Androidov sigurnosni model

Android, jedan od najpopularnijih operacijskih sustava na svijetu i dominantni OS na pametnim telefonima, ima, kao što znamo, Unixoidne korijene. Dapače, ako bismo inzistirali na kernelu kao definirajućoj komponenti sustava, mogli bismo mirne duše reći kako je Android samo još jedna od distribucija Linuxa.

Kernel kojeg koristi Android tek je blago prilagođeni Linux kernel. I ne samo to, većina arhitekture odražava Linux naslijeđe i Unix filozofiju: "ogulimo" li za korisnike i programere najvidljiviji sloj, onaj virtualizacijski, u kojem se nalaze Dalvik ili ART, imat ćemo osjećaj kako je tu sve jasno: zadavši naredbu "ls -l" u komandnom retku dobit ćemo vrlo familijaran ispis kakav očekujemo na svakom Unixoidu... na prvi pogled čini se kako je Android samo marketinški naziv za još jednu distribuciju Linuxa.

Istina je ipak malo drugačija: unatoč svojoj iznimnoj sličnosti, Android nije "Unix of the yesteryear". Riječ je o POSIX-oidu, sustavu koji je u velikoj mjeri sukladan tom standardu i svojoj starijoj braći, ali ne identičan; dapače, dovoljno je različit da ga POSIX puritanci apsolutno odbijaju svrstati u Unix-like operacijske sustave, u kojem slučaju bi i sam Linux mogao elegantno – otpasti.

No, ono što nas zanima jesu razlike u modelu sigurnosti koji Android definira kao svojevrstan hibridni model koji koristi dobro poznate sigurnosne aspekte Unixa (ne nužno na identičan način) i uz to i sigurnosne aspekte virtualnih strojeva. Sigurnost Androida kompleksna je tema, pa ćemo se u ovom tekstu baviti samo "grebanjem po površini", prikazom razlika koje su najvidljivije korisnicima i administratorima.

Jedna od prvih razlika koju ćemo uočiti jest način na koji Android tretira UID; na "klasičnom" Unixoidu UID predstavlja jedinstveni ID korisnika sustava i sve korisnikove datoteke obilježene su njegovim UID-om, uključujući i aplikacije koje je sam korisnik instalirao.

Android koristi UID na kreativniji način, pa svaka instalirana aplikacija dobiva svoj UID. Svijet Unixa zna za slične mehanizme (primjerice, Apache na sustavima često ima vlastiti UID), ali u slučaju Androida riječ je o ciljanom dizajnu: svaka aplikacija mora imati svoj jedinstveni UID. Za razliku od klasičnih sustava, Android ne dozvoljava (osim u posebnim slučejevima) pokretanje aplikacija pod istim UID-om: kad korisnik tableta pokrene aplikaciju na tabletu, ona neće biti pokrenuta pod njegovim, već pod vlastitim UID-om kojeg je sustav dodijelio toj aplikaciji.

Posebni slučajevi kad je dozvoljeno odstupanje od ovog pravila su dobivanje root privilegija i tzv. "shared UID": proizvođač aplikacije (preciznije, digitalni potpisnik APK datoteke) može u manifestu navesti tu mogućnost (android:sharedUserId), te od sustava dobiti dozvolu da ta aplikacija koristi isti UID kao i neka druga aplikacija, ali samo ako obje dijele istog digitalnog potpisnika. Na taj način olakšava se interoperabilnost aplikacija istog proizvođača: kako svaki proces ima svoj UID i svoj mali sandbox, modularne aplikacije (web browseri i sl.) imale bi gomilu nepotrebnog overheada u komunikaciji s dijelovima koji se nalaze u drugom sandboxu.

Slično kao i njegova starija braća, i Android posjeduje tablicu rezerviranih UID-a koji su hardkodirani u sustavu (android_filesystem_config.h). UID se na Androidu zapravo zove AID, no neka vas promjena naziva ne zavara: funkcionalnost je ista. No, ono što nije isto je količina raspoloživih AID-a: klasični Unix tradicionalno dozvoljava do 65536 UID-a, dok Android dozvoljava do 100000 UID-a, razvrstanih u nekoliko grupa: sistemski (1000-9999), aplikacijski (10000-19999), izolirani (99000-99999). Između njih uvukli su se u zadnjim verzijama Androida i višekorisnički UID-i, i to na posve neortodoksan ali učinkovit način: korisnici dobivaju svoj "mali" UID koji je "00" za prvog

korisnika uređaja (koji na taj način dobiva kvaziadministratorske ovlasti – može administrirati ostale korisnike i njihova prava, ali ni u jednom trenutku ne postaje root korisnik), a ide od "10" za drugog i sve ostale nove korisnike.

Elegantno i posve out-of-the-box rješenje koje pomiruje činjenicu da instalirane aplikacije imaju svoj jedinstveni AID kao i svaki korisnik jest jednostavno "lijepljenje" korisnikovog ID-a na AID aplikacije: tako će aplikacija čiji je AID 12345 u trenutku izvršavanja dobiti privremeni UID nn12345 – u slučaju prvog korisnika to će biti 0012345, a u slučaju drugog korisnika 1012345. Aplikaciji će biti dodijeljeni resursi (CPU, memorija, direktorij...) sukladno tom UID-u. Cijela je operacija za korisnika posve transparentna, a zadovoljava oba sigurnosna uvjeta: i da aplikacija ima svoj jedinstveni UID, kao i da svaki korisnik radi sa svojim privatnim podacima.

GID je doživio tek manje izmjene: za razliku od UID-a koji je poprilično radikalno promijenjen, GID i dalje radi svoj posao: definira grupu koja ima određena prava nad određenim dijelom sustava.

Na Androidu se GID zove "Android Permission" i slično kao na "pravom" računalu, aplikacija koja želi pristupiti određenom hardveru mora imati (osim svojeg koji joj je dodijeljen prilikom instalacije) odgovarajući GID koji joj to dozvoljava.

Iako je GID doživio tek manje, rekli bismo kozmetičke promjene, njegova važnost u sustavu drugačija je i možda i značajnija od one UID-a. Dok je primarna zadaća UID-a (AID-a) osigurati jasno vlasništvo nad resursima i korektnu sigurnosnu izolaciju, GID kontrolira prava pristupa neke aplikacije; procesiranje UID-a je za korisnika faktički transparentno, ali Androidov sigurnosni model nalaže svakoj aplikaciji da prilikom instalacije korisnika upozori koja sve prava želi dobiti od operacijskog sustava.

Željena prava dodjeljuju se sukladno zahtjevima koje autori aplikacije definiraju u tzv. "Android Manifest" datoteci (AndroidManifest.xml) koja predstavlja značajan odmak od "klasične" arhitekture Unixoida: svaka aplikacija mora sadržavati vlastitu Manifest datoteku koja sadrži ključne elemente: naziv aplikacije, prava koja aplikacija traži kako bi normalno funkcionirala (iako se u ovom segmentu autori aplikacije znaju zaletjeti i tražiti prava koja im inače ne trebaju, ali "neka se nađe..." - što predstavlja ozbiljan potencijalni sigurnosni problem), popis biblioteka koje aplikacija koristi, minimalni Android API potreban za ispravan rad aplikacije, te druga prava i obaveze.

Osim tih detalja koje možemo na neki način preslikati na "klasične" operacijske sustave, Manifest datoteka definira i funkcionalnosti tipične za Android, koje komponente dijelom također sudjeluju u sigurnosnom modelu: opis komponenti i njihovu potrebnu funkcionalnost i uvjete pod kojima se mogu ili trebaju izvršiti, popis prava koja trebaju imati druge aplikacije koje žele komunicirati sa tom aplikacijom, te druge funkcionalnosti koje ne spadaju u domenu sigurnosti.

Manifest sadrži i druge definicije koje su vezane uz sigurnost na nivou aplikacije, a koje nisu vezane uz sigurnost na nivou operacijskog sustava; one omogućuju finu kontrolu nad operacijama u korisničkom prostoru, poput čitanja i zapisivanja podaka (adresar, primjerice), razmjene podataka sa drugim aplikacijama i slično. U ovu kategoriju spadaju i prava koja definira sama aplikacija, a koja nisu vezana uz Androidov model prava: prava definirana u samoj aplikaciji koriste se za povezivanje specifičnih dijelova aplikacije i komunikaciju sa drugim aplikacijama.

Osnovu sigurnosti Android sustava, dakle, čine UID i GID koji se ponašaju slično ali ne identično ostalim Unixoidima, dok Manifest datoteka definira finije aspekte kontrole nad ponašanjem aplikacije. Ove tri komponente zajedno čine bazu za viši nivo sigurnosti sustava: sandboxing.

Sandboxing je jedan od načina izoliranja procesa od ostatka sustava: termin je primarno poznat iz svijeta web browsera, ali se svodi na funkcionalnost koju imaju druga slična rješenja: chroot, Solarisove zone i slično; sandboxing nije virtualizacija, već izoliranje procesa od ostatka sustava. Sandboxing je za aplikacije obavezan – bez obzira je li riječ o nativno pisanoj aplikaciji ili o Java aplikaciji koja se odvija kroz još jedan sloj virtualizacije (Dalvik ili ART), u trenutku izvršavanja Android će joj dodijeliti tražene resurse i odvojiti je u njen mali pješčanik, dati joj nešto pijeska (UID-om definiran direktorij za spremanje podataka) i ostaviti je da se igra. Aplikacije iz svog pješčanika ne mogu: izlaz iz definiranog direktorija ili posezanje za komadom memorije koji aplikaciji ne pripada

nije dozvoljeno, no aplikacijama još uvijek ostaje na raspolaganju komunikacijski kanal prema drugim aplikacijama: ako je u Manifest datoteci navedena želja za mogućnošću komunikacije prema određenim drugim aplikacijama (kroz zajednički UID, primjerice), Android će aplikaciji dozvoliti prava sukladna traženim, probušiti odgovarajuće rupe u pješčaniku i dozvoliti komunikaciju sa drugim pješčanicima (također kroz klasične mehanizme poput IPC-a). Sigurnosni model sandboxa je čvrst, ali ne nužno i neprobojan, a posebice je osjetljiv na "rootane" uređaje kojima je otključana mogućnost davanja root privilegija aplikacijama, jer aplikacije koje dobiju root privilegije mogu bez problema izaći iz okvira pješčanika i pristupati faktički cijelom uređaju.

Jedna od vrijednih razlika u načinu izvršavanja aplikacija na Android platformi je činjenica da, za razliku od uobičajenog načina rada gdje aplikacija može biti pokrenuta sa UID-om 0 (root) ili sa UID-om korisnika koji je aplikaciju pokrenuo, Android koristi sustav koji razlikuje preinstalirane aplikacije od onih koje je korisnik naknadno instalirao: sve preinstalirane (sistemske) aplikacije koriste specifični "platform key" koji im omogućuje da budu pokrenute sa "system" privilegijama, dok aplikacije koje instalira korisnik imaju ključ kojeg im je dao proizvođač aplikacije. Ovaj jednostavan princip omogućava automatsko podizanje privilegija svim aplikacijama koje koriste platform key, ali predstavlja i veliki sigurnosni problem: zaboravi li developer Androida definirati svoj platform key, sustav će koristiti podrazumjevani ključ kojeg je moguće izvući iz izvornog koda i njime potpisati vlastite aplikacije koje će zatim lako dobiti privilegije koje normalnim putem ne bi bile u stanju.

Podizanje Android sustava, koje također ima svoju ulogu u sigurnosti sustava i svoje različitosti u odnosu na klasike, možemo podijeliti na dva koraka: onaj kojeg obavlja init, i onaj kojeg obavlja Zygote. Kao i na drugim Unixima, nakon što kernel obavi pripremne radnje, prvi proces kojeg sustav pokreće je init. Androidov init nema tako složenu zadaću kao na Linux distribucijama, njegovo je tek da izvrši /init.rc i /init.*.rc datoteke i uspostavi Property Service (servis koji sadrži podatke o podešavanjima sustava, slično Windows Registry-u), nakon čega pokreće Zygote i njemu prepušta ostatak posla.

Zygote je glavni otpravnik poslova na Android uređajima: on pokreće proces nazvan system_server koji pak pokreće Android Framework (biblioteke i servisi) bez kojeg aplikacije ne bi mogle raditi, ali isto tako brine i o pokretanju aplikacija, koje Zygote pokreće tako što se forka i u forku (koji je Dalvik proces) pokreće aplikaciju. Na taj način štede se resursi sustava, ali i otvara potencijalna opasnost da infekcijom Zygote napadač automatski inficira i sve aplikacije pokrenute na inficiranom stroju.

Sigurnosni model Androida ovisi u dobrom dijelu o kvalitetnoj virtualizaciji, tj. o sposobnosti Dalvika ili ART-a da procese drže izoliranim od ostatka sustava. No, zbog dvojne naravi sustava, on je otvoren i za razne vrste napada. Kernel i biblioteke, primjerice, napisani su u nativnom kodu arhitekture i to ih čini otvorenim za klasične napade kakve viđamo i na PC računalu. Zbog svoje naravi, Dalvik je otporan na takve napade, ali nije neranjiv.

Standardni vektori napada na Android uređaje rijeđe koriste naprednije tehnike i sigurnosne propuste u OS-u ili u Dalvik/ART virtualizaciji; kao i na desktop računalima, najčešći vektori zaraze su oni koji uključuju interakciju sa korisnikom ("klikni ovdje da bi dobio nešto besplatno") čija je akcija nužna za instaliranje malware aplikacije ili obavljanje kakve štetne radnje. Ozbiljniji napadi na Android sustave često uključuju kombinaciju više propusta jer samo jedan obično nije dovoljan, tj. ne omogućava izlazak iz pješčanika i preuzimanje kontrole nad cijelim sustavom.

Valja imati na umu da sigurnosni propusti Androida velikim dijelom služe i kao put za "oslobađanje" uređaja, tj. način da njegov legitimni korisnik otključa mogućnost korištenja root privilegija; kako većina proizvođača uređaja eksplicitno ne dozvoljavaju tu mogućnost, jedini način za njeno dobivanje je iskorištavanje sigurnosnih propusta sustava. Time se otvara pandorina kutija: iako je za korisnika zapravo dobro da ima punu kontrolu nad uređajem, neznanje ili nepažnja mogu prouzročiti trajne štete; također, takvim se postupkom dijelom zaobilazi Androidov sigurnosni model (aplikacije zaista ogromnom većinom ne trebaju administratorske ovlasti) kojeg je time moguće kompromitirati, ali isto tako i ojačati (antivirusni i protulopovski programi, firewall i slične aplikacije koje za svoju funkcionalnost zahtjevaju visoke privilegije koje inače ne mogu dobiti).

Zbog nekih svojih višekorisničkih manjkavosti Android još uvijek nije sretno rješenje za BYOD i Enterprise okruženja (aplikacije poput Samsungovog Knox-a to pokušavaju riješiti) jer ne pruža

dovoljnu zaštitu između korisnika, taj će se problem u skoroj budućnosti riješiti. Isto tako, brzina kojom se Android mijenja i promjene koje pritom nastaju, a koje nisu trivijalne naravi, podjednako otežavaju i održavanje Android uređaja i njegovo kompromitiranje; u praksi postoji velika količina različitih uređaja sa različitim verzijama Androida, što značajno komplicira integraciju u šira okruženja i otežava administriranje istih.

Trebamo li očekivati "konačno rješenje" pitanja sigurnosti na Androidu? Ne – to se nikada neće dogoditi. Baš kao i u slučaju bilo kojeg drugog modernog računala, sigurnost i ranjivost evoluiraju zajedno sa sustavom: bilo je i bit će sigurnosnih problema. Od toga ne možemo pobjeći, no ono što ne bismo smjeli dozvoliti je "patch disconnect", kad proizvođači hardvera jednostavno odluče da neće izdavati nadogradnje za svoje "stare" (čitaj: starije od godine dana) uređaje. Upravo to je najveći sigurnosni problem Androida.

sri, 2014-11-19 06:07 - Radoslav Dejanović**Vote:** 0

No votes yet

Source URL: https://sysportal.carnet.hr/node/1450