseluruhan. Gambar ini disesuaikan untuk 49 model ponsel atau tablet berbeda, 36 negara, dan 23 operator berbeda. Mereka mengoperasikan versi Android dari 4.0.3 hingga 5.0.2.

Lanskap

Saat menganalisis gambar pabrik tersebut, ditemukan bahwa sekitar 13% dari aplikasi pra-instal mereka tidak dapat didekompilasi oleh Apktool atau dianalisis oleh Flowdroid. Di antara yang dapat dianalisis, Harehunter menemukan bersama-sama 21557 kelemahan (referensi gantung yang tidak dijaga) dalam 3450 aplikasi yang rentan. Perhatikan bahwa beberapa kelemahan ini mungkin terjadi lebih dari satu kali dalam aplikasi yang sama, dan beberapa aplikasi yang rentan muncul di beberapa perangkat. Riset ini mengungkapkan bahwa setiap gambar mengandung sejumlah besar kelemahan Hare, mulai dari 8 hingga 598. Rata-rata, 14,3% aplikasi prainstal di 4.X dan 11,7% di 5.X ternyata rentan. Juga seperti yang dapat dilihat dari tabel diatas, masalahnya juga meluas di berbagai produsen perangkat: baik Vendor A dan C memiliki porsi signifikan dari aplikasi sistem mereka yang melibatkan referensi gantung. Sebagai perbandingan, Vendor D memiliki jumlah kekurangan terkecil (29) dan rasio aplikasi rusak terendah (8%). Alasan yang mungkin adalah bahwa gambar OS yang dijalankan perangkatnya adalah yang paling tidak disesuaikan, yang meminimalkan kemungkinan untuk memperkenalkan Hares. Gambar 6 mengilustrasikan distribusi cacat pada berbagai kategori Rabbit. Sebagian besar masalah berasal dari nama tindakan yang tidak ditentukan. Sebagai perbandingan, persentase izin yang relatif rendah ditemukan terlibat dalam menggantung referensi.

Dampak Hares

Dampak Hares sangat signifikan. Selain serangan end-to-end yang dibuat dalam penelitian ini, sejumlah 33 sampel kelemahan juga diambil secara acak dan dianalisis secara manual apa yang dapat terjadi setelah dieksploitasi. Perhatikan bahwa karena kurangnya sejumlah besar perangkat fisik, yang dapat dilakukan hanyalah analisis statis untuk menyimpulkan kemungkinan konsekuensi setelah eksploit berhasil. Analisis semacam ini mungkin tidak akurat, tetapi tetap penting untuk memahami dampak dari kelemahan keamanan jenis ini yang belum pernah diketahui sebelumnya. Hasil analisis ini dapat dilihat pada Tabel 3.



Gambar 5: Distribusi Rabbit di Berbagai Kategori Rabbit

Seperti yang bisa dilihat di sini, 5 contoh Hares yang dipilih secara acak mungkin dieksploitasi untuk meluncurkan serangan Phishing serupa, karena nama paket dan aktivitas yang tidak ditentukan dan/atau nama tindakan untuk filter Intent aktivitas. Satu Hare yang ditemukan di Aplikasi Tugas HTC memungkinkan pengalihan Maksud melalui eksploitasi penyedia konten tidak terdefinisi yang digunakan untuk resolusi Maksud, seperti halnya serangan GoogleEmail. 4 Hares (pada perangkat seperti Note 8.0 dan S5) dapat menyebabkan kebocoran konten (catatan dan bookmark browser) setelah malware menyamar sebagai penyedia konten yang tidak ditentukan, tempat aplikasi korban memasukkan data ke dalamnya. 4

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

contoh mungkin mengungkap informasi pribadi pengguna ketika nama paket gantung dibajak. Khususnya, ditemukan bahwa pada Note 8.0, referensi gantung melibatkan Intent eksplisit yang dikirimkan ke paket yang tidak ada. Intent menyertakan URI konten yang mengarah ke data pribadi (misalnya, foto) dan juga izin FLAG_GRANT_URI_PERMIS SION yang memungkinkan penerima membaca data tanpa meminta izin. Akibatnya, aplikasi tidak sah yang menggunakan nama paket target dapat memperoleh akses ke data.

Selain itu, pada LG G3, referensi gantung ke penyedia konten yang tidak ada mungkin membuka pintu bagi musuh untuk menentukan penyedia tersebut untuk mencemari data yang disinkronkan ke perangkat pengguna lainnya. Selanjutnya, analisis ini mengungkapkan 3 contoh yang dapat menyebabkan serangan denial-of-service ketika musuh membuat penyedia konten yang tidak terdefinisi yang digunakan oleh aplikasi korban, dan menyetel bendera yang diekspor ke false. Dari kode aplikasi, serangan ini dapat menyebabkan pengecualian keamanan saat aplikasi korban mencoba membaca atau menulis ke penyedia ini. Beberapa Rabbit lain dapat menyebabkan situasi yang tidak terduga: misalnya, aplikasi dengan nama paket tertentu tidak akan muncul di Task Manager sistem dan aplikasi lain di LG G3 tidak dapat dipaksa untuk berhenti dari aplikasi Pengaturan LG. Juga ditemukan bahwa Hares di 3 aplikasi mungkin hanya menampilkan dialog atau notifikasi. Juga, ada 6 Rabbit terkait dengan layanan yang hilang yang fungsinya tidak dapat diketahui. Terakhir, titik masuk untuk 4 Hares tidak ditemukan yang bisa jadi merupakan kode mati.

Pihak yang bertanggung jawab

Untuk tujuan ini, 6 gambar dari Vendor A yang semuanya menjalankan 4.4.2 diperiksa, seperti yang dijelaskan pada Tabel 4.4. Persentase cacat Hare yang secara unik diperkenalkan oleh model ini berkisar antara 9% hingga 29%. Selanjutnya gambar dikelompokkan ke dalam subgrup dan memeriksa mana yang menunjukkan persentase kasus Hare umum tertinggi. Model tablet memiliki persentase Rabbit umum tertinggi 63%, sedangkan model ponsel memiliki Rabbit umum tertinggi kedua sebesar 56%. Kasing Hare yang umum antara model perangkat tablet dan ponsel paling banyak 38%. Jadi menyesuaikan OS ke model tablet atau ke model ponsel memperkenalkan banyak Hares. Sementara itu, kelemahan yang ditemukan pada model yang sama (Ponsel 3 yang menjalankan Android 4.4.2) yang disesuaikan untuk operator yang berbeda juga dibandingkan dan hasilnya ada pada Tabel 5.

Tabel 3: Kemungkinan Dampak dari 33 Rabbit yang Dipetik Secara Acak

Impact	Hare Category	# of Hares
Activity Hijacking	Package and Activity Name	3
Activity Hijacking	Action Name	2
Activity Hijacking	Provider Authority	1
Data Leakage	Provider Authority	4
Data Leakage	Package and Activity Name	1
Data Pollution	Provider Authority	1
D.O.S.	Provider Authority	3
Dialog Popup	Action Name of Activities	3
Others	Package Name	5
Impact Not Clear	Action Name of Services	6
Maybe Dead Code	All Categories	4

Tabel 4: Cacat Hare pada Berbagai Model Vendor A yang Menjalankan Android 4.4.2

Model	# of New Hares Introduced by Model
Tablet 1	106 (27%)
Phone 2	35 (21%)
Phone 3	75 (29%)
Tablet 4	57 (22%)
Tablet 5	22 (9%)
Tablet 6	72 (20%)

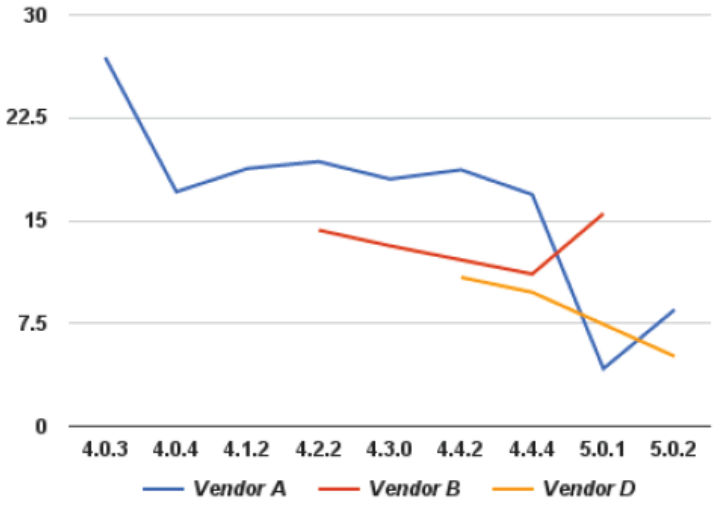
Seperti yang dapat dilihat dari Tabel 5 yang diberikan gambar Telepon 3, penyesuaiannya pada 6 operator menghasilkan sekitar 3% hingga 20% kekurangan. Jelas, baik pabrikan maupun operator

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

menyebabkan kelemahan Hare. Namun, yang pertama tampaknya harus lebih bertanggung jawab daripada yang terakhir. Selain itu, sebagian besar Rabbit kemungkinan besar akan diperkenalkan selama penyesuaian OS untuk model perangkat yang berbeda (ponsel atau tablet).

Kecenderungan

Gambar 6 lebih lanjut membandingkan rasio aplikasi yang rentan terhadap versi OS yang berbeda di beberapa produsen. Untuk perangkat Vendor A, ada kecenderungan yang dapat diamati bahwa versi yang lebih tinggi (5.0.1 dan 5.0.2) mengandung lebih sedikit Hares daripada yang lebih rendah: rasio yang salah berasal dari 26% pada 4.0.3 turun menjadi sekitar 8,2% pada 5.0. 2. Di sisi lain, untuk ponsel Vendor B, trennya hampir konstan: rasionya 14,3% pada 4.2.2 dan 15,1% pada 5.0.1 . Selain itu, pada semua perangkat ini, risiko Hare tetap signifikan, yang menunjukkan bahwa produsen belum menyadari tingkat kerentanan jenis ini.



Gambar 6: Rasio Aplikasi Rentan

Perlindungan tingkat aplikasi

Motivasi dan ide

Pada dasarnya, kelemahan Hare hanya dapat diperbaiki oleh produsen perangkat dan pengembang aplikasi, yang seharusnya menghapus referensi yang menggantung dalam kode mereka atau melakukan pemeriksaan keamanan yang tepat. Namun, mengingat meluasnya masalah dan akar penyebabnya, yaitu ekosistem Android yang kurang diatur. Sebelum solusi lengkap mereka dapat diimplementasikan, penting untuk membantu setiap pengguna Android melindungi sistem mereka, jika ada kekurangan ini. Dibandingkan dengan perlindungan lapisan kerangka kerja, yang hanya dapat diterapkan oleh produsen dan operator, solusi paling praktis adalah pertahanan tingkat aplikasi, karena yang perlu dilakukan pengguna hanyalah memasang aplikasi pelindung dari Google Play untuk mendapatkan perlindungan langsung terhadap ancaman terhadap kerentanan pada sistemnya. Ditemukan bahwa ini sebenarnya dapat dengan mudah dilakukan. Dalam penelitian ini, perlindungan sederhana dikembangkan menggunakan aplikasi, yang disebut HareGuard, untuk memindai aplikasi pihak ketiga lainnya setiap kali dipasang untuk memastikan bahwa mereka tidak memanfaatkan kerentanan Hare yang diketahui pada model perangkat tertentu. HareGuard mengumpulkan informasi model perangkat dan menanyakan database sisi server untuk mendapatkan semua Hares di dalam model (yang terdeteksi secara offline, misalnya, melalui Harehunter). Setiap kali aplikasi dipasang, HareGuard segera memeriksa file manifesnya untuk nama paket, aktivitas, tindakan, nama otoritas, dan izin yang ditentukannya, memastikan bahwa aplikasi tidak bermaksud membajak atribut yang hilang. Aplikasi pemindai ini dipanggil melalui startForeground, berjalan dengan notifikasi yang diposting di Notification Center.

Penerapan

Khususnya, segera setelah HareGuard diinstal, ia memanggil kelas Build untuk mengumpulkan informasi perangkat, termasuk Build.MANUFACTURER dan Build.MODEL, dan mengkueri database penelitan ini untuk semua kelemahan Hare pada perangkat. Pemindai juga menggunakan penerima Intent

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

dengan tindakan android.intent.action.PACKAGE_ADDED untuk memantau aplikasi baru yang diinstal dan android.intent.action.PACKAGE_CHANGED untuk mendeteksi apakah aplikasi diperbarui. Untuk setiap aplikasi baru atau yang baru saja diperbarui, ia menggunakan API openXmlResourceParser untuk membuka file manifesnya dan mengidentifikasi semua atribut yang ditentukannya. Atribut ini kemudian dibandingkan dengan sekumpulan referensi gantung yang diambil dari database Hare untuk mendeteksi risiko Hare: yaitu, menentukan atribut yang terkait dengan referensi gantung. Setelah risiko ditemukan, HareGuard memberi tahu pengguna, menjelaskan potensi bahaya keamanan kepadanya dan mendesaknya untuk memastikan bahwa aplikasi tersebut benar-benar berasal dari sumber yang dapat dipercaya atau hapus saja. Untuk membantu pengguna dalam proses ini, pemindai dapat membandingkan tanda tangan aplikasi dengan tanda tangan milik pihak yang berwenang, kapan pun ada di database. HareGuard diterapkan dalam penelitian ini, menggunakan database yang mendokumentasikan temuan yang dibuat oleh Harehunter saat memindai citra pabrik untuk perangkat seluler populer.

KESIMPULAN

Penerapan HareGuard terbukti efektif dalam mendeteksi semua aplikasi serangan yang dibuat secara sengaja. Evaluasi lebih lanjut terhadap kinerja juga dilakukan dan berdampak pada inangsistem minimum: pemindai ditemukan hanya menggunakan memori 4,29 MB dan mengonsumsi 0,29% CPU saat memindai manifes aplikasi. Hares bukan hanya beberapa bug acak yang terisolasi yang diperkenalkan oleh penyimpangan implementasi. Kehadiran kekurangan tersebut menyiratkan kelemahan dalam filosofi desain Android dan ekosistemnya. Pada dasarnya, Android adalah sistem yang kompleks, yang komponen dan aplikasinya dimaksudkan untuk bekerja bersama, yang menyebabkan hubungan saling ketergantungan yang sangat rumit di antara mereka. Sementara itu, ekosistem Android dikenal sangat beragam dan terdesentralisasi: setiap versi OS dikustomisasi dan dikustomisasi ulang oleh berbagai pihak hampir secara independen dan digunakan oleh siapa saja yang dapat membuat aplikasi untuk versi tersebut; sejauh ini sedikit panduan yang telah diberikan untuk membantu mengatur penyesuaian dan pengembangan aplikasi, memastikan bahwa mereka menghormati hubungan rumit yang ada antara komponen sistem dan aplikasi yang diperkenalkan oleh mereka sendiri dan pihak lain (AOSP, produsen, operator, pengembang aplikasi, dll). Dengan tidak adanya panduan seperti itu dan mekanisme penegakan yang tepat, referensi gantung menjadi tak terelakkan. Sebagaimana dibuktikan oleh penelitian ini, Hares tersebar luas, ada di setiap perangkat yang didiperiksa, dan juga memang sangat penting untuk keamanan, membahayakan data pengguna yang sensitif dan bahkan pelaksanaan yang tepat dari aplikasi sistem. Meskipun tidak semua masalah yang dilaporkan oleh Harehunter dapat dieksploitasi, yang bergantung pada kondisi untuk menjalankan kode yang rentan, meluasnya kode yang tidak terlindungi tersebut mengkhawatirkan: tanpa pemeriksaan mendalam terhadap kasus individual, tidak ada yang tahu apakah mereka dapat dieksploitasi dalam kondisi tertentu, memimpin untuk konsekuensi yang tidak terduga.

Untuk sepenuhnya menghilangkan risiko Hare, penting untuk memiliki hubungan interdependen yang terdokumentasi dengan baik dan membuatnya terbuka untuk pihak yang terlibat dalam penyesuaian OS dan pengembangan aplikasi. Selain itu, harus ada kebijakan yang mengharuskan siapa pun yang memodifikasi OS atau membuat aplikasi tidak boleh membuat hubungan yang menggantung seperti merujuk ke atribut yang tidak ada, dan mekanisme untuk pemeriksaan kepatuhan kebijakan. Penegakan kebijakan di sini dapat memanfaatkan program kompatibilitas Android yang ada, yang saat ini masih belum dapat melakukan pemeriksaan keamanan. Bagian yang menantang adalah kumpulan hubungan yang saling tergantung untuk semua versi Android yang dikenal. Informasi seperti itu belum ada. Penelitian ini menunjukkan bahwa pabrikan tampaknya tidak menyadari hubungan pada perangkatnya sendiri, sering kali merusaknya dan menyebabkan Hares saat menyesuaikan versi Android ke model yang berbeda. Alat yang sistematis, seperti Harehunter, diperlukan untuk mengidentifikasi informasi tersebut.

Sementara itu, upaya harus dilakukan untuk mengamankan setiap referensi atribut, yang terpenting di sini adalah otentikasi eksplisit sebelum referensi. Terlalu sering dilihat bahwa referensi hanya dilindungi secara implisit: misalnya, referensi ke aplikasi sistem diamankan dengan kehadiran aplikasi di perangkat, yang mengecualikan aplikasi lain yang menggunakan nama paket yang sama. Perlindungan semacam itu rapuh, benar-benar berantakan setelah aplikasi dihapus saat OS disesuaikan untuk model perangkat baru. Di sisi lain, pemeriksaan keamanan bisa lebih rumit dari yang terlihat. Lebih khusus lagi, meskipun referensi nama paket bisa langsung dijaga dengan tanda tangan cek. Atribut lain seperti penyedia konten, tindakan, dll. dapat langsung digunakan dan keberadaannya di perangkat tertentu sering kali diverifikasi dengan memeriksa model perangkat saat ini dan fitur lainnya. Kebenaran pemeriksaan semacam itu, sekali lagi, bergantung pada pengetahuan tentang hubungan komponen/aplikasi di berbagai versi, model, dll., yang perlu dipulihkan oleh Harehunter dan alat serupa lainnya.

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

Perlindungan sistem warisan

Bahkan sebelum bisa memikirkan cara melenyapkan Hares dalam mengembangkan sistem dan aplikasi di masa mendatang, masalah yang pertama-tama perlu diatasi adalah bagaimana mengamankan perangkat yang ada, yang, seperti ditunjukkan dalam penelitian ini, penuh dengan berbagai jenis kelemahan Hare. Teknik yang dikembangkan, Harehunter dan HareGuard, membuat langkah pertama untuk mengidentifikasi dan melindungi kerentanan ini. Khususnya, seperti yang disebutkan sebelumnya, Harehunter juga dapat memainkan peran penting dalam mengumpulkan hubungan saling ketergantungan untuk membantu mengamankan sistem dan aplikasi baru. Dengan potensinya yang besar, implementasi saat ini masih dalam tahap awal: ini memperkenalkan sekitar 14% positif palsu dan melewatkan 19% kasus yang benar-benar rentan dalam penelitian ini. Sebagian besar masalah disebabkan oleh Flowdroid, alat analisis statis yang mendukung sistem ini. Dapat dibayangkan bahwa Harehunter akan menjadi lebih efektif setelah penganalisa yang lebih mumpuni digunakan. Selain itu, untuk produsen perangkat yang memiliki kode sumber untuk semua layanan dan aplikasi sistem, alat yang mirip dengan Harehunter, tetapi bekerja dengan kode sumber, bisa lebih akurat dalam mendeteksi kelemahan Hare, dengan harapan bahwa arah ini akan dieksplorasi oleh akademisi dan industri dalam waktu dekat atau dimasa depan.

DAFTAR PUSTAKA

A. Al-Haiqi, M. Ismail, and R. Nordin, “On the best sensor for keystrokes inference attack on android,” in The 4th International Conference on Electrical Engineering and Informatics (ICEEI), Procedia Technology, 2013.

A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, “Android permissions demystified,” in Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS ’11, (New York, NY, USA), pp. 627–638, ACM, 2011.

B. Liu, J. Lin, and N. Sadeh, “Reconciling mobile app privacy and usability on smartphones: Could user privacy profiles help?,” in Proceedings of the 23rd International Conference on World Wide Web, WWW’14, (New York, NY, USA), pp. 201–212, ACM, 2014.

C. Lin, H. Li, X. Zhou, and X. Wang, “Screenmilker: How to milk your android screen for secrets,” in 21st Annual Network and Distributed System Security Symposium (NDSS), The Internet Society, 2014.

E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, “Tapprints: Your finger taps have fingerprints,” in Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services, MobiSys ’12, (New York, NY, USA) pp. 323–336, ACM, 2012.

F. Roesner and T. Kohno, “Securing embedded user interfaces: Android and beyond,” in Proceedings of the 22Nd USENIX Conference on Security, SEC’13, (Berkeley, CA, USA), pp. 97–112, USENIX Association, 2013.

F. Roesner, T. Kohno, A. Moshchuk, B. Parno, H. J. Wang, and C. Cowan, “User-driven access control: Rethinking permission granting in modern operating systems,” in Proceedings of the 2012 IEEE Symposium on Security and Privacy, 2012.

H. Huang, S. Zhu, K. Chen, and P. Liu, “From system services freezing to system server shutdown in android: All you need is a loop in an app,” in Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS ’15, (New York, NY, USA), pp. 1236–1247, ACM, 2015.

H. Zhang, D. She, and Z. Qian, “Android root and its providers: A double-edged sword,” in Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS ’15, (New York, NY, USA), pp. 1093–1104, ACM, 2015.

J. Caballero, G. Grieco, M. Marron, and A. Nappa, “Undangle: Early detection of dangling pointers in use-after-free and double-free vulnerabilities,” in Proceedings of the 2012 International Symposium on Software Testing and Analysis, ISSTA 2012, ACM, 2012.

K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, “Pscout: Analyzing the android permission specification,” in Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS ’12, (New York, NY, USA), pp. 217–228, ACM, 2012.

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

L. Li, A. Bartel, J. Klein, Y. L. Traon, S. Arzt, S. Rasthofer, E. Bodden, D. Octeau, and P. McDaniel, "I know what leaked in your pocket: uncovering privacy leaks on android apps with static taint analysis," arXiv preprint arXiv:1404.7431, 2014.

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.

M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel, "An empirical study of cryptographic misuse in android applications," in Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS '13, (New York, NY, USA), pp. 73–84, ACM, 2013.

M. Zhang, Y. Duan, Q. Feng, and H. Yin, "Towards automatic generation of security-centric descriptions for android apps," in Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security, CCS '15, (New York, NY, USA), pp. 518–529, ACM, 2015.

P. Brodley and leviathan Security Group, "Zero Permission Android Applications." https://www.leviathansecurity.com/blog/zero-permission-android-applications/. Accessed: 10/02/2013.

P. Ratazzi, Y. Aafer, A. Ahlawat, H. Hao, Y. Wang, and W. Du, "A systematic security evaluation of Android's multi-user framework," in Mobile Security Technologies (MoST) 2014, MoST'14, (San Jose, CA, USA), May 17 2014.

P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, "Android permissions remystified: A field study on contextual integrity," in Proceedings of the 24th USENIX Conference on Security Symposium, SEC'15, (Berkeley, CA, USA), pp. 499–514, USENIX Association, 2015.

Q. A. Chen, Z. Qian, and Z. M. Mao, "Peeking into your app without actually seeing it: Ui state inference and novel android attacks," in Proceedings of the 23rd USENIX Conference on Security Symposium, SEC'14, (Berkeley, CA, USA), pp. 1037–1052, USENIX Association, 2014.

R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "Whyper: Towards automating risk assessment of mobile applications," in Proceedings of the 22Nd USENIX Conference on Security, SEC'13, (Berkeley, CA, USA), pp. 527–542, USENIX Association, 2013.

R. Wang, L. Xing, X. Wang, and S. Chen, "Unauthorized origin crossing on mobile platforms: Threats and mitigation," in Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, (New York, NY, USA), pp. 635–646, ACM, 2013.

S. Fahl, M. Harbach, M. Oltrogge, T. Muders, and M. Smith, "Hey, you, get off of my clipboard," in In proceeding of 17th International Conference on Financial Cryptography and Data Security, 2013.

S. H. Kim, D. Han, and D. H. Lee, "Predictability of android openssl's pseudorandom number generator," in Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS '13, (New York, NY, USA), pp. 659–668, ACM, 2013.

T. Li, X. Zhou, L. Xing, Y. Lee, M. Naveed, X. Wang, and X. Han, "Mayhem in the push clouds: Understanding and mitigating security hazards in mobile push-messaging services," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, (New York, NY, USA), pp. 978–989, ACM, 2014.

W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of android application security," in Proceedings of the 20th USENIX conference on Security symposium, 2011.

X. Jin, X. Hu, K. Ying, W. Du, H. Yin, and G. N. Peri, "Code injection attacks on html5-based mobile apps: Characterization, detection and mitigation," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, (New York, NY, USA), pp. 66–77, ACM, 2014.

X. Zhang, K. Ying, Y. Aafer, Z. Qiu, and W. Du, "Life after app uninstallation: Are the data still alive? data residue attacks on android," in NDSS, 2016.

X. Zhou, S. Demetriou, D. He, M. Naveed, X. Pan, X. Wang, C. A. Gunter, and K. Nahrstedt, "Identity, location, disease and more: Inferring your secrets from android public resources," in Proceedings of the 2013 ACM SIGSAC Conferenceon Computer & Communications Security, CCS '13, (New York, NY, USA), pp. 1017–1028, ACM, 2013.

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

Y. Acar, M. Backes, S. Bugiel, S. Fahl, P. McDaniel, and M. Smith, "Sok: Lessons learned from android security research for appified software platforms," in 37th IEEE Symposium on Security and Privacy (S&P '16), IEEE, 2016.

Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in Proceedings of the 23rd USENIX Conference on Security Symposium, SEC'14, (Berkeley, CA, USA), pp. 1053–1067, USENIX Association, 2014.

Y. Michalevsky, G. Nakibly, A. Schulman, and D. Boneh, "Powerspy: Location tracking using mobile device power analysis," in 24th USENIX Security Symposium, 2015.

Z. Qu, V. Rastogi, X. Zhang, Y. Chen, T. Zhu, and Z. Chen, "Autocog: Measuring the description-to-permission fidelity in android applications," in Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS'14, (New York, NY, USA), pp. 1354–1365, ACM, 2014