# Acces root

4.

- Android-ul are la bază un kernel de Linux, care implementează modelul de securitate DAC.
- Asta înseamnă că un anumit user, root (UID=0) are putere absolută în sistem.
- Pe Android, userul root poate să: facă bypass la sandbox, să citească şi să scrie fişierele private ale oricărei aplicaţii, să modifice partiţii care ar trebui să fie read-only, să pornească şi să oprească servicii de sistem, să elimine aplicaţii de sistem.

5.

- Folosirea utilizatorului root poate afecta stabilitatea dispozitivului, de aceea nu este permis în mod implicit pe dispozitivele din producţie.
- În plus, Android-ul încearcă să minimizeze numărul de procese de sistem cu permisiuni de root, deoarece vulnerabilităţile acestor procese pot permite atacuri de tip “privilege escalation” (în care aplicaţiile third-party pot obţine drepturi de root).
- Dacă este activat SELinux în modul enforcing, procesele vor fi limitate de politica globală de securitate.
    - Asta înseamnă că dacă un proces root este compromis nu înseamnă neapărat acces complet la dispozitiv.
    - Dar tot vor putea accesa datele private şi vor putea modifica comportamentul sistemului.
    - Chiar şi un proces constrâns de SELinux poate exploata o vulnerabilitate din kernel pentru a obţine acces la root complet.

6.

- Totuşi accesul la root are nişte avantaje destul de mari:
    - Se poate face uşor debugging şi reverse engineering al aplicaţiilor.
    - Se pot realiza anumite personalizări ale sistemului
    - Se pot implementa anumite aplicaţii speciale de genul firewall, backup complet, partajarea reţelei, fără să fie nevoie de modificarea framework-ului sau de adăugarea unor servicii de sistem.

# Accesul root pe diferite variante de build

8.

- Sistemul de build al Android-ului poate genera diferite variante de build pentru un anumit dispozitiv. Acestea au un număr diferit de aplicaţii, utilitare şi valori ale proprietăţilor de

sistem (care generează un comportament diferit al sistemului). Anumite variante de build permit accesul root în shell.
- Tipul build-ului curent se poate obţine din proprietatea de sistem ro.build.type.
- Build-ul user nu include utilitare de diagnostic şi dezvoltare, daemonul adb (adbd) este dezactivat în mod implicit, nu permite debugging pentru aplicaţiile care nu au debuggable true în fişierul manifest, şi nu permite accesul root prin shell.
- Build-ul userdebug este similar cu user, doar ca în plus permite debugging-ul tuturor aplicaţiilor şi daemnonul adb (adbd) este activ în mod implicit.

9.
- Aici avem un demo pe un dispozitiv Pixel XL.
- Se obţine valoarea proprietăţii ro.build.type - build de tipul user.
- Se obţine valoarea proprietăţii ro.secure - daemonul adbd care rulează iniţial ca root, face drop la capabilităţi (cu excepţia CAP_SETUID şi CAP_SETGID); apoi adaugă câteva grupuri suplimentare pentru accesul la interfeţele de reţea, la sdcard şi la logurile de sistem; în final îşi schimba UID şi GID la AID_SHELL (UID=2000).
- Se poate vedea label-ul de SELinux al daemonului adbd, UID, GID, grupurile suplimentare şi capabilităţile.
- CapBnd =  0000000000c0 = (CAP_SETUID|CAP_SETGID )

10.
- Build-ul de dezvoltare (engineering) permite debugging-ul tuturor aplicaţiilor, adb este activ în mod implicit, proprietatea ro.secure este 0 (asta înseamnă că daemnonul adbd va continua să ruleze ca root şi va deţine întregul set de capabilităţi (nu face drop la capabilităţi).

11.
- Comanda su (substitute user) este folosită de obicei pentru a face switch de la un user normal la superuser. Are bitul de SUID setat, şi permite procesului apelant să obţină un shell root şi să ruleze o comandă ca al user (UID=0).
- În build-urile userdebug, accesul root poate fi obţinut fără restartarea ADB ca root, prin folosirea comenzii su.
- Doar utilizatorii root (UID=0) şi shell (UID=2000) au acces la comanda su implicită.
- Acesta va seta UID-ul şi GID-ul procesului la 0, şi va porni un shell nou.
- Orice comandă executată în acest shell îi va moşteni privilegiile.

## Rootarea dispozitivelor din producţie

13.
- În producţie (dispozitivele comercializate) vor exista doar build-uri de tip user.

- Daemonul adb va rula sub userul shell, şi nu va exista nici o comandă su pe dispozitiv.
- Nu sunt permise operaţii care modifică configuraţia de bază a sistemului de operare.
- Nu este permis accesul la kernelul de Linux.
- Astfel de operaţii sunt efectuate de comenzi care necesită privilegii de root.
- Obţinerea accesului de root pe un astfel de dispozitiv se numeşte rootare şi acest lucru necesită deblocarea bootloader-ului.
- Dispozitivele care nu permit deblocarea bootloader-ului nu vor putea fi rootate.

14.

- Avem două metode mai vechi de a roota un dispozitiv: prin scrierea unei noi imagini de boot sau prin modificarea imaginii de sistem.
- Pe unele dispozitive, un build user poate fi transformat într-un build engineering dau userdebug dor prin scrierea unei noi imagini de boot (ce conţine un kernel custom).
    - Acesta va schimba valorile proprietăţilor ro.secure şi ro.debuggable. Astfel, se va rula daemonul adb ca root şi va permite accesul root prin shell.
    - Totuşi, majoritatea build-urilor user curente, vor dezactiva acest comportament la compilare, şi valorile proprietăţilor de sistem vor fi ignorate de către daemonul adb.
- A doua modalitate de rootare a dispozitivelor este prin modificarea partiţiei system.
    - Mai exact, se despachetează imaginea system, se adaugă utilitarul su şi se suprascrie partiţia system cu noua imagine.
    - Acest lucru permite accesul root din shell dar si din aplicaţiile third-party.
    - Totuşi, din Android 4.3 nu este permis aplicaţiilor să execute programe cu SUID, prin faptul că Zygote face drop la capabilităţi şi partiţia system este montată cu flag-ul nosetuid.

15.

- În plus, versiunile mai noi au SELinux pe modul enforcing, ceea ce însemnă că execuţia unui proces cu privilegii root nu va schimba contextul de securitate, procesul va fi încă limitat de politica MAC.
- Astfel, SELinux şi alte măsuri de securitate vor trebui dezactivate, pentru obţinerea accesului root - prin înlocuirea imaginilor boot sau system.
- Totuşi acest lucru va împiedica primirea actualizărilor de sistem de la producător.
- De aceea este bine să păstrăm sistemul stock şi să facem cât mai puţine modificări pentru rootare.

16.

- Metoda cea mai folosită în ziua de azi este folosirea unui pachet OTA care adaugă şi modifică fişiere de sistem, fără să fie nevoie de înlocuirea întregii imagini de sistem.
- Majoritatea aplicaţiilor superuser includ un pachet OTA ce trebuie instalat (folosind un custom recovery) şi o aplicaţie manager care se poate actualiza.

17.

- Cea  mai populară soluţie pentru rootare este SuperSU.
- Include un pachet OTA şi o aplicaţie de management.
- Este dezvoltat în mod activ de către Chainfire (Jorrit “Chainfire” Jongma)

18.

- Pachetul OTA include câteva binare native compilate pentru: arm, arm64, armv7, mips, mips64, x86, x64.
- Include scripturi pentru instalarea şi pornirea daemonului SuperSU
- Include aplicaţia de management (apk-ul)
- De asemenea, include cele 2 scripturi (update-binary şi updater-script) care aplică efectiv modificările din pachetul OTA

19.

- Updater-script întâi montează read-write sistemul de fişiere rootfs şi partiţiile system şi data
- Apoi copiază fişierele la destinaţiile lor din sistemul de fişiere.
- Binarele native su şi daemonsu sunt copiate în system/xbin/
- Apk-ul aplicației de management este copiat în /system/app şi va fi automat instalat la rebootarea dispozitivului.
- Apoi scriptul install-recovery.sh este copiat în /system/etc. Acest script va fi folosit pentru a porni anumite componente la bootare.

20.

- Se vor seta permisiunile şi labelurile de securitate SELinux pentru binarele nou instalate (se va seta labelul u:object_r:system_file:s0)
- Va apela /system/xbin/su --install pentru a efectua nişte iniţializări după instalare.
- În final,va demonta partiţiile system şi data.
- Dacă scriptul se termină cu succes, dispozitivul va reboota în sistemul de operare principal.

21.

- Pe un dispozitiv mai nou, Samsung Galaxy S7 Edge, iniţializarea de la bootare este făcută de scriptul init.supersu.rc
- Vedem că serviciul daemonsu este pornit folosind scriptul launch_daemonsu.sh
- Acesta este pornit sub userul root şi are labelul de securitate u:r:supersu:s0

22.

- Serviciul daemonsu este folosit pentru a evita constrângerile de securitate discutate la începutul cursului (Zygote face drop la capabilităţi, SELinux în modul enforcing).

- Astfel, aplicaţiile vor folosi binarul su pentru a executa comenzi sub utilizatorul root, care mai departe va trimite comenzile printr-un socket Unix către daemonsu care le execută fiind root şi având contextul SELinux u:r:supersu:s0.

23-24.

- Putem vedea aici un output parţial al comenzii “ps -Z”, în timp ce rulează comanda “su -c sleep 100” din terminal.
- Aplicaţia de management SuperSU (eu.chainfire.supersu) rulează cu utilizatorul asociat aplicaţiei şi are labelul: u:r:untrusted_app:s0.
- Utilitarul su rulează sub userul shell şi domeniul SELinux u:r:shell:s0.
- Serviciul daemonsu rulează sub userul root şi domeniul u:r:supersu:s0. Observăm că este o instanţă de daemonsu asociată PID-ului comenzii su de mai sus (PID 24047). Deci practic procesul su trimite o comandă prin socket către daemonsu:0:24047 - o instanţă de daemonsu creată special pentru a rula această comandă.
- Daemonsu porneşte sush pentru a executa sleep-ul, care vedem că moşteneşte userul root şi domeniul u:r:supersu:s0. Vedem că procesul sush are ca părinte pe daemonsu.

25.

- Aplicaţia de management SuperSU are numele de pachet “eu.chainfire.supersu” (numele procesului).
- Această aplicaţie va întreba utilizatorul dacă permite accesul root pentru aplicaţiile care încearcă să folosească comanda su.
- Accesul root poate fi acordat o singură dată, pentru o perioadă sau permanent, pentru o anumită aplicaţie.
- SuperSU are o listă internă de aplicaţii care au primit acces root de la utilizator şi pentru care nu se va mai afişa nici un mesaj (întrebare) pentru utilizator.

26.

- În CyanogenMod - denumit mai nou LineageOS (cea mai populară versiune custom de Android), su este pornit direct ca un daemon din scriptul de iniţializare init.superuser.rc.
- Acest script defineşte serviciul su_daemon, care poate fi pornit şi oprit folosind proprietatea de sistem persist.sys.root_access.
- Valoarea acestei proprietăţi va determina dacă avem acces de root pentru aplicaţii, pentru shell-ul adb, pentru toate sau pentru nici una.
- Accesul root este dezactivat în mod implicit, dar este uşor de activat din Development Options.

# Tutorial complet de rootare

28.

- În continuare avem un tutorial pas cu pas de rootare a unui telefon Samsung Galaxy S7 Edge.
- În primul rând, din setări, activăm Developer Options, apoi din acel meniu activăm OEM Unlocking şi USB debugging.
- Downloadăm imaginea de TWRP pentru hero2lte - acesta este numele de cod pentru versiunea de telefon. TWRP este un recovery custom.
- Rebootam telefonul şi intrăm în modul Download (numit şi Odin) prin apăsarea tastelor [Volume Down]+[Home]+[Power].
- Folosind utilitarul Odin care rulează pe Windows, deschidem imaginea download-ată şi o scriem pe dispozitiv.
- Imediat, înainte să booteze în sistemul de operare principal, intrăm în modul recovery prin apăsarea butoanelor [Volume Up]+[Home]+[Power] - pentru a intra în TWRP.

29.

- În TWRP, întâi trebuie să formatăm partiţia de date (altfel nu ne va lăsa să modificăm partiţiile boot şi system).
- Vom downloada şi copia pe dispozitiv (pe sdcard) arhivele OTA SuperSU şi no-verity (cu adb push).
- Cu TWRP vom aplica pachetul OTA no-verity, apoi pachetul OTA SuperSU.
- Vom reboota telefonul în sistemul de operare principal - aici s-ar putea sa dureze cate minute (sau zeci de minute).
- Dacă intra în boot loop înseamnă că am făcut ceva greşit şi dispozitivul nu bootează.

30.

- Odată pornit sistemul de operare principal, putem downloada şi instala ultimele versiuni ale aplicaţiilor SuperSU şi TWRP.
- Din SuperSU vom acorda acces de root aplicaţiilor care cer acest lucru (de exemplu TWRP, shell).
- Din adb shell, vom da comanda “su” şi vom obţine un shell cu acces root.