Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.4:621.395
DOI: https://doi.org/10.24867/11BE05Matkovski

PRIMENA PRINCIPA BEZBEDNOG RAZVOJA ANDROID APLIKACIJA

THE USE OF PRINCIPLES OF SECURE DEVELOPMENT OF ANDROID APPLICATIONS

Marijana Matkovski, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu su predstavljeni osnovni principi bezbednog razvoja Android aplikacija. Date su teorijske osnove potrebne za razumevanje osnovnih bezbednosnih principa i predstavljene su najčešće ranjivosti mobilnih aplikacija zajedno sa preporučenim merama kako bi bila onemogućena ili maksimalno otežana njihova realizacija. Android platforma sama po sebi poseduje ugrađene bezbednosne mehanizme kako bi doprinela bezbednosti aplikacija, ali to veoma često nije dovoljno. Da bi korisnici i njihovi podaci bili sigurni, neophodno je da programeri vode računa o bezbednosti aplikacija prilikom svake faze njihovog razvoja. U cilju demonstracije načina na koji je moguće implementirati bezbednosne mehanizme, razvijena je Android aplikacija namenjena za rešavanje onlajn testova. U aplikaciji su implementirane preporuke date u OWASP top ten mobile listi. Bezbedan prenos podataka između aplikacije i servera koji aplikacija koristi zauzima posebno mesto u radu. Aplikacija je razvijena pomoću Java programskog jezika, dok je za raspored komponenti na ekranu korišćen XML.*

Ključne reči: *Android, bezbednost, ranjivost, aplikacija*

Abstract – *This paper analyses the security problems of the Android mobile applications. The theoretical foundations required to understand basic security principles are given and the most common vulnerabilities of mobile applications together with the recommendations on how to prevent or aggravate their realisation are presented. Even though the Android platform possesses built-in security mechanisms to improve security of applications, it often does not suffice. In order for users and their data to be secure, it is necessary for developers to take care of security of applications at every stage of their development. With the aim of demonstrating security mechanisms, an Android application for online testing has been developed. The recommendations given in the OWASP top ten mobile list have been implemented in the application. A secure data transfer between the application and the server used by the application occupies a special place in the paper. The application for online testing has been implemented using the Java programming language, and XML for the layout of the screen components.*

Keywords: *Android, security, vulnerability, application*

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Aleksandar Kupusinac, vanr. prof.

1. UVOD

Ubrzan razvoj tehnologije u proteklih par decenija doveo je do velikog napretka u razvoju mobilnih uređaja, sa posebnim akcentom na mobilne telefone. Širok spektar upotrebe i velike količine bitnih podataka koji su skladišteni ili se prenose kroz ove uređaje samo su neki od razloga zbog kojih su upravo oni postali meta za hakerske napade. Iako Android, kao trenutno najkorišćeniji operativni sistem za mobilne telefone [1] u svojoj osnovi poseduje ugrađene bezbednosne mehanizme, velika odgovornost za bezbednost je i dalje na programerima mobilnih aplikacija.

U radu su identifikovani najčešći bezbednosni rizici koji se javljaju prilikom upotrebe Android aplikacija. Predstavljene su aktuelne bezbednosne prakse za razvoj aplikacija, koje se izvršavaju na Android platformi. Kao primer aplikacije u kojoj su ispoštovane bezbednosne mere implementirana je TestIT Android aplikacija, namenjena za online testove, kako bi se omogućilo online ocenjivanje.

2. BEZBEDNOST ANDROID APLIKACIJA

Bezbednost Android aplikacija se pre svega odnosi na zaštitu osetljivih podataka koje korisnik čuva na mobilnom uređaju, ili razmenjuje preko njega. Digitalne pretnje dolaze sa svih strana, u različitim oblicima, a najlakša meta za sprovođenje pretnji u napad su mobilni telefoni sa Android operativnim sistemom.

2.1. Provera ranjivosti korišćenih biblioteka

Prilikom razvoja softvera, koriste se različite softverske biblioteke kako bi se ubrzala implementacija željenih funkcionalnosti. Upotreba programskih biblioteka može da prouzrokuje ranjivosti razvijenog softvera. Prilikom razvoja aplikacija, pre donošenja odluke o upotrebe neke biblioteke, preporuka je prvo proveriti njeno poreklo. U slučaju da je biblioteka pronađena na nekom sigurnom mestu, potrebno je proveriti da li su za nju poznate neke ranjivosti. Najbolje mesto za proveru postojećih ranjivosti je *CVE* rečnik [2]. Jedan od najpouzdanijih načina za proveru ranjivosti korišćenih biblioteka je upotreba softverskih alata koji na osnovu spiska korišćenih biblioteka generišu izveštaje sa pronađenim ranjivostima, na osnovu *CVE* rečnika. Jedan od najpopularniji alata za automatizovanu proveru ranjivosti je *OWASP Dependency-Check* [3].

2.2. HTTPS

HTTP je mrežni protokol koji je dugo predstavljao osnovni metod za prenos informacija između klijenata i

servera. Mana koja se javlja prilikom korišćenja ovog protokola je mogućnost krađe informacija koje se razmenjuju. Da bi se rešili ovi problemi, razvijen je HTTPS. HTTPS predstavlja bezbednu verziju HTTP protokola, gde S predstavlja skraćenicu od reči sigurnost (eng. *secure*). HTTPS koristi SSL/TLS protokol za kriptovanje podatka koji se razmenjuju između klijenta i servera i na taj način pruža zaštitu od napada presretanjem. Pored šifrovanja podataka, HTTPS omogućava autentifikaciju, odnosno pruža garanciju da je server ili internet stranica, upravo ona kojom se predstavlja.

2.3. Ugrađeni bezbednosni mehanizmi

Android u sebi poseduje mehanizme koji doprinose bezbednosti celokupnog sistema, pa samim tim i bezbednosti aplikacija koje su instalirane na uređajima sa Android operativnim sistemom. Najvažniji ugrađeni bezbednosni mehanizmi su: *sendboxing* - aplikacije su međusobno izolovane i ne mogu direktno da komuniciraju, enkripcija fajl sistema - zaštita od čitanja podataka u slučaju da telefon bude izgubljen ili ukraden, korisničke permisije - korisnikčke dozvole za pristup funkcijama sistema i korisničkim podacima koji se nalaze na uređaju, aplikacijske permisije - dozvole definisane aplikacijom za kontrolu pristupa podacima te aplikacije; dobro istestirana implementacija zajedničkih bezbednosnih funkcionalnosti koje koristi više aplikacija na Android platformi.

2.4. OWASP top ten mobile

OWASP top ten mobile [4] predstavlja jedan od najznačajnijih projekata OWASP zajednice (eng. *Open Web Application Security Project*). *OWASP* top ten mobile je lista koja identifikuje deset najčešćih tipova ranjivosti sa kojima se suočavaju mobilne aplikacije širom sveta. Poslednji put je ažurirana 2016. godine, sa ciljem da pruži smernice za programeranje mobilnih aplikacija i predstavi najbolje prakse kodiranja.

2.4.1. Nepravilna upotreba platforme

Nepravilna upotreba platforme se javlja kada aplikacija ne koristi pravilno funkcije operativnog sistema ili bezbednosne mehanizame koje pruža platforma. Komponente na Android platformi koje su najčešće zloupotrebljene su namere (eng. *intent*) i permisije za pristup fajlovima. Da bi ova vrsta ranjivosti bila izbegnuta, neophodno je da programeri budu upoznati sa najboljim praksama u kodiranju mobilnih aplikacija. Zbog čestih promena i izdavanja novih verzija Androida, najsigurnije mesto za pronalaženje najboljih praksi prilikom kodiranja je zvanična Android dokumentacija.

2.4.2. Nesigurno skladištenje podataka

Nesigurno skladištenje podataka je ranjivost koja se može ispoljiti u slučaju gubitka ili krađe mobilnog uređaja i u slučaju postojanja zlonamerne aplikacije na istom uređaju gde se nalazi i pogođena aplikacija. Ako napadač može da obezbedi fizički pristup mobilnom uređaju, tada pomoću odgovarajućeg softvera može da ostvari uvid u sve direktorijume aplikacije, koji često sadrže osetljive podatke. Napadač tada može da izmeni aplikaciju i time da osigura da će aplikacija njemu slati osetljive podatke koje bi inače slala na server. Da bi se izbegla ova ranjivost, programeri moraju da naprave dobar izbor, koje podatke je neophodno čuvati na mobilnom uređaju, a koje je dovoljno čuvati samo na serveru.

2.4.3. Nesigurna komunikacija

Prilikom razvoja mobilnih aplikacija neophodno je pretpostaviti da će razvijana aplikacija morati da koristi neki od protokola za komunikaciju sa serverom, kako bi slala i preuzimala podatke. Preporuka je da sva komunikacija bude šifrovana. Potrebno je osigurati da su TLS sertifikati validni, izdati od pouzdanog sertifikacionog tela, kao i da je verzija TLS-a ažurirana na verziju 1.2 ili 1.3.

2.4.4. Nesigurna autentifikacija

Autentifikacija predstavlja proces provere identiteta korisnika, odnosno proveru da li je korisnik baš onaj kojim se predstavlja. Kada napadač uvidi da šema za potvrdu identiteta koju koristi aplikacija poseduje neke ranjivosti, on zaobilazi bilo kakvu interakciju sa aplikacijom i direktno šalje zahteve ka serveru. Kako bi se sprečila nesigurna autentifikacija, preporučuje se da se provera identiteta sprovodi na serverskoj strani. Preporučuje se da aplikacije ne podržavaju trajnu potvrdu identiteta (opciju „zapamti me“), kao i da se izbegava autentifikacija pomoću četvorocifrenog *PIN* koda.

2.4.5. Nedovoljna kriptografija

Nedovoljna kriptografija predstavlja ranjivost koja nastaje zbog upotrebe slabih kriptografskih mehanizama. Ranjivost se zasniva na mogućnosti vraćanja šifrata u početni oblik, odnosno u otvoreni tekst. Može je sprovesti u napad svako ko ima mogućnoest pristupa nepravilno šifrovanim podacima. Kako bi mobilna aplikacija izbegla ranjivost nedovoljne kriptografije preporučuje se izbegavanje čuvanja osetljivih podataka na mobilnom uređaju. Preporučljivo je koristiti jake kriptografske mehanizme, za koje se procenjuje da će biti bezbedni barem još narednih deset godina.

2.4.6. Nesigurna autorizacija

Autorizacija podrazumeva postupak utvrđivanja čemu određeni korisnik sme da pristupi. Mobilne aplikacije su najčešće napravljene za korišćenje od strane jednog tipa korisnika. Veoma često više različitih aplikacija pristupa istom serveru; zbog čega je autorizacija neophodna. Nesigurna autorizacija predstavlja ranjivost koja se realizuje u napad tako što autentifikovani napadač, metodom nasilnog isprobavanja pokušava da pronađe ranjive pristupne tačke (eng. e*ndpoints*), kako bi izvršio neke od funkcionalnosti za koje nema ovlašćenje. Da bi se izbegla nesigurna autorizacija, preporuka je da se prilikom autorizacija oslanja samo na informacije sa servera i da se izbegava upotreba bilo kakvih informacija koje dolaze sa mobilnog uređaja. Prilikom svakog zahteva koji pristigne na serversku stranu, neophodno je proveriti da li korisnik koji je poslao zahtev ima pravo da pristupi endpointu na koji je zahtev poslat.

2.4.7. Nizak kvalitet klijentskog koda

Ranjivost niskog kvaliteta klijentskog koda veoma često proizilazi iz loše prakse kodiranja, kada različiti članovi

razvojnih timova kodiraju na različite načine, ali i zbog dokumentovanja programskog koda na nedovoljno jasan način. Kvalitet klijentskog koda podrazumeva i proveru ranjivosti korišćenih biblioteka. Za prevazilaženje ranjivosti niskog kvaliteta klijentskog koda preporučuje se poštovanje praksi oko kojih se slažu svi članovi razvojnog tima. Sav kod mora biti čitak, uredan i dokumentovan na standardizovan način. Neophodno je redovno raditi proveru ranjivosti korišćenih programskih biblioteka.

2.4.8. Izmenjeni kod

Napadači često modifikuju izvorne kodove poznatih aplikacija i postavlaju tako modifikovane aplikacije u nezvanične prodavnice aplikacija. Korisnici odatle preuzimaju aplikacije i instaliraju ih na svoje mobilne uređaje.

Napadači obično vrše izmene u kodu aplikacija sa ciljem da omoguće krađu korisničkih podataka poput kredencijala za logovanje, brojeva kreditnih kartica i slično. Ranjivost izmenjenog koda direktno je povezana sa ranjivošću reverznog inženjeringa. Da bi bilo omogućeno menjanje izvornog koda, on prvo mora biti dostupan.

2.4.9. Reverzni inženjering

Napadač može da preuzme aplikaciju iz prodavnice aplikacija i da dalje vrši njenu analaizu u svom lokalnom okruženju, koristeći različite alate. Analizom bajt koda aplikacije korišćenjem različitih alata omogućeno je razumevanje logike različitih funkcionalnosti, a u nekim slučajevima i potpuno otkrivanje izvornog koda. Da bi se sprečio reverzni inženjering preporučuje se upotreba alata za zamućivanje koda aplikacije. Preporuka je koristiti C/C++ programske jezike za pisanje osetljivih delova programskog koda.

2.4.10. Eksterne funkcionalnosti

Prilikom razvoja aplikacija, programeri često čuvaju delove koda koji olakšava pristup serveru i pojednostavljuje testiranje aplikacije. Taj kod je potreban samo u toku razvoja aplikacija, ne i u produkciji. U slučaju da taj kod dospe u produkciju, posledice mogu biti razne. Ova ranjivost se obično realizuje u napad tako što napadač preuzme aplikaciju u svoje lokalno okruženje i tu sprovodi dalja ispitivanja; kako bi otkrio postojanje osetljivih podataka. Prilikom analize aplikacije napadač prvenstveno analizira log poruke, konfiguracione datoteke i binarni kod aplikacije. Za sprečavanje ove ranjivosti, potrebno je obratiti pažnju da u produkcionoj verziji aplikacije ne sme postojati kod za testiranje kao i da u podešavanjima ne smeju postojati skrivene opcije.

3. APLIKACIJA ZA ONLINE TESTOVE

TestIT predstavlja Android aplikaciju namenjenu za rešavanje online testova. Glavna ideja je da implementirano softversko rešenje bude otporno na neke od najpoznatijih i najučestalijih napada.

3.1. Implementacija sigurne komunikacije

Za uspostavljanje sigurne komunikacije između TestIT aplikacije i servera, preko nebezbedne mreže korišćen je HTTPS protokol. Upotrebom HTTPS-a otežava se sprovođenje napada pomoću čoveka u sredini.

3.1.1. Zabrana nešifrovane komunikacije

Android 6.0 sa sobom je doneo novinu u vidu atributa usesCleartextTraffic koji služi kao zaštita od korišćenja nešifrovane komunikacije. Sve do verzije Androida 9.0 ovaj atribut je bio podrazumevano postavljen na vrednost *true*, čime je bio dozvoljen prenos nešifrovanog sadržaja. Od verzije Androida 9.0 to je promenjeno. U TestIT aplikaciji atribut *usesCleartextTraffic* je postavljen na vrednost *false*, kako bi se osiguralo da sva komunikacija mora biti šifrovana, nevezano za to na kojoj verziji Androida je pokrenuta aplikacija.

3.1.2. Zabrana starijih verzija TLS-a

Trenutno aktuelne verzije TLS-a su TLSv1.2 i TLSv1.3. Postoji tendencija da se u narednom periodu potpuno pređe na korišćenje verzije TLSv1.3. Android podrazumevano dozvoljava upotrebu svih verzija TLS-a, ali uvek prilikom uspostavljanja komunikacije (procesa rukovanja) prvo predlaže najnovije verzije. Kako su ranije verzije TLS protokola podložne velikom broju napada, da bi se aplikacija bolje zaštitila, potrebno je zabraniti upotrebu starijih verzija TLS protokola. TestIT dozvoljava isključivo verzije TLSv1.2 i TLSv1.3.

3.2. Autentifikacija

Za autentifikaciju u TestIT aplikaciji koristi se korisničko ime i lozinka. Uneto korisničko ime i lozinka se šalju na server gde se vrši autentifikacija. Na putu do servera podaci su šifrovani, jer je omogućena isključivo HTTPS komunikacija. Kao rezultat uspešne autentifikacije server vraća JWT *(JSON web token)*. Nakon dobijanja JWT, TestIT aplikacija ga čuvati kod sebe, kako bi mogla da ga šalje zajedno sa svakim narednim zahtevom. JWT se čuva u *Encrypted Shared Preferences*-u, gde je moguće čuvati podatke u obliku prova ključ - vrednost. Za enkripciju je iskorišćen *master key*, koji se u čuva u *Android keystore system*-u [5]. Na taj način se omogućava jednostavan pristup tokenu, a u isto vreme je postignuta i zaštita u slučaju neovlašćenog pristupa.

JWT koji ima neograničeno trajanje predstavlja pretnju u sistemu, jer može biti ukraden. Isticanje tokena predstavlja jedno od rešenja za potencijalni problem krađe tokena. Nakon isteka token više nije validan i korisnik ne može da pristupi serverskim metodama kojima mogu da pristupe samo autentifikovani korisnici, sve dok token ne bude obnovljen ili dok se korisnik ponovo ne uloguje. Praksa je da se koriste dva tokena prilikom komunikacije sa serverom. Kada se korisnik uspešno autentifikuje dobija token za pristup resursima i token za obnovu tokena za pristup resursima. U slučaju da se desi da na zahtev koji je poslat serveru, kao odgovor bude vraćen HTTP status kod 401, automatski će biti pokušana obnova tokena. Obnova tokena se vrši tako što se server šalje token za obnovu tokena na osnovu kojeg server generiše novi par tokena.

3.3. Smanjene veličine aplikacije i zamućivanje koda

Android uređaji u većini slučajeva imaju ograničen memorijski kapacitet i iz tog razloga je neophodno obratiti pažnju na veličinu aplikacija koje se razvijaju. *Android Gradle plugin* od verzije 3.4.0 koristi *R8* kompajler koji pruža mogućnost automatizacije

uklanjanja nekorišćenih delova koda. *R8* kompajler ima mogućnost da pronađe i nakon toga ukloni nekorišćene klase, metode, polja i atribute iz aplikacije ali i iz biblioteka koje su povezane sa aplikacijom. Smanjenju veličine aplikacije doprinosi i uklanjanje nekorišćenih resursa iz aplikacije i povezanih programskih biblioteka [6]. Uklanjanje nekorišćenih delova programskog koda i nekorišćenih resursa doprinosi smanjenju bezbednosnih rizika.

Zamućivanje koda podrazumeva zamenu stvarnih naziva klasa i njenih članova sa kraćim nazivima. Time se otežava reverzni inženjering, odnosno vraćanje prevedenog koda u originalni, što predstavlja jednu od veoma čestih pretnji za Android aplikacije [6].

Android studio u kojem je razvijana TestIT aplikacija koristi najnoviju verziju *Android Gradle plugin*-a (verziju 4.0.1) tako da je smanjenje veličine aplikacije, optimizaciju i zamućivanje moguće izvršiti prilikom prevođenja; upotrebom *R8* kompajlera.
Smanjivanje veličine koda i zamućivanje nije podrazumevano uključeno prilikom kreiranja novog projekta u Android studiju, već je potrebno dodati podešavanja u build.gradle datoteci.

3.4. Provera ranjivosti programskih biblioteka

Za proveru ranjivosti korišćenih biblioteka upotrebljen je *OWASP Dependency-Check*. Automatizovana provera ranjivosti programskih biblioteka koje su uključene u projekat sprovođena je u toku razvoja TestIT aplikacije prilikom svakog dodavanja novih programskih biblioteka. Takođe, provera ranjivosti vršena je više puta u toku razvoja, radi provere da li je u međuvremenu registrovana nova ranjivost korišćenih biblioteka. Dobijeni izveštaji otkrivenih ranjivosti poseduje lažne pozitivne slučajeve, odnosno otkrivene su ranjivosti koje ne mogu da utiču na aplikaciju. Iz tog razloga je izvršena provera *CVE* dokumentacije svake otkrivene ranjivosti i donet je zaključak da one ne utiče na razvijanu aplikaciju.

3.5. Curenje podataka

Snimanje i slikanje ekrana predstavlja jedan od veoma čestih načina za curenja podataka. Posebno su ugroženi prikazi u kojima se unose kredencijali za prijavljivanje, kao i bilo kakav vid lozinki. Zaštitu od ovakvog vida curenja podataka moguće je obezbediti postavljanjem sigurnosne zastavice ekrana *FLAG_SECURE*.

U slučaju da korisnik pokuša da iskoristi dodatnu aplikaciju za snimanje ekrana, na snimku će ekran TestIT aplikacije biti potpuno crn i time je onemogućeno curenje podataka snimanjem. Slikanje ekrana je takođe onemogućeno, a kao rezultat pokušaja slikanja korisnik će dobiti poruku da je slikanje zabranjeno od strane aplikacije ili organizacije.

3.6. Validacija korisničkog unosa

Provera korisničkog unosa pre slanja podataka na dalju obradu na serveru, predstavlja dobru praksu i postala je gotovo neophodna u modernim aplikacijama.

Zahvaljujući validaciji na klijentskoj strani, sprečava se slanje velikog broja zahteva na server koji bi u slučaju nedostatka validacije na serverskoj strani mogli rezultovati ozbiljnim bezbednosnim problemima.

U TestIT aplikaciji implementirana je validacija unetih podataka korišćenjem regularnih izraza za proveru dozvoljenog formata unosa. U slučaju nevalidnosti unetih podataka korisniku se prikazuje obaveštenje. Poruke obaveštenja su dizajnirane tako da ne odaju previše podataka, koje bi napadač mogao da iskoristi.

4. ZAKLJUČAK

Povećanje broja aplikacija na mobilnim telefonima doprinelo je porastu bezbednosnih rizika. Veliki protok podataka među kojima se mogu naći finansijski podaci kao i kredencijali za logovanje na različite sajtove, samo su neki od podataka koje je neophodno zaštititi. Najveći broj bezbednosnih rizika povezan je sa krađom korisničkih podataka, a na programerima je zadatak da te podatke zaštite. Kao pomoć programerima razvijena je *OWASP top ten mobile* lista u kojoj su predstavljene najčešće ranjivosti mobilnih aplikacija zajedno sa preporukama kako ih prevazići.

Primena osnovnih bezbednosnih principa, poput sprečavanja curenja bitnih podataka zabranom slikanja i snimanja ekrana, HTTPS komunikacije i validacije unetih podataka od strane korisnika, rezultovala bi značajno većom bezbednošću aplikacija. Ukoliko bi na razvoju mobilnih aplikacija radili programeri koji imaju svest o značaju eliminisanja bezbednosnih ranjivosti i ako bi prilikom kodiranja ispoštovali bezbednosne preporuke, značajno bi se povećao kvalitet dobijenih aplikacija. Samim tim, značajno bi bila smanjena verovatnoća efikasne realizacije sajber napada.

5. LITERATURA

[1] https://gs.statcounter.com/os-market-share/mobile/worldwide (pristupljeno u septembru 2020.)
[2] https://cve.mitre.org (pristupljeno u septembru 2020.)
[3] https://jeremylong.github.io/DependencyCheck/ (pristupljeno u septembru 2020.)
[4] https://owasp.org/www-project-mobile-top-10/ (pristupljeno u septembru 2020.)
[5] https://five.agency/encryption-on-android-with-jetpack-security (pristupljeno u septembru 2020.)
[6] https://developer.android.com/studio/build/shrink-code#keep-code (pristupljeno u septembru 2020.)

Kratka biografija:

Marijana Matkovski rođena je 18.02.1996. godine u Vrbasu. Završila je Srednju tehničku školu "Mihajlo Pupin" u Kuli 2015. godine. Fakultet tehničkih nauka u Novom Sadu je upisala školske 2015/2016. godine. Osnovne akademske studije završila je 2019. godine. Iste godine upisala je master akademske studije, modul Softversko inženjerstvo.
Kontakt: marijanamatkovski@uns.ac.rs