SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. xxxx

PROCES PROVOĐENJA FORENZIČKE ANALIZE ANDROID OPERACIJSKOG SUSTAVA

Valerija Šimić

Zagreb, lipanj 2014.

Sadržaj

1. Uvod

Naziv forenzika obuhvaća primjenu širokog spektra znanstvenih grana za utvrđivanje činjenica u sudskim ili upravnim postupcima [21]. Jedna od grana forenzike je digitalna forenzika. Digitalna forenzika definira se kao prikupljanje, zaštita i analiza dokaza u digitalnom obliku te njihova prezentacija u kasnijim eventualnim sudskim postupcima [22]. Primjena digitalne forenzike ne zaustavlja se samo na sudskim postupcima, nego ima daleko širu primjenu, od zaštite korisnika, sigurnosti korporacija do zaštite državnih interesa od internacionalne špijunaže. U današnje vrijeme je prijeko potrebna zbog toga što se tehnologija uvukla u sve aspekte našeg života i postala dio naše svakodnevice. Postali smo ovisni o tehnologiji i samim tim nas čini jako ranjivima. Svi naši podaci od datuma rođenja, mjesta stanovanja, obrazovanja, medicinskih kartona pa do dnevnika i bankovnih računa, praktički čitav naš život je pohranjen negdje u digitalnom obliku i postavljen „na izvolite“ ljudima koji su vješti sa njom. U zadnjih par godina pojavili su se pametni telefoni koji su osvojili ljude diljem svijeta. Među njima izdvojili su se oni sa Android operacijskim sustavima koji su u kratko vrijeme postigli dominaciju nad tržištem. U jednom prijenosnom telefonu, koji više ne služi samo za kratke poruke i pozive, sada se može pronaći velika količina podataka o vlasniku, njegovim navikama i interesima. Osoba koja zna šta radi bi to mogla lako zloupotrijebiti. S vremenom se Android operacijski sustav proširio i na druge platforme, kao što su tablet računala, televizije, mikrovalne pećnice pa i hladnjaci. Zbog toga, možemo zaključiti da će se sve više pojavljivati u raznim forenzičkim istragama, te sukladno s tim područje forenzike Android uređaja postaje sve važnije i neophodnije.

U ovom radu je obrađen proces forenzičke analize uređaja sa Android operacijskim sustavom. U drugom poglavlju se spominju smjernice kojim bi se trebale voditi osobe koje provode forenzičku istragu u sklopu digitalne forenzike. Potom će se ukratko reći nešto o vrstama forenzičkih istraga koje se tiču Android forenzike, podacima koji se mogu naći na uređajima, vrstama memorije i datotečnim sustavima na Androidima. Na kraju tog poglavlja su opisane tehnike akvizicije podataka. U trećem poglavlju je opisan sam proces forenzičke istrage na Android uređaju uključujući procedure rukovanja uređajem, pribavljanje podataka sa UMS uređaja, particija sustava i od pokrenutih aplikacija, analizu podataka i izvještaj. Na kraju, u četvrtom poglavlju je primjer napravljene forenzičke istrage opisane kroz izvještaj.

2. Android forenzika

U ovom poglavlju će se prvo iznijeti 4 principa koja bi se trebala poštivati kod pribavljanja digitalnih dokaza. Zatim će biti riječ o vrstama istraga. Potom će biti opisani vrste memorije, najvažniji datotečni sustavi na Androidima i mjesta gdje se nalaze korisni podaci za istragu, te će na kraju biti navedene i objašnjene tehnike akvizicije podataka s Android uređaja.

Forenzika Android uređaja je grana digitalne forenzike. Kao takva, u početku su se na nju odnosila pravila digitalne forenzike. Glavno pravilo je da se na pribavljenom uređaju na kojem se obavlja istraga, ne smije ništa promijeniti, u protivnom dokazi dobiveni s njega postaju nevažeći za sud. Međutim, to se pravilo za forenziku pametnih telefona ubrzo odbacilo zbog arhitekture samog uređaja i načina na koji on radi. Gotovo je nemoguće izvući bilo kakve podatke tj. digitalne dokaze bez ikakvih promjena na uređaju. Zbog potrebe za nekim pravilima koja bi mogla biti prihvaćena na sudu, date su neke smjernice.

Asocijacija ACPO – Association of Chief Police Officers iz Ujedinjenog Kraljevstva je sastavila dokument „Good Practice Guide for Computer-Based Electronic Evidence“ [3]. U tom dokumentu se spominju 4 osnovna principa kod pribavljanja i obrade računalno baziranih elektroničkih dokaza:

- Nijedna akcija, napravljena od strane agencija za provedbu zakona ili njihovih agenata, ne smije mijenjati podatke na računalu ili mediju za pohranu podataka koji potencijalno mogu biti predstavljeni na sudu.
- U okolnostima gdje osoba smatra neophodnim pristupanje originalnim podacima na računalu ili mediju za pohranu podataka, ta osoba mora biti kompetentna da to napravi i biti u mogućnosti da preda dokaz i objašnjenje relevantnosti i posljedice svojih postupaka.
- Tijek revizije ili neki drugi zapis svih procesa napravljenih na računalno baziranom elektroničkom dokazu treba biti napravljen i sačuvan. Neovisna treća strana, treba biti u mogućnosti ispitati te procese i dobiti isti rezultat.
- Osoba na čelu istrage ima odgovornost osigurati da se zakon i ovi principi poštuju.

Postoji više vrsta forenzičkih istraga. Najpoznatije su one čiji će dokazi biti predstavljeni na sudu. Tu se posebno treba paziti, kao što to nabrojana 4 principa zahtijevaju, na ponovljivost svih postupaka koje smo napravili na uređaju prilikom pribavljanja dokaza, što manje mijenjanje uređaja te pomno dokumentiranje svega napravljenog. Drugi slučaj su unutarnje istrage u raznim tvrtkama. One se mogu provoditi zbog utvrđivanja sigurnosti, provjere zaposlenika, curenja osjetljivih podataka, neprimjerene uporabe tvrtkinih resursa, napada na sustav i drugih situacija [1]. Razlika između istrage za sud i istrage unutar tvrtke je to što uglavnom tvrtke daju zaposlenicima svoje uređaje koji su još uvijek tvrtkino vlasništvo, te ih uvijek mogu zatražiti na provjeru i oni se uglavnom daju svojevoljno, dok to u istragama za sud u većini slučajeva nije tako. Također, digitalna forenzika može biti potrebna i kod razvoda, zahtijevanja skrbništva nad djecom i imovinskim sporovima. Forenzika igra veliku ulogu i u obrani države od potencijalnih napada i špijunaže, te pripremi za moguće napade u budućnosti ali i za pronalaženje tehnika za vlastitu špijunažu. Forenzičkom istragom se može poslužiti i obična osoba radi provjere sigurnosti vlastitih uređaja.

U ovom slučaju pričamo o Android operacijskim sustavima, a baza će biti pametni telefoni. Osim što pametni telefon može biti predmet istrage zbog podataka na njemu, npr. istraživanje dječje pornografije, ili zato što je bio predmet napada, pa se želi utvrditi koje su posljedice, on može biti i sredstvo napada. Primjeri za to su da može biti upotrebljen da se na njega spreme ukradeni podaci, da se pokrene snimanje ili slikanje u pozadini bez znanja vlasnika, te se tako izvuku osjetljive informacije, te npr. neki mrežni napadi.

Da bi se mogla izvesti forenzička analiza Android uređaja, potrebno je znati kako je izvedena memorija na tom uređaju da bi znali gdje tražiti dokaze.

U Android pametnih telefona razlikujemo dvije vrste memorije: promjenjivu (RAM) i nepromjenjivu NAND (eMMC i SD-card) memoriju. RAM i eMMC memorija su napravljeni zajedno na jednom čipu. To komplicira forenziku tih uređaja zato što se memorija ne može kao kod računalne forenzike jednostavno izvaditi, spojiti na blokator pisanja i prepisati.

RAM memorija služi za učitavanja, izvršavanje i manipuliranje ključnim dijelovima operacijskog sustava, aplikacija i podataka i ona se briše nakon gašenja uređaja [1]. U njoj se mogu naći jako važni podatci kao što su lozinke, korisnička imena, ključevi za kriptiranje, podatci aplikacija kao što je primjerice broj računa.

NAND memorija ostaje zapisana i nakon gašenja i ponovnog pokretanja uređaja. Podijeljena je na nakupine (engl. *chunks*) koje se još zovu i stranice (engl. *pages*) koje su organizirane u veće cjeline – blokove (engl. *blocks*). Karakterističnost NAND memorije je da se može pisati i čitati na razini jedne stranice, dok se briše na razini bloka. Blok, osim stranica, tj. prostora gdje se spremaju podatci, sadrži i *out of band* (OOB) dio memorije poslije svake stranice gdje se spremaju metapodaci. Ono što se nalazi u OOB ovisi o tome u kojem se dijelu memorije OOB nalazi, ali inače sadrži podatke *Memory Technology Device* (MTD) sustava koji je zadužen za upravljanje NAND memorijama i druge metapodatke stranica, te se i tu mogu naći korisni podaci[1].

Android operacijski sustavi ne podržavaju samo jedan datotečni sustav, već koriste i podržavaju više njih. Njihov popis se može naći u datoteci */proc/filesystems*. Podržavaju puno više datotečnih sustava nego što ih inače koriste. Najznačajniji datotečni sustavi su [1]:

1. *Rootfs* – služi kao točka vezivanja za korijenski datotečni sustav (*/*)
2. *Tmpfs* – datotečni sustav koji sprema podatke u virtualnu memoriju i u njemu se spremaju privremene datoteke, te se ti podaci brišu gašenjem uređaja *(/dev, /mnt/asec, /app-cache, /mnt/sdcard/.android_secure*)
3. *Cgroup* (*control group*) – je datotečni sustav koji pruža mogućnost pristupanju i definiranju različitih parametara jezgre [9] (*/dev/cpuctl/acct*)
4. *Proc* – sadrži informacije o jezgri operacijskog sustava, procesima i konfiguraciji sustava (*/proc*)
5. *Sysfs* – izvozi informacije o uređajima i upravljačkim programima iz modela jezgre operacijskog sustava prema korisničkom prostoru [10] (/sys)
6. *Devpts* – koristi se za virtualne terminal sjednice (*/dev/pts*)

7. *Ext3* i *ext4* (*extended file system*) – standardni Linux datotečni sustavi (*/cache, /system, /data/data*)

8. *Yaffs2* (*yet another flash file system*) - napravljen isključivo za posebne potrebe NAND memorije kao što je upravljanje lošim blokovima, međutim u novije vrijeme je zamijenjen s *ext4* datotečnim sustavom (*/proc/yaffs*)

9. *Vfat* i *fat32* – koriste se na SD-kartici i eMMC-u zbog interoperabilnosti s Windowsima koji su najrašireniji računalni operacijski sustav *(/mnt/sdcard, /secure/aasec, /mnt/emmc*)

Prvih 6 su pseudo datotečni sustavi koji nisu bazirani na mediju, tj. to su virtualni datotečni sustavi koji nisu zapisani ni na jedan fizički uređaj [1].

U forenzičkoj istrazi razlikujemo dvije vrste korisnih podataka: podaci u mirovanju (engl. *data at rest*) i podaci u pokretu (engl. *data in transit*)[1]. Pod *podacima u mirovanju* se podrazumijevaju oni pohranjeni na trajnoj memoriji, npr. SD kartici ili eMMC. Kad je riječ o pametnim telefonima, među njima se može naći jako puno informacija korisnih za istragu. Neki primjeri su:

- SMS/MMS
- Zapisnici poziva
- Glasovne poruke
- Financijske aplikacije
- Kalendarski zapisi
- IM
- Web povijest
- Povijest pretraživanja na Google pretraživaču
- Slike, videozapisi, zvučni zapisi
- Geografske lokacije
- Tvrtkine datoteke
- Korisnička imena i lozinke
- Zapisi aplikacija te ostalo

Podatci u pokretu su podatci koji se trenutno prenose preko mreže, to može biti Wi—Fi mreža, mobilna mreža, bluetooth i ostale, te podaci koji se nalaze u RAM-u. Oni se gube gašenjem uređaja. To su na primjer:

- Lozinke i korisnička imena
- Podaci financijskih aplikacija, brojevi kartica ili neki drugi podaci koji se zbog svoje osjetljivosti također ne spremaju na mobitel

Postoji više tehnika kojim se mogu pribaviti informacije s Android uređaja. U nastavku su objašnjene tehnike ručne, logičke i fizičke akvizicije podataka sa uređaja s Android operacijskim sustavom.

Ručna akvizicija obuhvaća ručno pregledavanje podataka na mobitelu uz interakciju s ekranom i tipkama te se uglavnom koristi uz neku drugu tehniku. Sva interakcija bi trebala biti nekako zabilježena, primjerice snimljena digitalnom kamerom. Tijekom ručne ekstrakcije neizbježno se mijenja stanje uređaja, a podaci mogu biti obrisani i prepisani, te je preporučljivo minimizirati upotrebljavanje ove metode. S njom se ne mogu pregledati obrisani podaci, a dodatne poteškoće nastaju ako je mobitel na nepoznatom jeziku.[5]

Logička akvizicija podrazumijeva izvlačenje podataka s uređaja povezanoga s forenzičkom postajom (engl. *forensic workstation*). *Forenzička postaja* je sklopovlje, posebni uređaj ili primjerice stolno računalo, koje je pripremljeno za izvođenje digitalne forenzike svih digitalnih uređaja ili samo jednog dijela. Logičkom akvizicijom dobijemo podatke koje možemo vidjeti npr. kada pregledavamo SDkarticu u pregledniku. Dobivaju se uglavnom neizbrisani, tj. alocirani podaci s mobitela i to je postignuto pristupanjem datotečnom sustavu. Neki izbrisani podaci se mogu pronaći u SQLite bazama podataka koje su dobivene akvizicijom. U većini slučajeva ova je tehnika dovoljna za pronalaženje potrebnih informacija. Prednost u odnosu na fizičku akviziciju je to što su logičke tehnike puno brže, a nedostatak je što se njima dobiva malo ili ništa izbrisanih podataka.[1]

Za logičke tehnike, potrebno je da opcija traženja pogrešaka putem USB sučelja (engl. *USB debugging*) bude omogućena, a administratorski (engl. *root*) pristup nije nužan. Bez administratorskog pristupa se ne može pristupiti nekim dijelovima memorije, npr. */data* direktoriju, dok administratorski pristup omogućuje pristup svim podacima. Postoji mnogo komercijalnih alata za logičku akviziciju podataka s Android uređaja primjerice: Cellebrite UFED [11], Compelson MOBILedit! [12], Encase Neutrino [13], Micro Systemation XRY [14], te neki otvorenog koda kao što su AFlogical OSE [15] i Adel [16]. U dodacima B, C i E piše više o alatima za Android forenziku. Većina njih koristi *Android Debug Bridge* (ADB) alat koji stvara vezu između računala i Android uređaja i omogućava slanje naredbi na uređaj i interakciju između uređaja i računala [17]. Naredba od najvećeg značaja za ovo područje je `adb pull` koja omogućava povlačenje podataka s mobitela te se s njom može napraviti logička slika sustava na računalu koju kasnije analiziramo.

Još jedna logička tehnika je i analiza sigurnosne kopije (engl. *backup*) mobitela. Podrazumijeva analizu postojeće kopije ili pravljenje nove sigurnosne kopije preko aplikacija koje su isporučene direktno s mobitelom ili se instaliraju s Google Play trgovine [1].

Fizičkom akvizicijom se podrazumijeva bit-po-bit kopija memorije zaobilaženjem datotečnog sustava. Prednost ovih tehnika je što se kopira ne samo alocirani, nego i nealocirani prostor memorije tj. uz postojeće podatke kopiraju se i oni izbrisani i odbačeni. Postoje programske i sklopovske metode pribavljanje bit-po-bit slike sustava. Sklopovske metode su JTAG (*Join Test Action Group*) i *chip-off*. One ne zahtijevaju ni administratorski pristup ni uključenu opciju traženja pogrešaka putem USB sučelja, čime istodobno rješavamo i problem uređaja sa zaključanim zaslonom ili SIM karticom, dok programske metode zahtijevaju administratorski pristup. Sklopovske metode su mnogo teže i kompliciranije za izvesti od programskih, te zahtijevaju stručno osoblje. Njima se može nepovratno oštetiti uređaj. Programske metode su manje opasne u smislu da se njima ne može fizički uništiti uređaj, a najveći problem se svodi na dobivanje administratorskog pristupa uređaju ako nije dostupan. Dobivanjem administratorskog pristupa mijenja se

uređaj, i zbog toga taj proces treba biti dobro dokumentiran i treba se znati točno što je mijenjano, te se čitav mora isprobati prije na pomoćnom uređaju koji je isti kao ispitivani.[1]

Postoje tri vrste administratorskog pristupa:

1. Privremeni administratorski pristup
2. Potpuni administratorski pristup
3. Administratorski pristup preko sustava za oporavak

Administratorski pristup se provjerava sa naredbom `adb shell su`. Ako se na ekranu ispiše znak # znači da je omogućen. [18]

Fizička akvizicija zahtjeva puno kompliciraniju analizu slike sustava nakon njenog stvaranja.

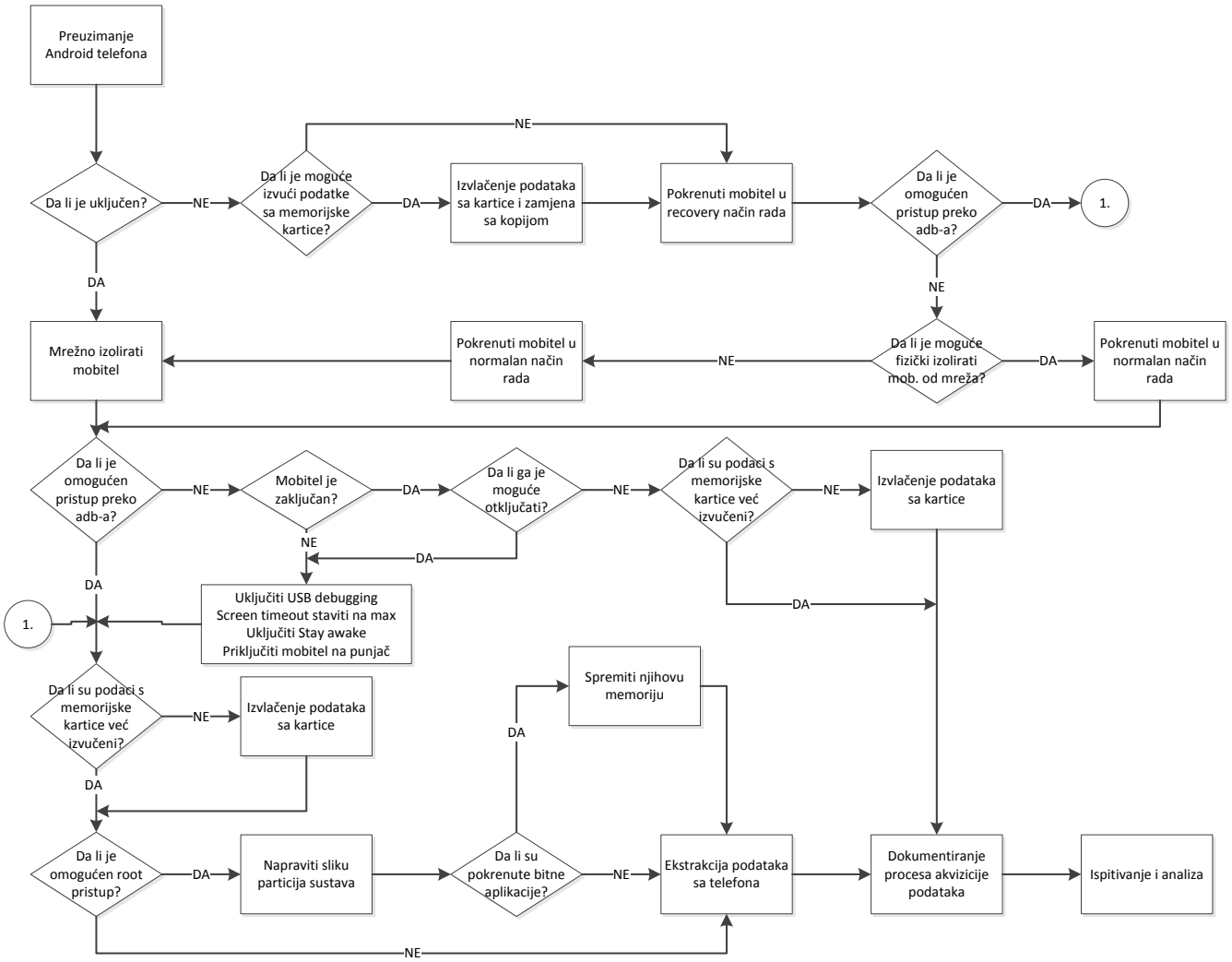
3. Procedure za rukovanje Android pametnim telefonom u sklopu forenzičke istrage

Ako se radi o kriminalističkoj forenzičkoj istrazi u kojoj je pronađen/zaplijenjen Android uređaj, postoji nekoliko stvari koje bi se trebale napraviti čim uređaj dođe u posjed forenzičara. Prave procedure još uvijek nema zbog velikog broja različitih pametnih telefona, a kad je riječ o Android uređajima stvari se još kompliciraju zbog otvorenosti operacijskog sustava i ogromnog broja različitih verzija, što službenih, što promijenjenih od strane korisnika. Ovdje će se iznijeti neki karakteristični koraci za istragu bez komercijalnih alata, međutim u ovakvim istragama se postupci razlikuju od slučaja do slučaja i ovise o tome što se s istragom želi postići.

Prvo će se opisati tijek forenzičke istrage na Android uređaju od trenutka preuzimanja, zatim će se opisati procesi pribavljanja forenzičke slike Android UMS-a, particija sustava i memorije pokrenutih procesa. Na kraju će biti riječ o analizi pribavljenih podataka i pravljenju izvještaja nakon istrage. U 4. poglavlju pokazan je primjer istrage kroz formu izvještaja.

Prilikom nalaska uređaja koji nosi moguće potencijalne dokaze važne za slučaj, uređaj se prvo treba obraditi forenzički kao i svi ostali fizički forenzički dokazi, na primjer treba se provjeriti postoji li DNK i potražiti otiske prstiju [5]. Trebaju se potražiti originalni kablovi od uređaja ako postoji mogućnost za to. Tijekom cijelog procesa svaka promjena na uređaju treba se pomno bilježiti i dokumentirati, te je poželjno minimizirati diranje ekrana ako postoji mogućnost da bude potrebno napraviti tehniku *smudge attack* [19].

Slika 1 prikazuje tijek forenzičke istrage Android pametnog telefona. S manjim izmjenama preuzeta je iz rada „Acquisition and Analysis of Digital Evidence in Android Smartphones“ [7]. Čitav proces bit će opisan u nastavku.



Slika 1 Tijek Android forenzike [7]

Nakon fizičke obrade treba provjeriti je li uređaj upaljen. Ako nije upaljen treba provjeriti postoji li SD kartica. U slučaju da postoji, treba napraviti sliku podataka na njoj te napraviti identičnu kopiju kartice koja se potom vrati u mobitel. Mobitel se tada pokreće u sustavu za oporavak, te se priključuje preko USB kabela na forenzičku postaju. Tu treba pokušati uspostaviti vezu preko ADB-a naredbom `adb devices`. Ako nam naredba vrati serijski broj mobitela, znači da možemo nastaviti s logičkom ekstrakcijom podataka, a ako nam ne vrati ništa, mobitel pokrećemo u normalnom načinu rada. Prva stvar koju trebamo napraviti kad uključimo mobitel ili dobijemo mobitel koji je već uključen je mrežna izolacija. Andrew Hoog u knjizi Andorid forensics [1] navodi nekoliko načina mrežne izolacije od kojih svaka ima neke dobre i loše strane:

- Način rada u zrakoplovu (engl. *Airplane Mode*) je u većini slučajeva najbolji izbor, jer pravi izolaciju od svih mreža kao što su mobilne, Wi-Fi, Bluetooth, ali potencijalni minus mu je što mijenja stanje uređaja. Zbog toga se sve treba dobro dokumentirati. Tu opciju možemo uključiti na dva načina:
    - Postavke (engl. *Settings*) – Bežično povezivanje i mreže (engl. *Wireless and networks*) – Način rada u zrakoplovu (engl. *Airplane mode*)

- Pritisnuti i držati tipku za uključivanje/gašenje uređaja i zatim kliknuti na Način rada u zrakoplovu (engl. *Airplane mode*). Ovako se mogu mrežno izolirati i uređaji kojima je zaslon zaključan.

- Vađenje SIM kartice čime se uređaj izolira od mobilne mreže, međutim to ne utječe na Wi-Fi i Bluetooth mreže, a dodatni minus je to što u većine uređaja, za vađenje SIM kartice se treba izvaditi baterija što znači gašenje mobitela, gubljenje podataka iz RAM-a, moguća aktivacija šifriranja i/ili zaključavanja zaslona uređaja (PIN, gesture, password) ili čak i pokretanje brisanja čitave memorije.

- Može se zatražiti od mobilnog operatera da onemogući SIM karticu uređaja, ali time se postiže samo jedan aspekt mrežne izolacije.

- Faradayeva zaštita – vrećice, kutije, šatori i sobe. Učinkovitost vrećica je upitna, a najbolje rješenje je soba koja je jako skupa [1]. Omogućava izolaciju od svih mreža, međutim smanjuje trajanje baterije zato što se uređaj konstantno pokušava spojiti na mreže. Zbog toga ga je potrebno što prije priključiti na napajanje.

- Isključivanje uređaja je zadnja opcija koja je definitivno najučinkovitija u mrežnoj izolaciji, ali kao što je rečeno mijenja uređaj, brišu se privremene datoteke i postoji mogućnost aktiviranja šifriranja, zaključavanja ili automatskog brisanja svih podataka prilikom ponovnog pokretanja uređaja.

Ako je uređaj uključen i zaslon još uvijek aktivan treba se [1]:

- Produžiti vrijeme aktivnosti ekrana mobitela na neograničeno ili na maksimalno u postavkama mobitela. Ta mogućnost nije na svim Android uređajima na istom mjestu, ali je uglavnom na jednom od ova dva:

    - Postavke (engl. *Settings*) – Zvuk i zaslon (engl. *Sound and display*) – Mirovanje (engl. *Screen timeout*)

    - Postavke (engl. *Settings*) – Sigurnost (engl. *Security*) – Zaključaj nakon..(engl. *Lock phone after..*)

- Uključiti opcija traženja pogrešaka putem USB sučelja, koja će nam kasnije omogućiti povezivanje forenzičke postaje i mobitela preko ADB-a i akviziciju podataka. Ta opcija se nalazi u:

    - Postavke (engl. *Settings*) – Mogućnosti za razvojne programere (engl. *Applications and Development*)– Traženje pogrešaka putem USB sučelja (engl. *USB debugging*)

- Uključiti opciju *Stay Awake* da se za vrijeme punjenja ne gasi ekran. Ona se nalazi u:

    - Postavke (engl. *Settings*) – Mogućnosti za razvojne programere (engl. *Applications and Development*)– Ne pokreći mirovanje (engl. *Stay Awake*)

- Priključiti mobitel na napajanje.

Ukoliko zaslon nije aktivan prvo treba provjeriti da li se može pristupiti uređaju preko ADB-a na već opisan način. Ako može, nastavljamo s logičkom ekstrakcijom, a ako ne,

trebamo vidjeti možemo li pristupiti uređaju i uključiti traženje pogrešaka putem USB sučelja prema već opisanom načinu.

Uređaj može biti i zaključan, te tako pristup postavkama nemoguć. Postoji više vrsta ključeva – uzorak, PIN i alfanumerička lozinka. Uzorak (engl. *Pattern lock*) ima najmanji broj kombinacija i najlakše ga je probiti, dok je alfanumerička lozinka najjača. Međutim, nikad se ne ide pogađati lozinka jer uređaj može imati mehanizam koji nakon određenog broja neispravno unesenih lozinki briše sve u svojoj memoriji. Ukoliko na uređaju postoji zaključavanje uzorkom može se pokušati tehnika *Smudge attack* [19] tj. slikanje mobitela pod različitim kutevima, osvjetljenjem i različitim postavkama fotoaparata, da bi se dobio uzorak iscrtavanja koji je ostao od zadnjeg otključavanja mobitela. Također, ako postoji mogućnost, treba se potražiti lozinka među stvarima vlasnika uređaja ili je dobiti od samog vlasnika. Postoji još par tehnika otključavanja uređaja, a neke od njih su:

- Tražiti od Google da ponovno postavi lozinku, međutim, tu moramo biti priključeni na mrežu, što nije dobro jer može doći signal za primjerice brisanje svih podataka.
- Ako su poznati podaci gmail računa i mobitel je na mreži, možemo ući na Google Play trgovinu preko računala i daljinski instalirati aplikaciju *Screen lock bypass od Thomasa Cannona* [20]. Nakon što se instalira treba se aktivirati, primjerice na način da se uređaj stavi na punjenje, što pokrene aplikaciju preko koje se može ukloniti zaključavanje. Ta se aplikacija može instalirati preko USB kabela, ukoliko postoji mogućnost spajanja preko ADB-a, te se za aktivaciju instalira još jedna aplikacija po izboru.
- Postoji i način otključavanja mobitela, koji se ne preporučuje zbog mogućnosti pokretanja automatskog brisanja čitave memorije uređaja, ali ako su poznati podaci gmail računa, može se iskoristiti maksimalan broj pokušaja unošenja lozinke i onda će zaslon prikazati da unesemo adresu elektroničke pošte i lozinku i otključat će se uređaj.

Ukoliko ipak nikako ne možemo uključiti traženje pogrešaka putem USB sučelja, treba mobitel pokrenuti u *sustavu za oporavk* (engl. *recovery*). U njemu se mogu naći alati za popravak operacijskog sustava, brisanje svih korisničkih podataka, povratak na početne postavke, kao i instaliranje službenih nadogradnji. Da bi se pristupilo sustavu za oporavak, prilikom podizanja sustava treba držati kombinaciju nekih tipaka. Za različite mobitele su potrebne različite kombinacije tipki i one se u većini slučajeva mogu pronaći na Internetu. Većina sustava za oporavak koji su došli s mobitelom i nisu mijenjani, neće nam biti od velike koristi, međutim, ako je sustav za oporavak promijenjen, administratorski pristup a time i ADB, bi mogli biti omogućeni. Promijenjeni sustav za oporavak koji je poznat pod imenom prilagođeni (engl. *custom*) sustav za oporavak se uglavnom stavlja na uređaj zbog prilagođenih (engl. *custom*) ROM-ova, nekih aplikacija, dodatnih mogućnosti uređaja, razvoja programa itd. Omogućuju pravljenje rezervnih kopija sustava i obnavljanje iz njih, selektivno brisanje podataka, instaliranje neslužbenih nadogradnji itd. Najpoznatiji je ClockWorkMod, a tu su i TWRP, Amon Ra i drugi. [13]

U slučaju da ne možemo pristupiti uređaju preko ADB-a, treba se posavjetovati s nadležnom osobom oko toga što napraviti sljedeće. Može se izvaditi kartica, napraviti njena slika te analizirati podaci na njoj, pokušati dobiti ADB i administratorski pristup instaliranjem prilagođenih sustava za oporavak ili nekim drugim alatima te izvući podatke fizičkim metodama ekstrakcije kao što su JTAG i chip-off.

Nakon što smo dobili ADB pristup, trebamo napraviti ekstrakciju podataka s memorijske kartice i unutarnje memorije uređaja. Bez administratorskog pristupa možemo izvući neke informacije o sustavu preko naredbi kao što su `dmesg`, `dumpsys`, `logcat`, te dostupnih particija. Ukoliko imamo administratorski pristup radimo sliku svih particija sustava, te ako postoje pokrenute aplikacije koje potencijalno sadrže korisne informacije treba napraviti sliku njihove memorije.

Ako nemamo administratorski pristup, a potreban nam je, možemo ga probati dobiti pokretanjem mobitela u sustavu za oporavak. Ukoliko ga ni tada ne dobijemo, instaliranje prilagođenog sustava za oporavak je forenzički najbolji način za njegovo omogućavanje. Zatim su tu alati za dobivanje privremenog i trajnog administratorskog pristupa kao što su debugfs i psneuter, te alati koji su specifični za različite proizvođače uređaja koji inače služe za popravku telefona, ali se mogu iskoristiti i za dobivanje administratorskog pristupa. To su npr. Motorolin RSD Lite, sbf_flash, Samsungov Odin Multiloader i drugi. Treba uzeti u obzir da se ovim tehnikama radi pisanje na uređaj, što ga mijenja. Zbog toga treba postojati opravdan razlog za ove tehnike, prethodno testiranje na zamjenskom uređaju koji je isti kao i uređaj na kojem se vrši istraga te dobra i detaljna dokumentacija čitavog procesa.[1]

Ako uređaj nije šifriran mogu se napraviti JTAG ili chip-off tehnike, međutim obje su jako komplicirane i zahtijevaju stručno osoblje te postoji mogućnost da se uređaj ne može ponovno vratiti u funkcionalno stanje, posebno u slučaju chip-off tehnike. U ovim tehnikama se fizički prepisuje bit-po-bit memorija uređaja i zahtijevaju puno opsežniju i kompliciraniju analizu nego što je to slučaj s logičkim tehnikama.

3.1 Procesi izrade forenzičkih slika

U pojedinim dijelovima procesa provođenja forenzike na Android uređaju, kao što je već spomenuto, potrebno je napraviti forenzičku sliku dijela njegove memorije. Forenzička slika je bit-po-bit kopija čitave memorije ili njenoga dijela, tako da su kopija i ono što je kopirano identični. U nastavku će biti objašnjeno kako napraviti forenzičku sliku Android USB masovne pohrane (engl. *Android USB Mass Storage*, UMS) i particija sustava, te kako pribaviti memoriju pokrenutih procesa.

Pod *Android USB masovnom pohranom* podrazumijevaju se vanjska *Secure Digital* (SD) kartica i ugrađena *Embedded MultiMediaCard* (eMMC). One su jako slične po izvedbi, kao što je spomenuto, obadvije koriste NAND memoriju, bazirane su na *MultiMediaCard* specifikacijama i imaju ugrađene kontrolere memorije. Njihova glavna razlika je to što je eMMC integriran u čip i ne može se izvaditi bez rastavljanja uređaja, a SD kartica se može lako prenositi s jednog uređaja na drugi.

Forenzičku sliku Android UMS uređaja radimo preko UMS sučelja ili `dd` naredbom i adb prosljeđivanjem pristupa. Prije se slika SD kartice radila tako da bi se ona izvadila, te bi se napravila slika preko USB blokatora pisanja. To se više ne preporučuje, osim ako je uređaj već ugašen. Gašenjem pametnog telefona se gube korisni podaci koji se nalaze u RAM memoriji uređaja. Danas se sve više aplikacija pokreće sa SD kartice, i za vrijeme dok je pokrenuta, njeni podaci na kartici nisu šifrirani. Kad se aplikacija ugasi, nad tim podacima se aktivira šifriranje i više ne možemo doći do njih.

Slika se može napraviti `dd` naredbom, ali ju je bolje napraviti sa dc3dd programom koji je otvorenog koda i omogućava dodatne opcije koje su od velike pomoći u Android forenzici kao što je hashiranje, verifikacija dokumenta i druge.[1] Sam proces opisan je u nastavku.

Preuzimanje i instalacija dc3dd programa se može napraviti sljedećim nizom naredbi u Linuxu:

```
mkdir -p ~/src
cd ~/src
curl
http://sourceforge.net/projects/dc3dd/files/dc3dd/7.1.0/
dc3dd-7.1.614.tar.gz  > dc3dd-7.1.614.tar.gz
tar xzf dc3dd-7.1.614.tar.gz
cd dc3dd-7.1.614/
./configure
make
sudo make install
```

Poželjno je napraviti skriptu koja skraćuje pisanje dc3dd naredbe i spremiti je u */usr/local/bin* direktorij kako bi mogla biti pokrenuta iz bilo kojeg direktorija. Skripta koja se nalazi u dodatku A kao i postupak pravljenja slike UMS uređaja preuzeti su sa stranice viaForensics [8]. Naredbom

```
sudo nano -w /usr/local/bin/acquire-disk.sh
```

otvaramo program nano za uređivanje teksta. Sljedeće treba kopirati skriptu i spremiti ju, te joj promijeniti dozvole.

```
sudo chmod 755 /usr/local/bin/acquire-disk.sh
sudo /usr/local/bin/acquire-disk.sh
```

Uređaj bi trebao biti spojen na forenzičku postaju preko blokatora pisanja. Nakon što se omogući UMS sučelje na uređaju treba pokrenuti naredbu `dmesg` i provjeriti koje su particije SDcard i eMMC. Naredba se poziva sa:

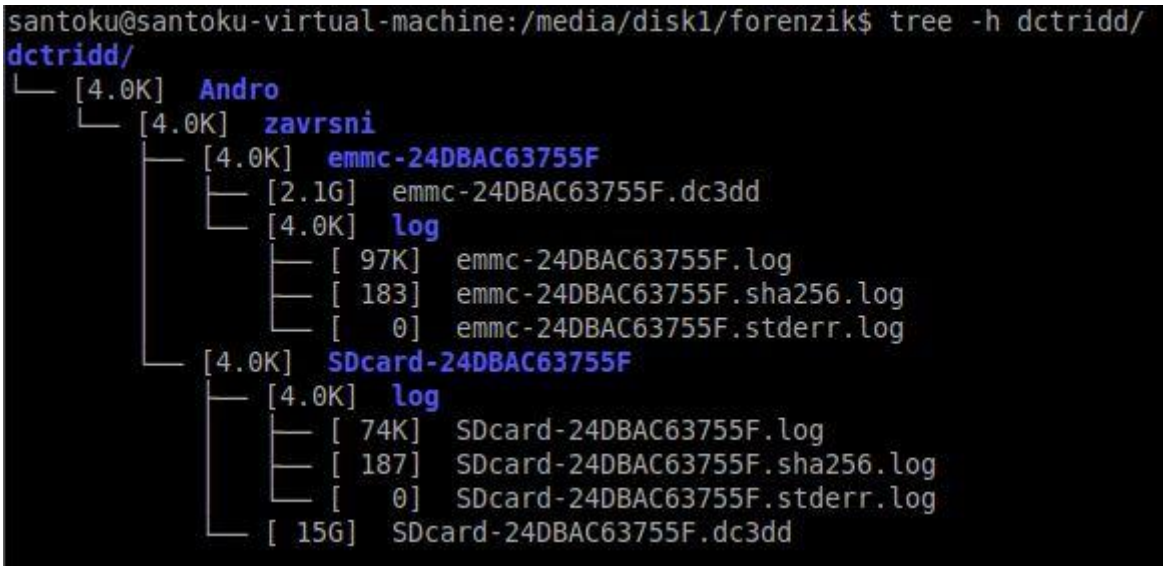
```
sudo   acquire-disk.sh   client_name   case_name   tag
serial_number sourcedev destpath
```

gdje su parametri *client_name* gdje se unosi ime datoteke u koju će se spremiti čitava akvizicija, *case_name* – ime slučaja, *tag* – broj dokaza kojem pravimo forenzičku sliku, *serial_number* – serijski broj uređaja, *sourcedev* – adresa izvorišnog uređaja, *destpath* – adresa na kojoj će se spremiti dobiveni podaci. Primjer:

```
sudo   acquire-disk.sh   forenzika   zavrsni   predmet01
unknown-serialno /dev/sdc ~/sd-emmc
```

Struktura direktorija može se provjeriti sa:

```
tree -h sd-emmc/
```



Slika 2 Struktura direktorija dobivenog skriptom za dc3dd

Ovim postupkom je napravljena forenzičku slika UMS uređaja, zajedno sa sažetkom za dd sliku, log datoteku s podacima o datumu, vremenu, informacijama o sustavu i naredbama provedenim, stderr.log gdje su upisane pogreške, te sha256.log s popisom svih log datoteka i njihovih sha256 sažetaka. Rukovanje dd slikama će biti objašnjeno malo kasnije u poglavlju.

Slike particija sustava se rade nakon što je napravljena slika UMS uređaja. Da bi se mogle napraviti slike svih particija potrebne su administratorske ovlasti. Najvažnije particije za ispitati po knjizi Android Forensics [1] su prikazane u tabeli 1.

Tabela 1 Najvažnije particije za ispitivanje

/proc	podaci o sustavu
/data/data	podaci aplikacija
/data	podaci aplikacija i sustava
/cache	cache podaci od sustava i nekih aplikacija
/mnt/asec	dekriptirane .apk datoteke aplikacija koje su spremljene kriptirane na SD kartici, ali ovdje služe za komunikaciju pokrenute aplikacije sa sustavom
/app-cache	cache podaci nekih aplikacija
/mnt/sdcard	Sdcard
/mnt/emmc	eMMc

Postupak [7] kreće pristupanjem naredbenom retku (engl. *command line*) s administratorskim pristupom naredbama `adb shell` i `su`. Sljedeće, s naredbom `mount` se provjeri koji su datotečni sustavi prisutni na uređaju. Particije se nalaze u */dev/mtd/*

direktoriju i možemo ih izlistati s naredbom `ls /dev/mtd/mtd*`. Sliku particija pravimo naredbom `dd`, gdje se pod *if* upisuje put do datoteke iz koje se čita, pod *of* put datoteke u koju se piše, a pod *bs* broj bajtova po koliko će se odjednom čitati tj. pisati. Primjer naredbe:

```
dd if=/dev/mtd/mtd6ro of=/mnt/sdcard/mtd6ro_system.dd
bs=4096
```

Tom naredbom se sprema slika particije na memorijsku karticu, a prebacivanje na forenzičku postaju se radi sa naredbom `adb pull`.

Drugi način pravljenja slike particije [1] je preko prosljeđivanja pristupa (engl. *port forwarding*), gdje se podaci ne spremaju na memorijsku kartici, nego se prosljeđuju na određena vrata na računalu. To se radi tako da se na računalu otvore dva terminala. Prvo se u jednom terminalu omogući veza između dvije točke:

```
adb forward tcp:31333 tcp:31333
```

Zatim se u drugom, s naredbama `adb shell` i `su` pokrene terminal na Android uređaju s administratorskim pristupom. Pravljenje bit-po-bit kopije memorije i prosljeđivanje na netcat se radi s naredbom:

```
dd if=/dev/mtd/mtd6 bs=4096 | /dev/username/nc –l –p
31333
```

I na kraju se treba u prijašnjem terminalu namjestiti da se računalo spoji na vrata na Android uređaju i primi proslijeđenu datoteku:

```
nc 127.0.0.1 31333 > dd of=userdata.dd bs=4096
```

Dobivene dd slike se trebaju dodati kao datotečni sustavi (engl. *mount*) da bi se iz njih izvukli logički podaci, a to nije uvijek jednostavno. Može se napraviti i logička ekstrakcija koja odmah povlači logičke podatke s uređaja, međutim particija mora biti dodana na uređaju. Particije koje se pribavljaju sa uređaja bi trebale biti postavljene da se sa njih može samo čitati. To se radi naredbom `mount` sa opcijama *–o ro* koje postavljaju particiju samo za čitanje i *–t* nakon kojeg dođe datotečni sustav i na kraju put do particije. Primjer:

```
mount –o ro –t yaffs2 /dev/block/mtdblock8 /data
```

Ako je već dodana kao datotečni sustav, dodaje se opcija *remount*:

```
mount –o ro, remount –t yaffs2 /dev/block/mtdblock8
/data
```

Na kraju se datoteke povuku sa `adb pull`. Primjer:

```
adb pull /data/data/com.android.providers.telephony sms
```

Ako je potrebno, mogu se izvući *podaci koje koriste još uvijek pokrenute aplikacije* [7]. Oni mogu sadržavati korisne podatke kao što su lozinke, korisnička imena, kriptografski ključevi, razne informacije zapisane u bazama podataka itd. Za to je potreban administratorski pristup na uređaju.

Prvo se treba spojiti sa `adb shell` i naredbom `su`. Da bi se saznao identifikacijski broj procesa koristi se naredba ps. Zatim je potrebno promijeniti dozvole direktorija */data/misc*:

```
chmod 777 /data/misc
```

Potom se ubija proces sa:

```
kill -10 pid
```

Gdje je pid je identifikacijski broj procesa. Nakon što se proces ubije, stvara se datoteka sa slikom memorije tog procesa u */data/misc* direktoriju. Na kraju tu datoteku možemo prenijeti na forenzičku postaju s naredbom `adb pull`.

3.2 Analiza prikupljenih podataka i izvještaj

Nakon akvizicije podataka sa Android uređaja potrebno je napraviti analizu onog što je prikupljeno. Kod analize dobivenih podataka treba prvo odlučiti gdje postaviti fokus. U nekim istragama će naprimjer fokus biti slike, videozapisi i geolokacije, dok će kod drugih možda biti bankovni podaci, transakcije, sms-ovi i pozivi. Zbog toga treba odlučiti u kojem će se smjeru ići i koliko duboko. Zatim treba, ukoliko već nije, povezati vlasnika uređaja sa ispitivanim uređajem. To se može postići primjerice analizom SMS-ova, google računa, poziva ili podataka iz raznih aplikacija. Potom se kreće na analizu podataka dobivenih sa UMS uređaja i podataka prikupljenih sa telefona. Na kraju se dokumentiraju alati i tehnike korištene u analizi i pribavljanju podataka, te pravi izvještaj.[7]

Poslije svake forenzičke istrage treba sastaviti detaljan izvještaj koji opisuje sve napravljene korake i zaključke izvučene na kraju istrage. Izvještaj treba sadržati informacije o stanju uređaja, svim akcijama poduzetim tijekom rukovanja uređajem i u obradi dokaza, rezultate tih akcija i objašnjenje zaključaka izvedenih iz dokaza. Dokazi, te alati, tehnike i metode korištene u istrazi mogu biti preispitivane na sudu, zato je potrebno čitav postupak istrage dobro dokumentirati. U slučaju da se pozove neovisna treća strana da provjeri vjerodostojnost istrage, ona mora doći do istih rezultata kao i istražitelji provodeći postupke opisane u izvještaju. Što bi izvještaj prema NIST-u trebao sadržavati može se pronaći u dodatku D.

4. Primjer forenzičke istrage

Identitet agencije koja pravi izvještaj: Završni

Identifikacijski broj slučaja: 001

Istražitelj koji je zadužen za slučaj: Valerija Šimić

Identitet osobe koja je preuzela uređaj: Valerija Šimić

Datum preuzimanja uređaja: 21.5.2014.

Datum izvješća: 22.5.2014.

Opisna lista stvari dostavljenih na ispitivanje:

USB kabel i mobitel Huawei Ascend g300 24DBAC63755F sa Android 4.0.3 OS-om, pokrenut u normalnom načinu rada sa zaključavanjem zaslona metodom uzorka, kućište blago oštećeno.

Identitet i potpis ispitivača: Valerija Šimić

Oprema i okruženje korišteno u ispitivanju:

-Santoku Linux virtualna mašina pokrenuta na VMWare playeru,na laptopu hp ProBook 4530s

-Originalni USB kabel od mobilnog uređaja

-Canon PowerShot SX200IS za pribavljanje slika uređaja i tehniku 'Smudge attack'

Akvizicija podataka:

Uređaj je preuzet u pokrenutom stanju, te je odmah napravljena mrežna izolacija tako da je uključen način rada u zrakoplovu držanjem tipke za isključivanje i odabirom željene opcije iz pokrenutog izbornika. Zatim je napravljena tehnika 'Smudge attack' (slika 3). Mobitel je slikan sa različitim osvjetljenjem da bi se pronašao, ukoliko postoji, ostatak uzorka za otključavanje zaslona. Uzorak je uspješno dobiven. Mobitel je potom spojen na forenzičku postaju, te sa naredbom 'sudo adb devices' je provjereno da li se može spojiti preko Android Debug Bridge-a. Naredba nije prikazala niti jedan uređaj. Zbog toga, na mobitelu je uključeno uklanjanje programske podrške i opcija ne pokreći mirovanje, preko izbornika postavke sustava -> razvojne opcije (slika 4). Zatim su uslijedile akcije:

- -pravljenje slike unutrašnje i vanjske memorijske kartice pomoću programa dc3dd i skriptom acquire-disk.sh preuzetom sa stranice viaForensics (dodatak A)
- -logička akvizicija pomoću alata AFLogical-ose
- -provjeravanje administratorskog pristupa koji nije postojao
- -ispitivanje /proc particije pomoću naredbe cat
- -logička akvizicija SDkartice i eMMc uređaja naredbom 'adb pull'

Čitav postupak prikazan je na slici 5.

Dobivene datoteke:

- -slike uređaja na početku procesa
- -datoteke u dctridd dobivene skriptom acquire-disk.sh
- -datoteke dobivene AFLogical-ose
- -datoteke dobivene naredbom 'adb pull'

Detalji rezultata istrage:

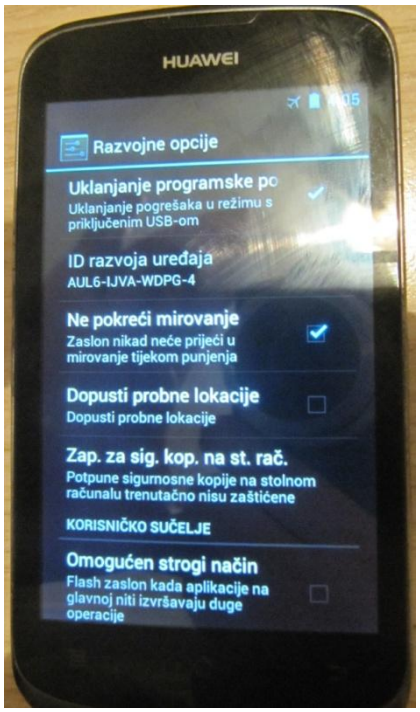
-Akvizicijom podataka pomoću AFLogical programa, dobili smo 1MMS, 1451 SMS i 501 poziv. Utvrdili smo tko je vlasnik uređaja iz brojnih poruka koje su poslane i dostavljene na mobitel tijekom zadnje godine. Također znamo s kim vlasnik najviše komunicira i u kakvim je odnosima sa svojim kontaktima, te još neke detalje o njemu.

-Ispitivanjem podataka dobivenih sa naredbom adb pull naišli smo na HuaweiBackup direktorij koji je sadržavao podatke o kontaktima, alarmima, Web oznakama, kalendaru i podsjetnicima, zapisima poziva, glazbene liste, SMSove i sistemske postavke. U tom direktoriju su pronađene 3 zip datoteke backupa različitih datuma nastanka. U direktoriju WhatsUp su pronađene slike koje su slane putem te aplikacije. Zatim, pronađene su slike u direktoriju DCIM koje ukazuju na vlasnika. Također su nađene i druge korisnikove datoteke pod Documents, Downloads, Music i Books.

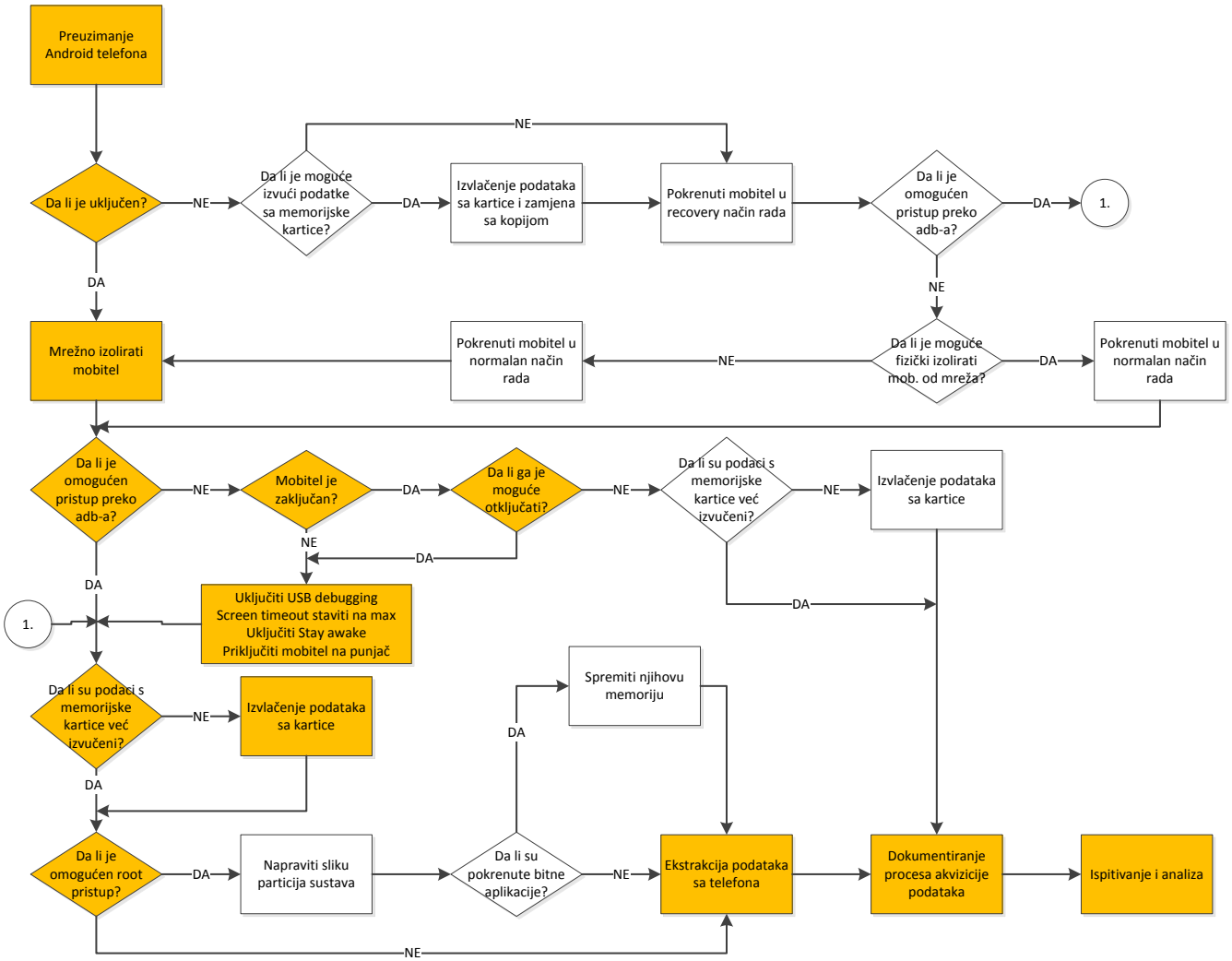
Zaključak izvještaja:

Nepotrebna daljnja analiza uređaja jer je izvučeno dosta podataka i nema opravdanih razloga za nastavak procesa. Također smo uspjeli povezati vlasnika sa uređajem.

Slika 3 Otkrivanje uzorka za otključavanje



Slika 4 Pokretanje opcija na uređaju



Slika 5 Tijek istrage

5. Zaključak

Svaki slučaj je slučaj za sebe i donosi različite izazove. Tako se primjerice u jednoj istrazi može dogoditi da se ne može pristupiti nikakvim podataka, a u drugoj da se može izvući praktički sve. Nakon završetka pisanja ovog rada gdje sam opisala proces Android forenzike i nakon što sam napravila vlastitu forenzičku istragu na pametnom telefonu sa Android operacijskim sustavom, mogu reći da je područje Android forenzike jako kompleksno. Najviše zbog mnogobrojnih različitih verzija i kombinacija Android operacijskih sustava i pametnih telefona, te načina pristupanja podacima koji se razlikuju od uređaja do uređaja. Jedan veliki problem je i sam proces dokazivanja vjerodostojnosti dobivenih podataka jer se u velikoj većini slučajeva ne može proći bez mijenjanja dobivenog uređaja sa kojeg pribavljamo dokaze. Mislim da se ovo područje nikada neće moći standardizirati do točke da postoji jedinstven proces forenzičke istrage na Android uređajima, jer se ta grana industrije razvija prebrzo, i dok se jedan uređaj analizira, lansira se mnoštvo novih mobitela, sa novim mogućnostima i novim verzijama Android operacijskih sustava, a pokraj toga su i korisnici koji imaju mogućnost razvijanja Androida i prilagođavanja svojim potrebama.

6. Literatura

1. Andrew Hoog, *Android forensics*, Sjedinjene Američke Države: Syngress, 2011
2. Konstantia Barmpatsalou, Dimitrios Damopoulos, Georgios Kambourakis, Vasilios Katos, *A critical review of 7 years of Mobile Device Forensics*, Digital Investigation 10, 323-349str., Elsevier Ltd., 2013
3. ACPO *Good Practice Guide for Computer-Based Electronic Evidence*, 7safe, www.7safe.com/electronic_evidence (4.5.2014.)
4. SWGDE *Best Practices for Mobile Phone Forensics Version: 2.0*, SWGDE, 11. veljače 2011.
5. Rick Ayers, Sam Brothers, Wayne Jansen, *Guidelines on Mobile Device Forensics* Revision 1 (Draft), NIST, rujan 2013.
6. Jerry Hildenbrand, *What is recovery?* http://www.androidcentral.com/what-recovery-android-z 1. veljače 2012. (1.6.2014.)
7. André Morum de L. Simão, Fábio Caús Sícoli, Laerte Peotta de Melo, Flávio Elias de Deus, Rafael Timóteo de Sousa Júnior, *Acquisition and Analysis of Digital Evidence in Android Smartphones*, IJoFCS 1, 28-43str., 2011. http://dx.doi.org/10.5769/J201101002 (10.6.2014.)
8. Andrew Hoog, *Android forensics*: Imaging Android USB Mass Dtorage Devices https://viaforensics.com/resources/reports/android-forensics/imaging-android-usb-mass-storage-devices/#how-to-forensically-image-sd-card (4.5.2014.)
9. Bill Anderson, *Understanding the Android File Hierarchy*, 13. siječnja 2013. http://www.all-things-android.com/content/understanding-android-file-hierarchy
10. Wikipedia, *sysfs*, 8. svibnja 2014. http://en.wikipedia.org/wiki/Sysfs (10.6.2014.)
11. Cellebrite *UFED*, http://www.cellebrite.com/mobile-forensics (10.6.2014.)
12. Compelson *Mobiledit*, http://www.mobiledit.com/forensic (10.6.2014.)
13. *EnCase Neutrino*, https://viaforensics.com/resources/white-papers/iphone-forensics/encase-neutrino/ (10.6.2014.)
14. MicroSystemation *XRY Logical*, http://www.msab.com/xry/xry-logical (10.6.2014.)
15. Viaforensics *Aflogical OSE*, https://github.com/viaforensics/android-forensics (10.6.2014.)
16. Michael Spreitzenbarth, *Android Data Extractor Lite*, http://forensics.spreitzenbarth.de/adel/ (10.6.2014.)
17. *Android Debug Bridge* http://developer.android.com/tools/help/adb.html (10.6.2014.)
18. Kevin Swartz, *HOWTO Complete a Logical Examination on Android Using F/OSS Tools: Webinar Video*, 30. rujna 2013. https://viaforensics.com/android-

forensics/howto-complete-logical-examination-android-foss-tools-webinar-video.html (4.5.2014.)

19. Adam J. Aviv, Katherine Gibson, Evan Mossop, Matt Blaze, and Jonathan M. Smith, *Smudge Attacks on Smartphone Touch Screens*, https://www.usenix.org/legacy/event/woot10/tech/full_papers/Aviv.pdf (10.6.2014.)
20. Thomas Cannon, *Screen Lock Bypass Reset*, 28. svibnja 2012. https://play.google.com/store/apps/details?id=net.thomascannon.screenlockbypass.fix (11.6.2014.)
21. Wikipedia, *Forenzika*, 15. kolovoza 2013. http://hr.wikipedia.org/wiki/Forenzika (11.6.2014.)
22. INsig2, *Digitalna forenzika* http://www.insig2.hr/digitalna-forenzika-d2-5?lang=hr (11.6.2014.)
23. Valerija Šimić, *Android Forensic Tools*, 23. ožujka 2014. http://sgros-students.blogspot.com/2014/03/android-forensic-tools.html (10.6.2014.)
24. Valerija Šimić, *Open source Android forensic tools*, 23. ožujka 2014. http://sgros-students.blogspot.com/2014/03/open-source-android-forensics-tools.html (10.6.2014.)
25. Valerija Šimić, *LiME*, 6. travnja 2014. http://sgros-students.blogspot.com/2014/04/lime.html (10.6.2014.)

Naslov, sažetak, ključne riječi

Proces provođenja forenzičke analize Android operacijskog sustava

U ovom radu je obrađen proces forenzičke analize uređaja sa Android operacijskim sustavom. U početku su spomenute smjernice kojim bi se trebale voditi osobe koje provode forenzičku istragu u sklopu digitalne forenzike. Potom je ukratko rečeno o vrstama forenzičkih istraga koje se tiču Android forenzike, podacima koji se mogu naći na uređajima, vrstama memorije i datotečnim sustavima na Androidima te tehnikama akvizicije podataka sa Android uređaja. Zatim je obrađen proces same forenzičke istrage uključujući procedure rukovanja uređajem, pribavljanje podataka sa UMS uređaja, particija sustava i od pokrenutih aplikacija, analizu podataka i izvještaj, te na kraju i primjer napravljene forenzičke istrage opisane kroz izvještaj.

Ključne riječi: Android, forenzika, pametni telefoni,

Title, summary, keywords

The process of conducting a forensic analysis of the Android operating system

This paper describes the process of forensic analysis of devices with the Android operating system. In the beginning are guidelines that should guide officers who are conducting forensic investigation within the digital forensics. Then, there is something about types of forensic investigations concerning Android devices, data that can be found on them, types of memory, files systems and techniques of data acquisition on Android devices. Next, it contains the description of the process of forensic investigation, including procedures for handling Android smartphones, obtaining data from the UMS devices, system partitions and running applications, data analysis, report, and finally on the end, an example of forensic investigation that has been made and described through the report.

Ključne riječi: Android, forensics, smartphones

Dodatak A: Shell skripta za dc3dd [1]

U ovom dodatku nalazi se skripta korištena i opisana u poglavlju 3.1.

```
#!/bin/bash

CLIENT="${1}"
CASE="${2}"
TAG="${3}"
SERIALNO="${4}"
SOURCEDEV="${5}"
DESTPATH="${6}"

OUTPUTPATH=$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO
LOGFILE=$OUTPUTPATH/log/$TAG-$SERIALNO.log
STDERRLOG=$OUTPUTPATH/log/$TAG-$SERIALNO.stderr.log
SEPERATOR="-------------------------------------------\r"

if [ "$#" != 6 ]; then
        echo "Usage: acquire_disk.sh CLIENT CASE TAG SERIALNO SOURCEDEV
DESTPATH"
        exit 2
fi

# check directories, created if needed
if [ ! -d "$DESTPATH" ]; then
        echo "Destination path [$DESTPATH] does not exist, exiting"
        exit 1
fi

if [ -d "$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO" ]; then
        echo "$DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO already exists,
can't overwrite evidence"
        exit 1
fi

GOTROOT=`whoami`

if [  "$GOTROOT" != "root" ]; then
       echo "must be root to execute"
       exit 1
fi

mkdir -p $OUTPUTPATH/log

echo -e "Start date/time" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`/bin/date`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "uname -a" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
```

```
echo -e "`uname -a`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "dmesg | tail -50" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`dmesg | tail -50`\n" >> $LOGFILE 2>> $STDERRLOG

echo -e "lshw" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`lshw`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`fdisk -v`
echo -e "fdisk -l $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`fdisk -l $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`mmls -V`
echo -e "mmls $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`mmls $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`fsstat -V`
echo -e "fsstat $SOURCEDEV [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`fsstat $SOURCEDEV`\n" >> $LOGFILE 2>> $STDERRLOG

VERSION=`dc3dd --version 2>&1 | grep dc3dd`
echo -e "dc3dd [$VERSION]" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "dc3dd if=$SOURCEDEV of=$OUTPUTPATH/$TAG-$SERIALNO.dc3dd verb=on
hash=sha256 hlog=$OUTPUTPATH/log/$TAG-$SERIALNO.hashlog
log=$OUTPUTPATH/log/$TAG-$SERIALNO.log rec=off\n" >> $LOGFILE
dc3dd if=$SOURCEDEV of=$OUTPUTPATH/$TAG-$SERIALNO.dc3dd verb=on
hash=sha256 hlog=$OUTPUTPATH/log/$TAG-$SERIALNO.hashlog
log=$OUTPUTPATH/log/$TAG-$SERIALNO.log rec=off

echo -e "ls -lR $DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`ls -lR $DESTPATH/$CLIENT/$CASE/$TAG-$SERIALNO`\n" >> $LOGFILE

echo -e "End date/time" >> $LOGFILE
echo -e "$SEPERATOR" >> $LOGFILE
echo -e "`/bin/date`\n" >> $LOGFILE

#sha256sum all log files
cd $OUTPUTPATH/log/
sha256sum * > $TAG-$SERIALNO.sha256.log
```

Dodatak B: Android forensic tools [23]

According to IDC Worldwide Mobile phone tracker, Android has dominant position in the smartphone market with the share of over 80%. Because of its power on mobile market and applications who are far beyond just calling and texting, Android phones have potential to be in the middle of many investigations, as they potentially keep a lot of useful information. That is why it is very important to have good forensic support for finding and processing evidences on Android systems.

To facilitate their work and automate the process of forensic investigation, investigators use forensic tools. Forensic tools help identify, preserve, extract, analyze and document digital evidence. There are many tools available for computer forensics, but for Android that isn't the case. The main reason is a huge number of different versions of the Android operating system issued by different manufacturers, and those issued by users.

There are commercial tools, some of which have free versions available for download, and open source tools. Most popular commercial tools are:

- XRY
- Cellebrite UniversalForensic Extraction Device (UFED)
- Oxygen Forensic® Suite 2013 (free version Free Oxygen Forensic® Suite 2013(Standard))
- viaExtract (trial version)
- MOBILeditForensic (trial version)
- SAFT (free version)
- EnCaseForensic

Open source license means that software is freely redistributable, access to the source code is provided, allows the end user to modify the source code at will, and doesn't restrict the software's end use. Some open source tools:

- Opensource Android Forensics Toolkit (OSAF-TK)
- AFLogical™ Open SourceEdition
- Santoku
- Andorid-keystore
- Android Pattern Lock Cracker
- ADEL
- The Sleuth Kit
- LiME
- Volatility

- bulk_extractor

The last four, LiME, Volatility, The Sleuth Kit and bulk extractor, are forensic tools that are not only used for Android forensics, but also for forensics of other computer systems.

Comparison of some android forensic tools

In paper ″Android Forensic Capabilityand Evaluation of Extraction Tools″, Vijith Vijayan, among other things, evaluated the following forensic tools: Oxygen Forensic, MOBILedit Forensic, AFLogical and manual extraction when tested with actual forensic data. Tools were tested on HTC Desire and HTC Sensation XE. His conclusions were:

- **Oxygen forensic** was installed on the forensic workstation with a registration key provided by the company. It requires the mobile device to be in USB debugging mode while connecting to the workstation. Oxygen forensic extracted all of the contacts, call logs, images, files and system information, but of 64 audio files it extracted only 39. It also didn't produce the SMS/MMS list. In the final report produced by the tool SMS/MMS section was noted as 'Section not found'.

- **MOBILedit,** he worked with was lite version, downloaded from the Internet. MOBILedit needs to have the USB debugging mode enabled in the mobile phones, like Oxygen forensic. Phones can be connected to computer with cable or through wireless connection. This tool installed a small application on the mobile phone to pull the data. MOBILedit extracts the contacts, system-info, call logs and messages, but several contacts and call logs were missed in this test.

- **AFLogical Open Source Edition** was the next tested tool. This tool is an open source version of AFLogical. It's the lightweight software with no graphical front end, used from the command line, unlike the tools tested before. The Android phones were connected to the forensic workstation with enabled USB debugging mode. SD card has to be removed before extraction because ViaForensic warns that the contents may be deleted in the process. AFlogical needs ADB to communicate with the Android devices. The AFlogical was accurate in displaying the contacts, call lists, and messages in a nice readable csv format, however, there is no support for other data.

- **Manually extracting data** is different from the way how the previous software tools for Android forensics do it. He created images of the SD cards in dd file format using FTK imager, then opened them in the hex editor and searched for various keywords. The idea was to find out the system informations from the images, but manual extraction of data didn't give expected outcome. Keyword search resulted in no image files and documents. More number of images were found using manual search. This method was the most time consuming of all for him, but I'm not sure is that because of imaging SD cards or searching for keywords.

Analysing the result presented above, it can be concluded that Oxygen forensic was able to extract most of the actual data, while MOBILedit was the worst. But, we must take into account that the Oxygen tool in this testing was full version and MOBILedit and viaForensic both were lite versions.

Dodatak C: Open source Android forensics tools [24]

This post summerizes the results of experiments with the open source forensic tools for Android devices. All the experiments were done using SE Xperia 8, which was rooted and with installed Android version 2.3.7 and Clockworkmod recovery on it.

Android Data Extractor Lite

Android Data Extractor Lite (ADEL) requires a phone to be rooted and insecure kernel or a custom recovery installed on it. It also needs a predefined configuration for each device to be present in *xml/pfone_configs.xml* file. Currently there is only configuration for the Samsung Galaxy S2 with Android 2.3.3 available at the moment this post was written. Although there was no configuration for SE Xperia 8, ADEL was able to extract some information from the device.

It isn't hard to start the program. First, you have to download the source and unpack it. Next, connect the phone using USB cable to the computer. In this particular case OSAF Linux was installed within a VMWare Player, and this requred additional step of connecting phone to virtual machine. I did that in VMWare Player's settings under Removable devices. ADEL also needs ADB installed on the computer, i.e. OSAF Linux. The first problem I came across was the lack of the path to ADB. Solution to that problem is to type the following command in terminal:

```
export PATH=$PATH:<path to sdk>/sdk/platform-tools/adb
```

note that you have to change <path to sdk> into exact directory you've unpacked the SDK. For example:

```
export   PATH=$PATH:/home/username/android/sdk/platform-
tools/adb .
```

From the terminal position yourself in the directory ADEL-master, where you should have previously unpacked ADEL archive, and start the program using command:

```
adel.py -d device -l 4
```

ADEL execution ended with the message:

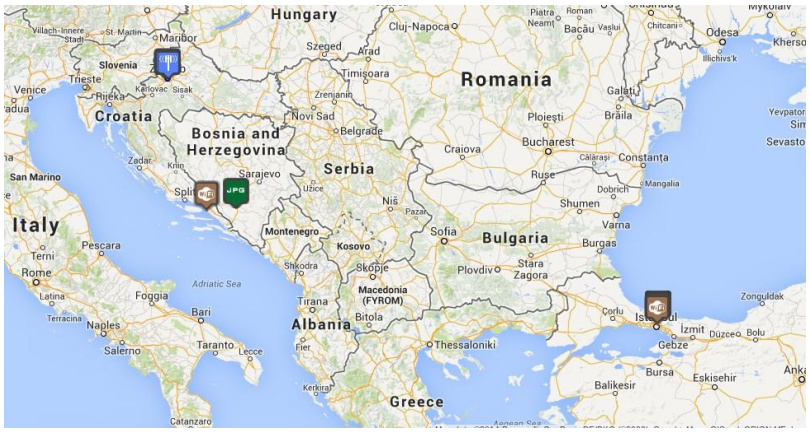
"PhoneConfig: ----> No suitable config found for device"

because there was no configuration for the phone used in this test. It created a subdirectory under its main directory, who's name was composed of the date and time when the examination was performed along with a id of the phone. In the directory were documents and reports from the phone. In total, there were four directories: databases, log, pictures and xml; and files: LocationInformation.log and map.xml. In the directory pictures were all the photos and videos from the camera folder on SD card, but none of the pictures from other locations, such as download folder. Log folder contained adel.log file with recorded information about process of performing ADEL examination. I couldn't find anything useful in directories databases and xml. It could be that it is necessary to have specific configuration for them. LocationInformation file contained latitude, longitude, date and time of some relevant events, for example Wi-Fi connections or where and when some photo was taken. This is how it looks like:

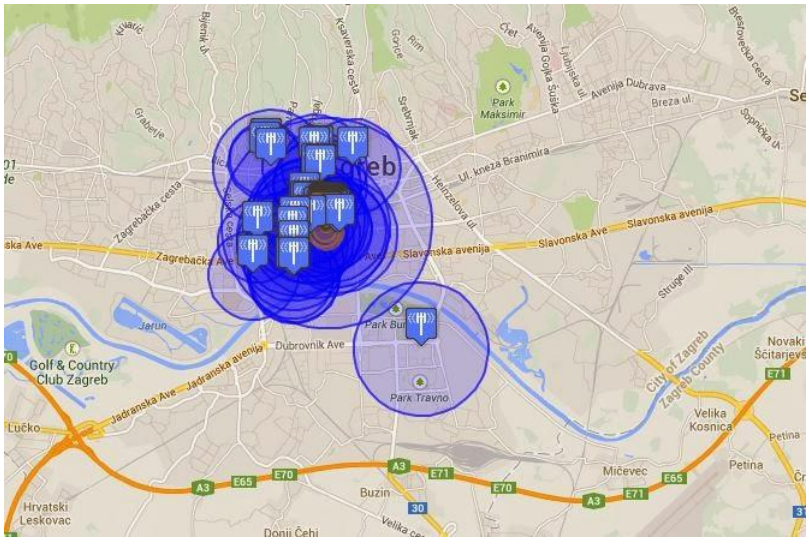
LocationInformation.log

key	accuracy	confidence	latitude	longitude	time
JPEG	500	0	43.3480938889	17.8041266667	01/01/13 14:18:45
219:1:0:38877	-1	0	0.000000	0.000000	03/19/13 09:23:40 +0000
219:1:20150:38342	814	75	45.813823	15.953152	03/19/13 09:28:04 +0000
219:1:0:22587	-1	0	0.000000	0.000000	03/19/13 09:28:05 +0000
219:1:0:22585	-1	0	0.000000	0.000000	03/19/13 09:28:05 +0000
219:1:0:38692	-1	0	0.000000	0.000000	03/19/13 19:25:08 +0000
219:1:0:38691	-1	0	0.000000	0.000000	03/19/13 19:25:08 +0000
219:1:20151:22513	912	75	45.790704	15.960840	03/19/13 19:25:08 +0000
219:1:20151:38667	1165	75	45.803682	15.964580	03/20/13 05:00:30 +0000
219:1:20150:22517	963	75	45.809430	15.968112	03/20/13 05:05:30 +0000
219:1:20150:22874	795	75	45.812899	15.968161	03/20/13 05:10:35 +0000
219:1:20150:22560	1294	75	45.812569	15.955019	03/20/13 05:15:39 +0000
219:1:0:22558	-1	0	0.000000	0.000000	03/20/13 05:20:07 +0000
219:1:20150:22586	808	75	45.814358	15.953439	03/20/13 05:20:07 +0000
219:1:0:38683	-1	0	0.000000	0.000000	03/20/13 06:02:53 +0000
219:1:0:39065	-1	0	0.000000	0.000000	03/20/13 06:02:53 +0000
219:1:0:38317	-1	0	0.000000	0.000000	03/20/13 06:02:53 +0000
219:1:0:38314	-1	0	0.000000	0.000000	03/20/13 06:15:14 +0000
219:1:0:38312	-1	0	0.000000	0.000000	03/20/13 06:15:14 +0000
219:1:0:22554	-1	0	0.000000	0.000000	03/20/13 06:15:14 +0000
219:1:0:22553	-1	0	0.000000	0.000000	03/20/13 06:15:14 +0000
219:1:0:22552	-1	0	0.000000	0.000000	03/20/13 06:15:14 +0000
219:1:20151:22553	1179	75	45.799416	15.973265	03/20/13 07:12:56 +0000
219:1:0:38669	-1	0	0.000000	0.000000	03/20/13 08:59:07 +0000
219:1:0:38530	-1	0	0.000000	0.000000	03/20/13 08:59:07 +0000
219:1:0:38313	-1	0	0.000000	0.000000	03/20/13 08:59:07 +0000
219:1:0:38781	-1	0	0.000000	0.000000	03/20/13 08:59:07 +0000

Map.html opens a map with all that locations marked on it:



and when we zoom in:



I don't know what else this tool can do, but I think that location information is very useful thing in confirming or determining the locations where was a potential criminal and at what time. Since it has this option too, I think that with configuration it could be a very powerful tool for Android forensics.

Santoku

Santoku is Linux distribution with open source tools for mobile forensics, security and analysis. It includes development, penetration testing, device forensics and reverse engineering tools and wireless analyzers. There are several programs for device forensics: AFLogical Open Source Edition, Android Brute Force Encryption, ExifTool, iPhone Backup Analyzer, libimobiledevice, scalpel and Sleuth Kit. I tried data extraction with AFlogical OSE on it.

First you have to download Santoku and install it. I had one problem while I was installing Santoku on virtual machine, I was constantly receiving message:

"This kernel requires an x86-64 CPU, but only detected an i686 CPU. Unable to boot - please use a kernel appropriate for your CPU."

and then it would all freeze. The solution was to enable virtualization technology in BIOS, and then pull out the battery from the laptop for a few seconds. After that I run Santoku again and it was ok, so I could continue with AFlogical.

AFlogical Open Source Edition

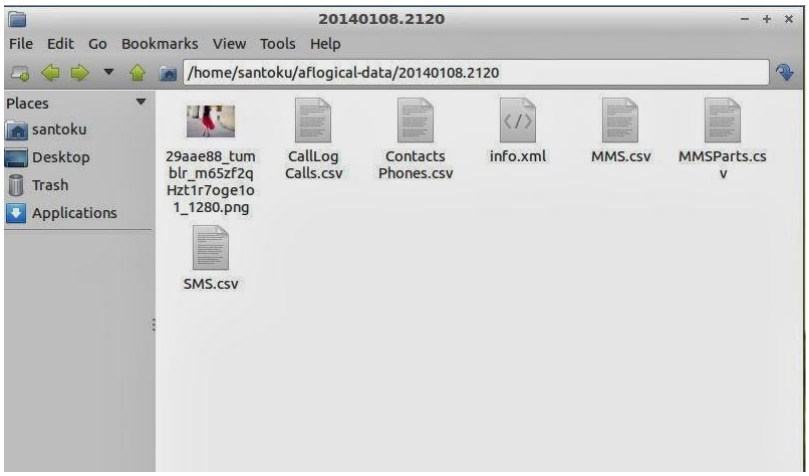
While I was working with AFlogical Open Source Edition (OSE), I followed instructions from tutorial from Santoku howto blog. First you have to start up AFlogical with clicking menu in the lower left corner and then choosing Santoku -> Device Forensics -> AFLogical OSE. Next, you connect your phone over USB with USB debugging mode enabled and check is it connected with command

```
adb devices
```

You also need to install program on your phone with command:

```
aflogical-ose
```

Then you have to check on your phone what you want to extract, click capture button, wait until it's done, and click Enter in AFlogical on computer. It can extract calls, SMS, MMS, MMS parts and contacts. AFlogical creates directory, named with time and date of extraction, in directory aflogical-data, and save extracted files there. That directory, in my case, looked like this:



- This picture, that was extracted, was content of mms on my phone.
- In CallLog Calls were records of my calls with id, phone number, date, duration, type - incoming, outgoing and missed call, new, name and numbertype.
- In Contacts Phones were my contacts and all information about them that were written on my phone. With all the things you normally see in your contact book, like name of the person, phone number, organization... ,there were also information about how many times the number was contacted, is there custom ringtone, time of last contact with number and whether it was sent to the voicemail.
- In info.xml I found very interesting informations about my phone, like IMSI, IMEI-MEID, ICCID, version of Android, sdk version, device name, brand, id, model, and then there were few information about some application from device.
- MMS and MMSParts files had a lot information i didn't understand, but there were the important ones like text of the message, date and number of contact.
- SMS file displayed all my text messages with, among other things, the number of the sender or recipient, date, protocol, status, type, service center number, network and seen status.

When you're done with AFlogical on your phone, you can uninstall it with command:

adb uninstall com.viaforensics.android.aflogical_ose

My opinion on AFlogical is that is a very good tool and it does what is expected of it. It pulled out all the contacts, call logs, MMS and SMS. It is easy to run it and work with AFlogical. The disadvantage is that it doesn't have possibility to extract more things, such as images, videos, data from applications.., but we need to consider that this is still a light version of the product.

Android Pattern Lock Cracker

Android Pattern Lock Cracker is a little tool cracks the pattern lock on Android devices.

First you download source code and unpack it. You will got folder named androidpatternlock-master. This tool requires android sdk installed on computer and rooted

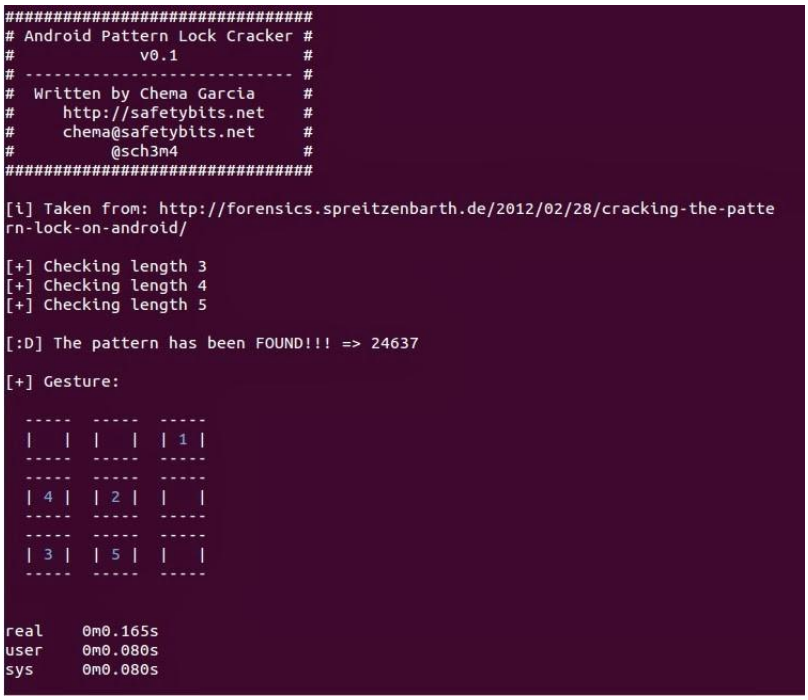
phone with usb debugging mode enabled. Then you have to pull gesture.key file from my phone. You need to connect phone to the computer over usb cable and write these commands in terminal:

```
export PATH=$PATH:<path to sdk>/sdk/platform-tools/adbcd
<path  to androidpatternlock-master>/androidpatternlock-
masteradb pull /data/system/gesture.key
```

You have to change <path to sdk> and <path to androidpatternlock-master> into exact directories you've unpacked the SDK and androidpatternlock in previous command, and then you start the program with:

```
time python crack.pattern.py gesture.key
```

I got this report with display of pattern under Gesture, in terminal:



This tool is simple and it works fine. I don't have any complains.

After I tried these Android forensic tools, I can tell that they all extracted potentially useful informations. I can't compare them because they gave different data. ADEL extracted photos and videos from camera directory on SD card and informations with map about previous locations of examined phone. AFlogical provided me with informations about phone, contacts, calls, SMS and MMS. And in the end there is Android Pattern Lock Cracker which showed me lock pattern of phone. All of them resulted with complete and valid informations.

Dodatak D: Izvještaj

Izvještaj, prema NIST-u [5], bi trebao sadržavati:

- Identitet agencije koja pravi izvještaj
- Identifikacijski broj slučaja
- Istražitelj koji je zadužen za slučaj
- Identitet osobe koja je preuzela uređaj
- Datum preuzimanja uređaja
- Datum izvješća
- Opisna lista stvari dostavljenih na ispitivanje, uključujući serijski broj, marku i model
- Identitet i potpis ispitivača
- Oprema i okruženje korišteno u ispitivanju
- Kratak opis koraka napravljenih tijekom ispitivanja, kao što su traženje stringova,traženje slika, i povrat izbrisanih datoteka
- Dodatni materijali kao što su ispisi određenih dijelova dokaza, digitalne kopije dokaze, i dokumentacija chain of custody
- Detalji rezultata istrage:
    - Specifične datoteke koje se odnose na zahtjeve
    - Druge datoteke kao što su obrisani podaci koji podupiru rezultate
    - Pretraživanje stringova i ključnih riječi
    - Dokazi koji su vezani uz Internet kao što su Web promet, zapisi čavrljanja, privremene datoteke, e-mail...
    - Analiza grafičkih slika
    - Pokazatelji koji upućuju na vlasnika
    - Analiza podataka
    - Opis relevantnih programa i ispitanih predmeta
    - Tehnike koje su korištene da sakriju ili maskiraju podatke kao što su enkripcija, steganografija, sakriveni atributi, sakrivene particije i anomalije imena datoteka
- Zaključak izvještaja

Dodatak E: LiME [25]

From their site: "LiME is a Loadable Kernel Module (LKM), which allows the acquisition of volatile memory from Linux and Linux-based devices, such as those powered by Android."

I tried to follow the instructions for LiME from youtube video, and extract memory from my SE Xperia 8, but I ran on some errors and failed. Here is what I did.

Phone has to be rooted and have debugging mode enabled. LiME also requires Java JDK 6 or 7, Android SDK and arme-eabi tool (I downloaded ARM eCross EABI Toolchain for Linux). Then, the source of the kernel running on the phone we want to examine is necessary. That's because LiME is a kernel module and it has to be compiled for the kernel running on the mobile phone in order to be possible to insert and run it. I lost some time until I found mine here, but that is the site only for Xperia phones. You need to download and unpack it. There should be a kernel directory which will be used later. You have to add arme-eCross-eabi and adb to the path with these commands in terminal, but change <path to arm-eCross-eabi> and <path to SDK> into exact directories you've unpacked the SDK and arm-sCross-eabi:

```
export PATH=$PATH:<path to arm-eCross-eabi>/arm-eCross-eabi/bin

export PATH=$PATH:<path to SDK>/SDK/platform-tools/adb
```

Next, download LiME and unpack it. Then connect your phone to the computer and pull the kernel config from the phone with:

```
adb pull /proc/config.gz
```

If there is no file like that in proc directory on your phone, as was in my case, you can try to find it in source code directory in *kernel/arch/arm/configs*, but with .defconfig extension. You have to rename that directory, no metter where you found it, into .config and move it in kernel directory. Next, like tutorial says, type:

```
make ARCH=arm CROSS_COMPILE=arm-eCross-eabi- modules_prepare
```

I got a few errors reported:
"/home/osaf/Android/arm-eCross-eabi/bin/../libexec/gcc/arm-eCross-eabi/4.5.2/cc1: error while loading shared libraries: libgmp.so.3: cannot open shared object file: No such file or directory".

If you get message like that, you may find solution on this link. It says that you have to install some packages, so just copy command they wrote:
sudo apt-get install libgmp3c2 freeglut3 freeglut3-dev -y

and that problem is solved. Then run make command again. If you get this message:
*"make: *** No rule to make target `modules_prepare'. Stop."*

That is resolved with some modification in make command:

```
make ARCH=arm CROSS_COMPILE=arm-eCross-eabi- modules
```

Another problem that can arise:
*"make:/<path to>/arm-eCross-eabi/bin/arm-eabi-gcc:Command not found...make[1]: *** [kernel/bounds.s] Error 127make:*** [prepareO] Error 2"*

It should continue past that error when "ARCH=arm" is removed from the command line. So, open Makefile in kernel directory and find this line:

```
ARCH                            ?=$(SUBARCH)

CROSS_COMPILE     ?=
```

Where you replace $(SUBARCH) with arm, and write in terminal:

```
make CROSS_COMPILE=arm-eCross-eabi- modules
```

You can also got this message:
*"The present kernel configuration has modules disabled.Type 'make config' and enable loadable module support.Then build a kernel with module support enabled make: *** [modules] Error 1"*

So, the .config file has to be modified as well. Just write:

```
CONFIG_MODULES=y

CONFIG_MODULE_UNLOAD=y
```

in it and cross compile command should make the rest of the needed changes.
When cross compiling is finished, it's time to prepare the module for compilation. You have to position in *lime/src* directory and make new *Makefile.1* with the content from tutorial, but with changed KDIR, placed tabs where they are needed and manually entered address to the arm-eCross-eabi-. There is *Makefile.sample*, where you can see example of makefile with correct tabs. Then write in terminal:

```
make -f Makefile.1
```

You can get this error:
"strip: Unable to recognise the format of the input file `lime.ko"

But, tutorial says that it is ok. Next, you have to move kernel module - lime.ko to your phone:

```
adb push lime.ko /sdcard/lime.ko
```

set up the port:

```
adb forward tcp:4444 tcp:4444
```

open a shell on the Android device:

```
adb shell
```

then install kernel module and redirect output to TCP port:
cd /sdcard

```
insmod lime.ko path=tcp:4444 format=lime
```

and in new terminal on PC, connect to the port on the device with netcat and redirect output to memory.lime:

```
nc 127.0.0.1 4444 > memory.lime
```

After that, you need to get your memory extracted. For me, it didn't work. I just got message:
"Bad adress"

and I didn't know how to resolve that problem.

I can't say anything about this tool, because I couldn't run it and I don't know where is the problem. It returned too many error messages, for which I had to google solutions, and I can't guarantee that they are all good. Although I didn't manage to get positive outcome, I hope it will help you.