CARNET

Reverzni inženjering Android aplikacija

CERT.hr-PUBDOC-2020-1-397

CERT.hr

Sadržaj

Ovaj dokument izradio je Laboratorij za sustave i signale Zavoda za elektroničke sustave i obradbu informacija Fakulteta elektrotehnike i računarstva Sveučilišta u Zagrebu.

1 Uvod

Za razumijevanje teme „Reverzni inženjering Android aplikacija“ potrebno je razjasniti pojmove:

- Android/Android mobilni telefon,
- (Android) aplikacija,
- reverzni inženjering,

te je potrebno pojasniti i zašto bismo uopće imali potrebu za reverznim inženjeringom Android aplikacije.

Android/Android mobilni telefon se u kontekstu ove teme odnosi na „pametni“ telefon s operacijskim sustavom Android. Operacijski sustav Android razvila je tvrtka Google, temelji se na jezgri operacijskog sustava (engl. *kernel*) Linux i, osim na pametnim telefonima, koristi se na tabletima, televizorima (Android TV) i pametnim satovima (Wear OS). Za razliku od iOS-a (operacijski sustav kojega je razvila tvrtka Apple za upotrebu na iPhone i iPad uređajima), Android je slobodan software (engl. *free and open source software*) i može se slobodno koristiti na uređajima drugih tvrtki, ne samo Google-a. To ga je učinilo najbrže rastućim i trenutno dominantnim operacijskim sustavom za mobilne uređaje (1) (2).

U ovom dokumentu se pod pojmom „aplikacija“ podrazumijeva računalni program, tj. softverska aplikacija napravljena za upotrebu na Android telefonu. Android je dominantni mobilni operacijski sustav između ostaloga i zato što podržava veliki broj aplikacija koje se na njemu mogu izvoditi. Aplikacije se mogu preuzeti i instalirati putem aplikacijske trgovine „Google Play“ koja sadrži veliku bazu objavljenih i odobrenih aplikacija za Android. Te aplikacije mogu biti besplatne ili komercijalne, a obično su pisane programskim jezikom Java. Veliki broj dostupnih aplikacija posljedica je laganog procesa odobravanja aplikacija u aplikacijskoj trgovini Google Play – što je dobro jer je korisnicima zbog toga dostupan velik broj aplikacija. No u usporedbi s aplikacijama dostupnim za iPhone i iPad uređaje, Android aplikacije su često niže kvalitete te se češće dogodi da čak sadrže zlonamjerni kôd (3) (4).

Reverzni inženjering je postupak kojim se gotov softver, izvršni kôd, (u ovom slučaju Android aplikacija) upotrebom specijaliziranih alata analizira s ciljem proučavanja načina na koji on funkcionira. Primjerice, korisno je napraviti reverzni inženjering neke sumnjive Android aplikacije koju bismo htjeli instalirati na svoj Android uređaj. Ovdje se pod pojmom sumnjive aplikacije misli na aplikacije koje su potencijalno zlonamjerne, tj. koje možda potajice prikupljaju informacije o korisniku i/ili napadaču daju kontrolu nad korisnikovim uređajem. Naime, jednako kao i u slučajevima zlonamjernog softvera za osobna računala, i aplikacije za Android mogu biti zlonamjerno izrađene ili modificirane kako bi omogućile napadaču da ostvari svoj cilj – obično osiguravanje financijske dobiti ili ciljano prikupljanje informacija o žrtvi.

2 Teorijski pregled reverznog inženjeringa Android aplikacija

Suvremeni pametni telefoni su puno više od običnih mobilnih telefona za telefonske pozive i SMS poruke. To su moćna računala i:

- na njima čuvamo mnoštvo privatnih podataka, primjerice fotografija,
- oni prate i pohranjuju naše kretanje,
- koriste se za bezgotovinska plaćanja, a u nekim slučajevima pohranjuju i podatke o bankovnim karticama,
- ...

Na njima se nalaze osobni podatci koji su zanimljivi mnogim trećim stranama – od marketinških kompanija koje zanima koje su sklonosti i interesi korisnika telefona kako bi mogli što efikasnije prodati proizvod, do kriminalaca koji profitiraju na temelju ilegalno stečenih osjetljivih podataka ili na druge načine, npr. šifriranjem podataka žrtve i zahtijevanjem otkupnine za dešifriranje. Jedan od najefikasnijih načina za prikupljanje nečijih osjetljivih podataka je taj da upravo sam mobilni uređaj odradi cijeli posao – upotrebom zlonamjernih aplikacija.

Reverzni inženjering ima različite primjene, no jedna od glavnih primjena je upravo analiza potencijalno zlonamjernog softvera, tj. konkretnije u ovom slučaju, analiza potencijalno zlonamjernih Android aplikacija. U ovom kontekstu, reverznim inženjeringom će se obično dati odgovor na sljedeća pitanja:

- Je li analizirana aplikacija zlonamjerna (opasna)?
- Ako je, što ona točno radi?

Za jasniji pregled tehnika reverznog inženjeringa, možemo ih okvirno podijeliti po dvije dimenzije:

- osnovne tehnike i napredne tehnike reverznog inženjeringa,
- tehnike statičke i tehnike dinamičke analize.

Osnovne tehnike bi u ovom slučaju bile one koje se većinom provode automatizirano (dakle glavni dio posla provodi neki program/sustav, a ne stručnjak) te za koje nije potrebno duboko tehničko predznanje. Tipičan primjer ovakve osnovne tehnike reverznog inženjeringa bilo bi korištenje sustava za automatsku analizu kao što je VirusTotal ili HybridAnalysis. Takve osnovne tehnike reverznog inženjeringa zlonamjernog softvera detaljnije su opisane u prethodnom dokumentu Nacionalnog CERT-a: „Osnovna analiza zlonamjernog softvera pomoću *online* alata“.

Osnovne tehnike reverznog inženjeringa ne treba podcijeniti samo zato što ih je lagano provesti – i stručnjaci koji dobro vladaju naprednim tehnikama prvo će koristiti osnovne tehnike, jer je njima moguće dobiti puno korisnih informacija u malo vremena. Primjerice, učitavanjem datoteke neke aplikacije na VirusTotal ponekad će odmah biti jasno da je

aplikacija zlonamjerna te čak i što ona točno radi. U takvim slučajevima nije potrebno ni provoditi napredniju analizu.

Napredne tehnike reverznog inženjeringa bi bile one koje „ručno“ provode stručnjaci s dubokim tehničkim razumijevanjem cijelog postupka. Takve su tehnike nužne u slučajevima kada se pojavi neki novi program ili aplikacija, pa sustavi kao VirusTotal ne daju puno korisnih informacija, ili kada je poznato da je program/aplikacija zlonamjerna, ali treba precizno otkriti što ona točno radi.

Po pitanju druge podjele tehnika, statičke i dinamičke analize:

- statička analiza obuhvaća tehnike analize softvera bez njegovog pokretanja – tipični primjer statičke analize bila bi analiza strojnog, odnosno izvornog kôda,

- dinamička analiza obuhvaća tehnike koje se oslanjaju na pokretanje softvera – primjerice, pokretanje aplikacije u kontroliranom okruženju (tzv. „pješčanik“, engl. *sandbox*) i bilježenje što sve ona tada radi (kojim datotekama pristupa, komunicira li mrežno s udaljenim poslužiteljima i sl.).

Ova podjela neovisna je o prvoj – osnovne tehnike reverznog inženjeringa (npr. korištenje servisa kao što je VirusTotal) u pozadini automatski provode i statičku i dinamičku analizu, te isto tako napredne tehnike reverznog inženjeringa obuhvaćaju napredne tehnike statičke i dinamičke analize.

Metoda reverznog inženjeringa, tj. analize zlonamjernog softvera, znatno ovisi o platformi. Primjerice, kod napredne analize izvršnih datoteka za operacijski sustav Microsoft Windows (analiza datoteka s nastavkom *.exe*), često će se prvo provesti dinamička analiza (pokretanje programa u virtualnom računalu uz odgovarajuće alate) zato što je njom moguće dobiti puno korisnih informacija u kratkom vremenu, a tek zatim statička analiza (analiza alatom kao što je IDA Pro) kojom je moguće detaljno istražiti program, no uz značajni vremenski ulog. Takva metoda je posljedica činjenice da je obično izrazito teško dobiti izvorni kôd iz izvršne datoteke (*.exe*), pa je statička analiza vremenski zahtjevna jer stručnjaci moraju analizirati *assembly* kôd.

U slučaju reverznog inženjeringa Android aplikacija, situacija je drugačija – zbog strukture datoteka u kojima se nalaze Android aplikacije, često je moguće prilično lako dobiti izvorni kôd aplikacije u programskom jeziku Java. Zato je u metodi reverznog inženjeringa Android aplikacija statička analiza, konkretnije provjera izvornog kôda, često sljedeći korak nakon osnovne analize servisima kao što su VirusTotal ili HybridAnalysis.

3 Reverzni inženjering Android aplikacije kroz praktični primjer

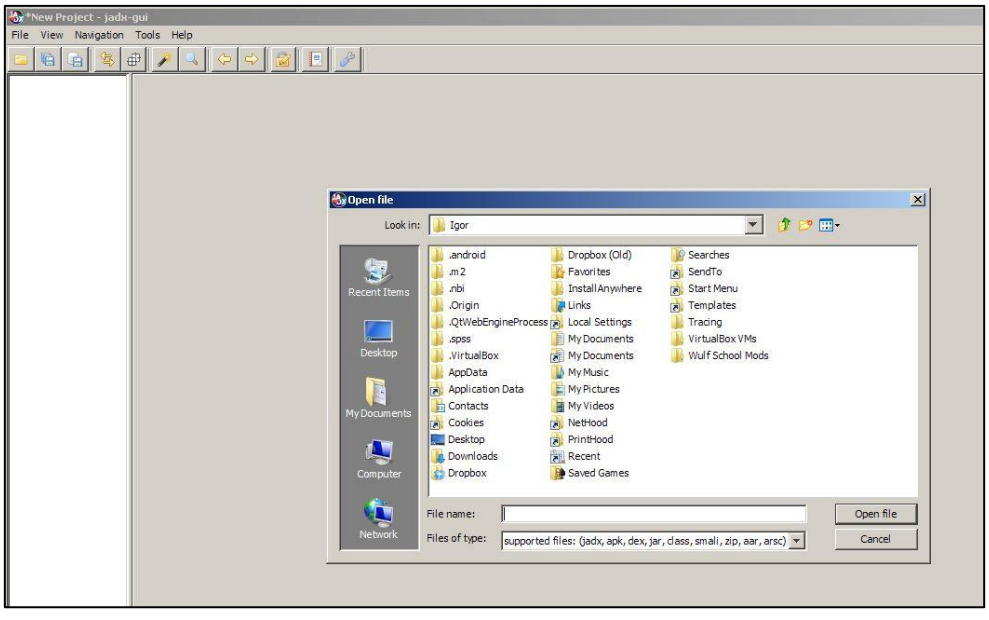
Kako je u dokumentu Nacionalnog CERT-a opisan osnovni reverzni inženjering/analiza pomoću *online* alata, te je taj postupak primjenjiv i za Android aplikacije, fokus ovog dokumenta bit će na uobičajenom sljedećem koraku – ručnoj statičkoj analizi Android aplikacija.

Među najboljim sveobuhvatnim alatima za ručnu statičku analizu Android aplikacija je JADX. JADX je slobodan softver (engl. *free and open source software*) koji iz datoteke Android aplikacije (u APK formatu) može rekonstruirati izvorni kôd, dekodirati druge bitne podatke i metapodatke (npr. podatke u internim datotekama *AndroidManifest.xml* i *resources.arsc*) te pružiti korisniku te informacije kroz pregledno grafičko sučelje. JADX je dostupan za operacijske sustave Windows, macOS i Linux te ga je moguće preuzeti sa službene stranice. Nije potrebno provesti poseban postupak instalacije, već je dovoljno raspakirati preuzetu arhivu.

Ovo će poglavlje demonstrirati ručnu statičku analizu Android aplikacije kroz jednostavan primjer. Kroz poglavlje će se analizirati prilično jednostavna aplikacija koja tvrdi da „pretvara mobitel u svjetiljku“ (paljenjem LED bljeskalice mobitela). Cilj analize je otkriti sadrži li ta aplikacija neke zlonamjerne funkcionalnosti.

Za početak analize potrebno je doći do APK datoteke u kojoj se nalazi Android aplikacija. To je moguće napraviti preuzimanjem APK datoteke s mobitela alatom Android Debug Bridge (adb) ili preuzimanjem datoteke s nekog *online* servisa kao što je APKMirror (5).

Jednom kada je APK datoteka spremna, treba pokrenuti JADX. To je moguće napraviti otvaranjem direktorija *bin* iz preuzete arhive, te pokretanjem odgovarajuće datoteke – „jadx-gui.bat“ na operacijskom sustavu Windows, odnosno „jadx-gui“ na operacijskim sustavima macOS i Linux. Nakon pokretanja, automatski se otvara prozor u kojem je potrebno odabrati APK datoteku koju želimo analizirati, kao što je prikazano na slici 3.1.



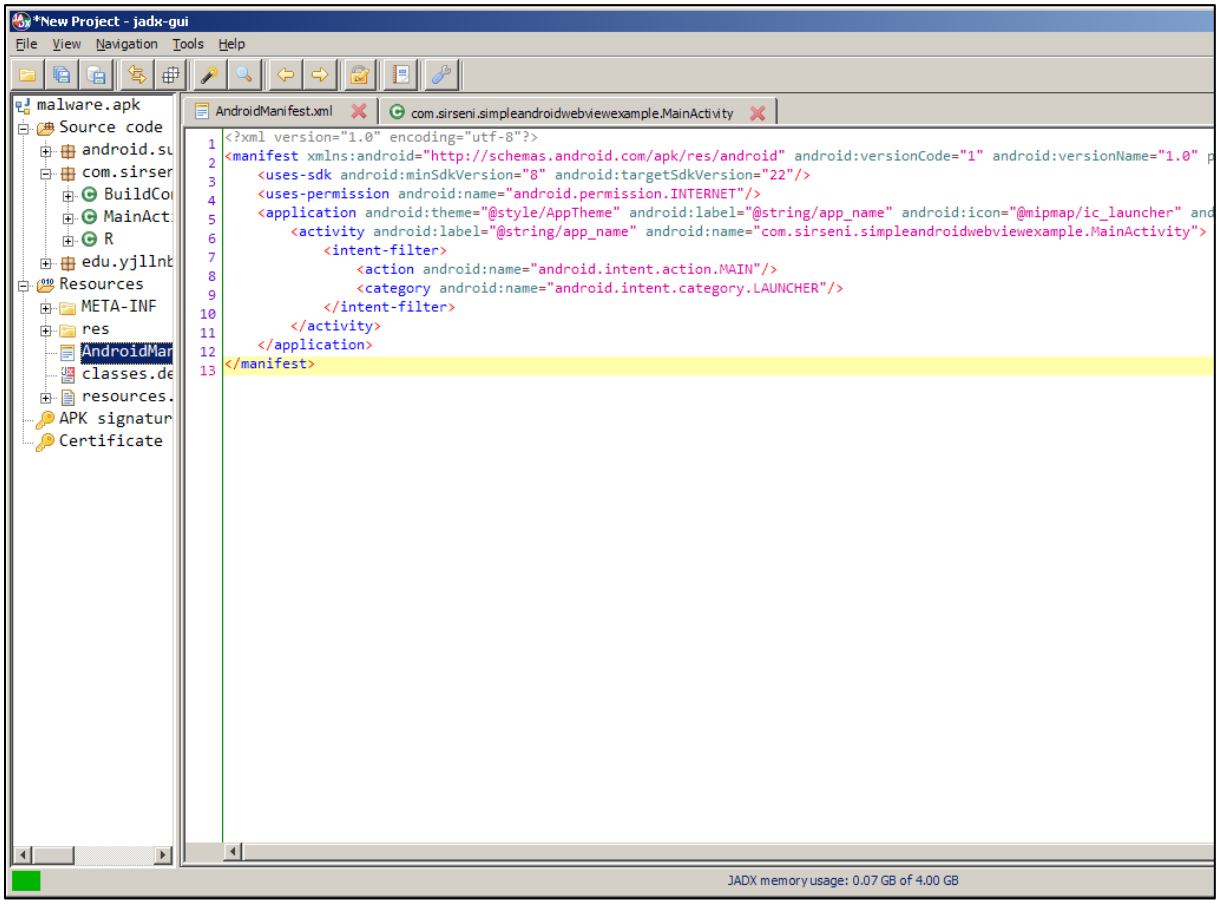
Slika 3.1 početni prozor grafičkog sučelja alata JADX

Nakon odabira željene APK datoteke, JADX analizira datoteku te prikazuje glavni prozor koji je podijeljen u dva dijela. S lijeve se strane nalaze elementi analizirane aplikacije hijerarhiski prikazani u obliku stabla, dok se s desne strane prikazuje izvorni kôd odabranog elementa aplikacije.

Dobar početni korak analize je provjera interne datoteke *Android Manifest.xml*. Internu datoteku *Android Manifest.xml* imaju sve Android aplikacije, a ona sadrži informacije kao što su ime aplikacije, popis komponenata aplikacije (aktivnosti, servisi...) i popis dozvola aplikacije (npr. traži li aplikacija dozvolu za pristup kontaktima korisnika, mikrofonu, kameri...) (6).

Analizu započinjemo provjerom dozvola koje aplikacija zahtjeva za pristup sustavu – korisno je za početak provjeriti jesu li dozvole u skladu s namjenom aplikacije. Aplikaciji koja pretvara mobitel u svjetiljku bi za rad trebala biti potrebna dozvola za pristup kameri, tj. bljeskalici. Ako osim te dozvole traži i dozvole za dodatne dijelove sustava, to je indikacija da bi se moglo raditi o zlonamjernoj aplikaciji.

Slika 3.2 prikazuje pregled datoteke *AndroidManifest.xml* u sučelju alata JADX. Kada se prouči *uses-permission* direktiva (linija 4), vidljivo je da aplikacija zahtjeva pristup internetu, što je sumnjivo za aplikaciju koja se predstavlja kao obična svjetiljka. Još je neobičnije da je to jedina dozvola koju aplikacija zahtjeva, budući da bi ova aplikacija za rad trebala imati pristup bljeskalici.

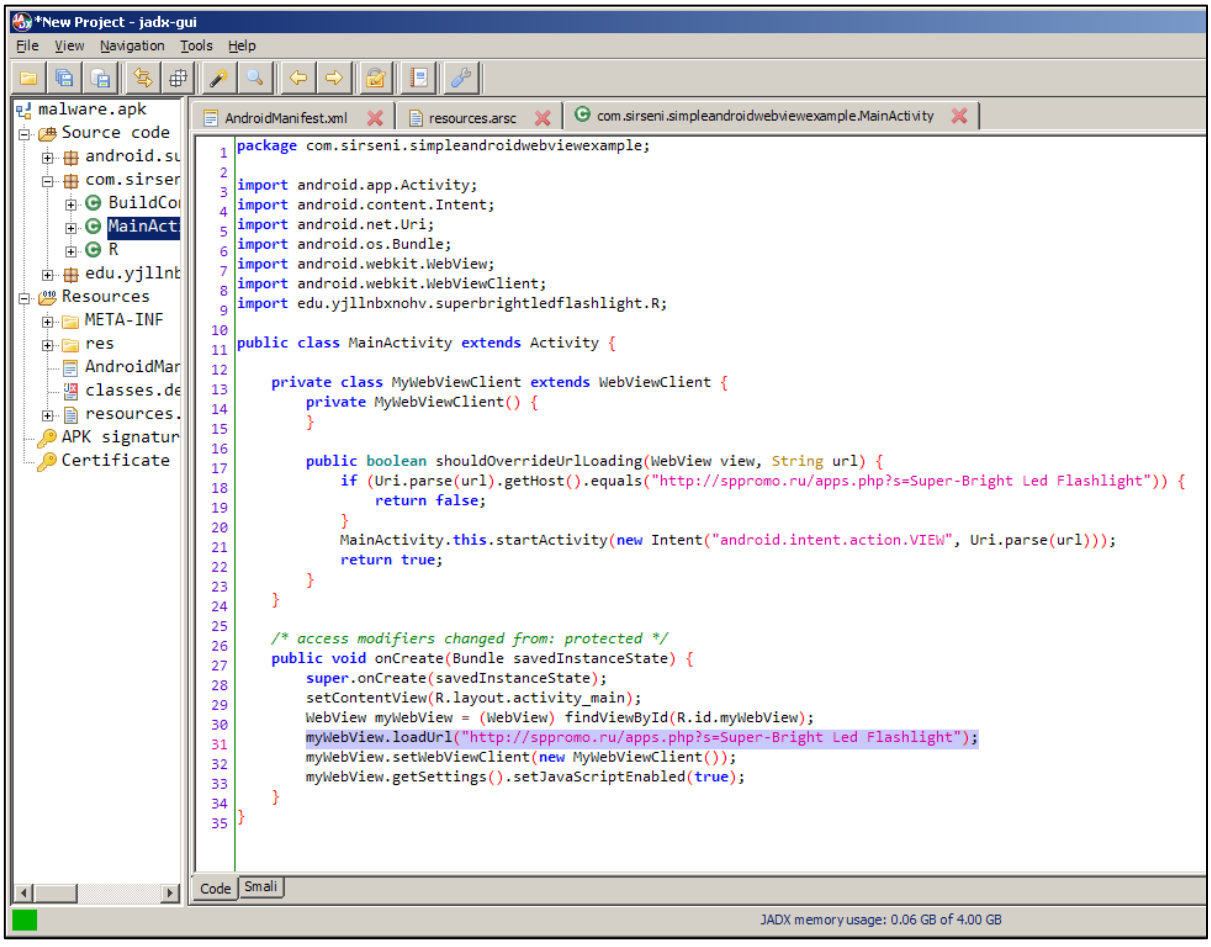


Slika 3.2 Pregled datoteke *AndroidManifest.xml* alatom JADX

Uz to, iako aplikacija navodno služi samo za paljenje bljeskalice mobitela, kada se promotri njena glavna aktivnost (engl. *main activity*) na liniji 6, vidi se da se ona zove „com.sirseni.simpleandroidwebviewexample.MainActivity“. Takav je naziv indikator da je programski kôd aplikacije zapravo samo donekle izmijenjeni primjer korištenja Androidove *WebView* funkcionalnosti (7). Na temelju ovoga je moguće posumnjati da aplikacija zapravo otvara neku web stranicu unutar svog sučelja, umjesto da pali bljeskalicu.

U sljedećem koraku analize prelazimo na pregled izvornog kôda. Izvorni kôd čak i manjih aplikacija je obično prilično opsežan i složen, tako da se rijetko kad analizira sav izvorni kôd. Zato je potrebno procijeniti koji bi dijelovi kôda mogli sadržavati zlonamjernu funkcionalnost i započeti analizu od njih.

U ovom primjeru, na temelju analize interne datoteke *AndroidManifest.xml* moguće je zaključiti kako je glavna aktivnost (čiji naziv spominje *WebView* funkcionalnost za otvaranje web stranice) dobra početna točka za analizu izvornog kôda. U lijevom dijelu sučelja u kojem su hijerarhijski prikazani elementi aplikacije odabiremo Java klasu glavne aktivnosti aplikacije („com.sirseni.simpleandroidwebviewexample“, pa „MainActivity“). Time se u desnom dijelu sučelja otvara njen izvorni kôd. Na slici 3.3 prikazan je izvorni kôd navedene klase unutar sučelja alata JADX.

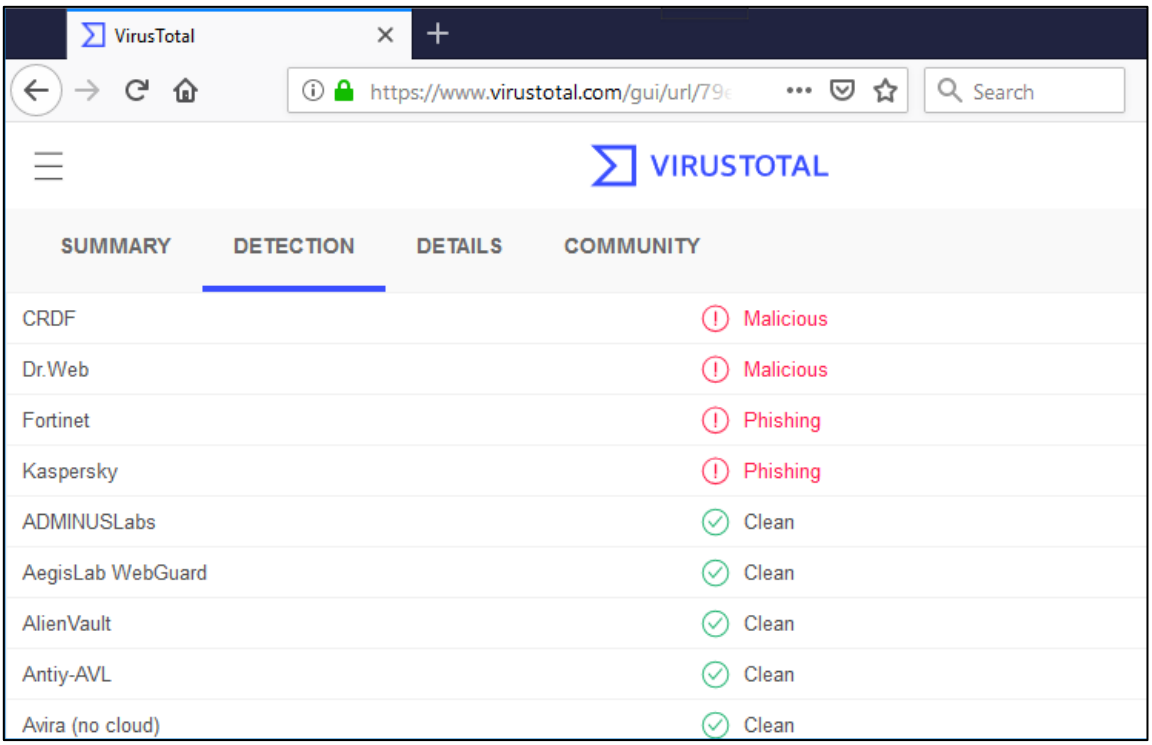


Slika 3.3 Izvorni kôd glavne aktivnosti aplikacije u sučelju alata JADX

Već i bez pretjeranog znanja programiranja Android aplikacija, moguće je naslutiti što prikazani kôd radi – prilikom otvaranja aplikacije, unutar njenog sučelja otvara se web stranica na URL-u: http://sppromo.ru/apps.php?s=Super-Bright Led Flashlight

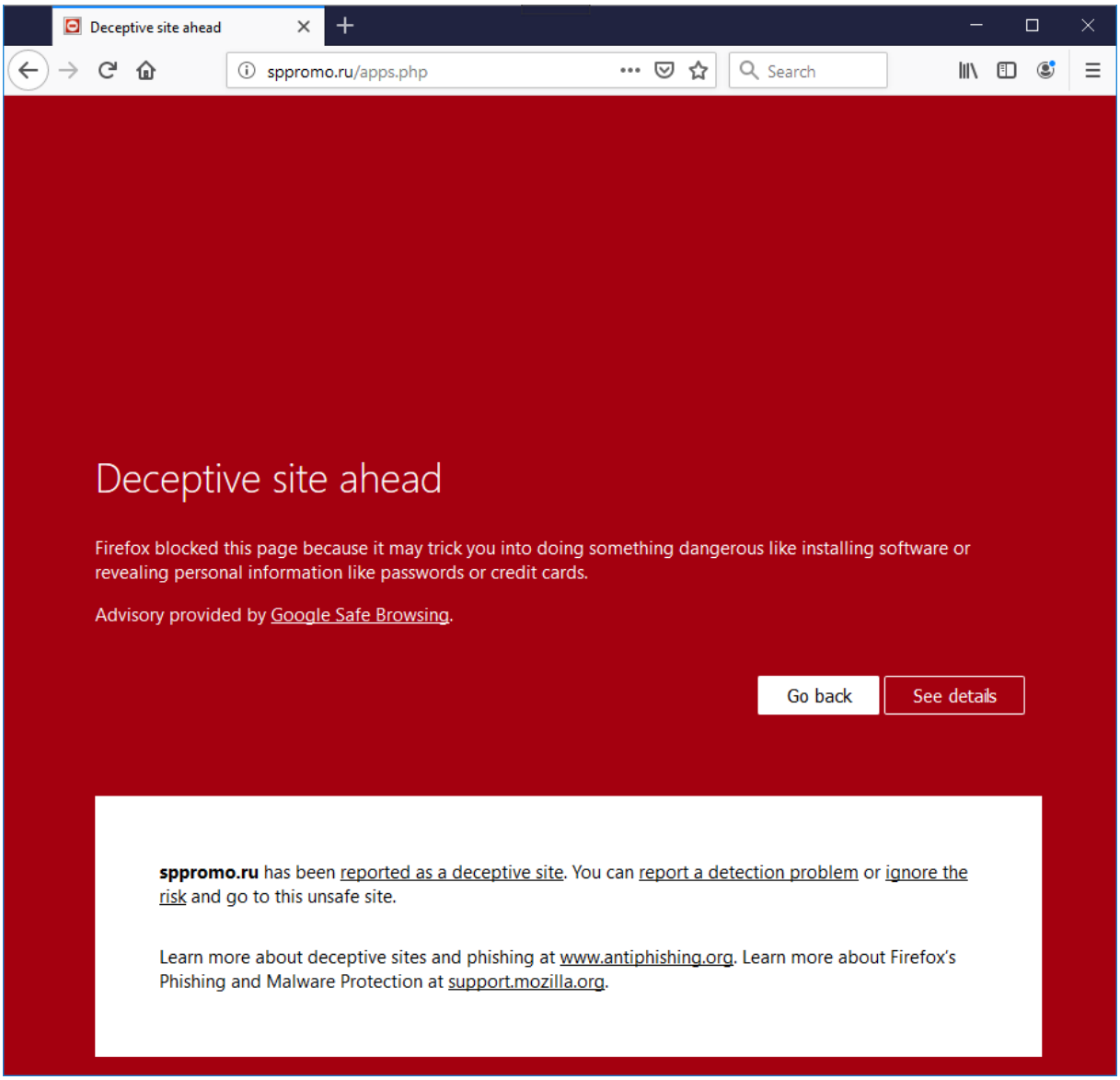
U ovakvim situacijama treba biti oprezan – ovakve URL-ove nikako ne treba otvarati bez dodatnih mjera zaštite jer bi se na njima mogao nalaziti zlonamjerni sadržaj!

U ovakvoj situaciji, za brzu i sigurnu analizu URL-a može biti koristan VirusTotal ili neki sličan servis. Slika 3.4 prikazuje rezultat analize pronađenog sumnjivog URL-a servisom VirusTotal.



Slika 3.4 Rezultat skeniranja sumnjivog URL-a servisom VirusTotal

Za dodatnu provjeru, možemo pokušati pristupiti web stranici web preglednikom, **ali isključivo uz dodatne mjere zaštite, npr. unutar izoliranog virtualnog računala**. Kao što je prikazano na slici 3.5, u ovom slučaju ne uspijevamo u prvom pokušaju ni otvoriti stranicu jer nas web preglednik zaustavlja – ova web stranica je već od ranije prijavljena kao *phishing* web stranica pa se zato nalazi na popisu zlonamjernih web stranica programa *Google Safe Browsing*.



Slika 3.5 Otvaranje sumnjivog URL-a web preglednikom unutar izoliranog virtualnog računala

U ovom trenutku možemo stati s analizom i zaključiti da se ova aplikacija lažno predstavlja kao aplikacija koja pretvara mobitel u svjetiljku (paljenjem LED bljeskalice mobitela) – ona zapravo uopće ne pali bljeskalicu, već nakon otvaranja aplikacije preusmjerava korisnika na opasnu *phishing* web stranicu.

Unatoč tome što je ovaj primjer reverznog inženjeringa Android aplikacije bio kratak, on je prilično reprezentativan. Uobičajeni postupak reverznog inženjeringa Android aplikacije će obično trajati duže od ovog primjera, no sam postupak će biti prilično sličan: analizirat će se interna datoteka *AndroidManifest.xml*, pregledat će se sumnjivi dijelovi izvornog kôda, analizirat će se dodatni unutarnji ili vanjski resursi te će se po potrebi provesti neki oblik naprednije analize (8) (9).

4 Zaključak

Kada naiđemo na potencijalno opasnu Android aplikaciju, postupak reverznog inženjeringa nam može dati odgovor na pitanja:

- Je li ova aplikacija zlonamjerna (opasna)?
- Što ova aplikacija točno radi?

Kod reverznog inženjeringa, prvi je korak obično analiza pomoću *online* alata (npr. VirusTotal, Hybrid-Analysis) kao što je opisano u dokumentu Nacionalnog CERT-a.

Zatim, ako takva analiza nije dala tražene informacije, dobar sljedeći korak u reverznom inženjeringu Android aplikacija je ručna statička analiza aplikacije alatom JADX, kao što je opisano na primjeru u prethodnom poglavlju.

U nekim slučajevima, ni takvom analizom neće biti moguće doći do svih željenih odgovora. Tada, ovisno o konkretnoj situaciji, postoje brojni sljedeći koraci u analizi:

- ako je aplikacija zaštićena od analize (engl. *packing*, *obfuscation*), potrebno je zaobići takve zaštite prije analize funkcionalnosti same aplikacije,
- ako aplikacija sadrži i nativni (strojni) kôd, primjerice u vanjskim bibliotekama (engl. *libraries*), onda je taj kôd potrebno analizirati odgovarajućim alatima (npr. IDA, Ghidra, radare2),
- ako je statička analiza aplikacije previše složena, moguće je aplikaciju (po potrebi) izmijeniti uz pomoć alata kao što je *apktool* te pokrenuti unutar virtualne okoline (emulatora) ili na stvarnom Android telefonu (namijenjenom samo za analizu zlonamjernih aplikacija),
- ako značajni dio funkcionalnosti aplikacije ovisi o vanjskom poslužitelju (npr. aplikacija u pozadini komunicira s web stranicom), onda može biti korisno analizirati mrežni promet ili čak (ako je to dopušteno) kroz interakciju s vanjskim poslužiteljem saznati više o funkcionalnosti sustava.

Za one koji žele naučiti više o naprednim zlonamjernim Android aplikacijama, kako one funkcioniraju te kako ih analizirati, korisno je pratiti članke o analizi takvih aplikacija koje objavljuju sigurnosne tvrtke i stručnjaci, primjerice:

- SpyDealer: Android Trojan Spying on More Than 40 Apps
- The Rotexy mobile Trojan – banker and ransomware
- Mobile Malware Analysis : Tricks used in Anubis
- Skygofree: Following in the footsteps of HackingTeam

5 Literatura

1. **Karch, Marziah.** What Is Google Android? *Lifewire.* [Mrežno] 21. svibnja 2019. [Citirano: 7. kolovoza 2019.] https://www.lifewire.com/what-is-google-android-1616887.

2. **StatCounter Global Stats.** Mobile Operating System Market Share Worldwide. [Mrežno] [Citirano: 7. kolovoza 2019.] http://gs.statcounter.com/os-market-share/mobile/worldwide.

3. **Clark, Bryan.** Why Are iOS Apps Still Better Than Android Apps? *MakeUseOf.* [Mrežno] 15. veljače 2016. [Citirano: 7. kolovoza 2019.] https://www.makeuseof.com/tag/ios-apps-still-better-android-apps/.

4. **Costello, Sam.** 5 Reasons iPhone Is More Secure Than Android. *Lifewire.* [Mrežno] 24. lipnja 2019. [Citirano: 7. kolovoza 2019.] https://www.lifewire.com/reasons-iphone-is-more-secure-than-android-2000308.

5. **Margaritelli, Simone.** Android Applications Reversing 101. [Mrežno] 27. travnja 2017. [Citirano: 7. kolovoza 2019.] https://www.evilsocket.net/2017/04/27/Android-Applications-Reversing-101/.

6. **Android Developers.** App Manifest Overview. [Mrežno] [Citirano: 7. kolovoza 2019.] https://developer.android.com/guide/topics/manifest/manifest-intro.

7. —. WebView. [Mrežno] [Citirano: 7. kolovoza 2019.] https://developer.android.com/reference/android/webkit/WebView.

8. **Alderson, Elliot.** MFSocket: A Chinese surveillance tool. [Mrežno] 26. lipnja 2019. [Citirano: 7. kolovoza 2019.] https://medium.com/@fs0c131y/mfsocket-a-chinese-surveillance-tool-58e8850c3de4.

9. —. Reverse Engineering of the Anubis Malware — Part 1. [Mrežno] 29. listopada 2018. [Citirano: 7. kolovoza 2019.] https://medium.com/@fs0c131y/reverse-engineering-of-the-anubis-malware-part-1-741e12f5a6bd.

10. **Fox-Brewster, Tom.** Check the permissions: Android flashlight apps criticised over privacy. *The Guardian.* [Mrežno] 3. listopada 2014. [Citirano: 7. kolovoza 2019.] https://www.theguardian.com/technology/2014/oct/03/android-flashlight-apps-permissions-privacy.