Abstrakt

Inšpiráciou pre túto prácu bol projekt bývalého spolužiaka v ktorom potreboval vyriešiť prihlasovanie sa do webovej aplikácie. Požiadavkou bolo aby šlo o dvoj-faktorovú autentifikáciu. Riešením je generovanie jednorazových hesiel. Systém , ktorý by posielal heslá cez sms na mobil by bol pre celý projekt finančne nerealizovateľný. Vznikla teda potreba vyvinúť lacné, ideálne bezplatné, riešenie pre daný problém.

Cieľom tejto bakalárskej práce je vyvinúť mobilnú aplikáciu na bezpečné prihlasovanie pomocou generovania jednorazového hesla. Na prihlasovanie sa do backend webovej aplikácie sa prostredníctvom vyvíjanej aplikácie vygeneruje z predtým načítaného QR kódu jednorazové heslo. Generovanie jednorazového hesla z QR kódu prebieha pomocou HOTP (hmac one time password). Aplikácia má bežať offline.

Mobilná aplikácia dostane základné informácie o užívateľovi (po bežnej kontrole prostredníctvom zadaného emailu používateľa) prostredníctvom QR kódu vygenerovaným backend webovskou aplikáciou. Na základe informácií v prístupnom QR kóde vygeneruje jednorázové heslo na prihlásenie sa do backend aplikácie. Backend na webe vykonáva rovnakú činnosť ako mobilná aplikácia. Generovanie jednorázového hesla z QR kódu prebieha pomocou HOTP, TOTP alebo "challenge-response" autentifikáciou. Samotným snímaním QR kódu sa uskutoční prenos citlivých údajov, na základe ktorých sa pomocou HOTP, TOTP alebo challenge response generuje jednorázové prihlasovacie heslo. Citlivé údaje sú zdieľané, teda pôjde o symetrické šifrovanie. Je potreba porovnať HOTP, TOTP a challenge response, a zistiť ktorý spôsob je pre danú aplikáciu najvýhodnejší. Mobilná aplikácia má pracovať v off-line režime.

V nasledujúcich kapitolách postupne približujeme najprv teoretické základy, neskôr implementačnú stránku vyvíjanej aplikácie.

1. Android

Prečo práve android?

Je to momentálne najbežnejší operačný systém používaný na mobiloch. Vyvíjaná aplikácia má fungovať na mobilných zariadeniach, preto sme sa rozhodli jej mobilnú časť vyvíjať na operačný systém android.

Spoločnosť Android Inc. Bola založená v roku 2003 Andym Rubinom, Richom Minerom,Nickom Searsom a Chrisom Whitom. V roku 2005 bola odkúpená spoločnosťou Google. Android je operačný systém postavený na jadre Linuxu , konkrétne verzie 2.6. Príčinou použitia Linuxového jadra bola možnosť jednoduchej kompilácie na rôznych zariadeniach.

Architektúra systému

Operačný systém je rozdelený na 5 základných vrstivev. Vrstvy nemusia byť od seba prísne oddelené.

1. Jadro operačného systému – rozhranie medzi použitím hardvérom a zvyškom softvéru na vyšších vrstvách. V prípade Androidu je jadrom Linux vo verzii 2.6.

2. Knižnice – Napísané v C alebo C++, využívané rôznymi súčasťami systému. Vývojárom sú poskytované prostredníctvom Android Application Framework. Medzi tieto knižnice patria napríklad Media Libraries, SQLite, OpenSSL.

3. Android Runtime – obsahuje aplikačný virtuálny stroj Dalvik vyvynutý špeciálne pre android. Dalvik Virtula Machine je registrovo orientovaná architektúra, využíva základné vlastnosti linuxového jadra, ako je správa pamäte alebo práca s vláknami. Dôvodom vývoja nového virtuálneho stroja bol fakt, že Java Virtual Machine nie je voľne šíriteľný a tiež jeho optimalizácia na mobilné zariadenia hlavne v oblasti pomeru úspory energie a výkonu. Táto vrstva tiež obsahuje základne knižnice jazyka Java.

4. Application framework – podstatná pre vývojárov, poskytuje prístup k službám ,ktoré môžu by použité priamo v aplikáciach. Napríklad dáta iných aplikácií, aplikácie bežiace na pozadí, upozorňovací stavový riadok. Základna sada služieb obsahuje najmä:

    4.1. Sada prvkov View – prvky použité pre zostavovanie používatelskeho rozhrania.

    4.2. Content providers – prístup k obsahu iných aplikácií (napríklad kontakty).

    4.3. Resource manager – prístup k zdrojom (resources) ako grafika, reťazce, pripojené súbory.

4.4. Notification manager – umožnuje zobrazovať upozornenia v stavovom riadku.

4.5. Activity manager – riadi životný cyklus aplikácií, orientácia v zásobníku s aplikáciami.

5. Základne aplikácie – najvyššia vrstva systému. Aplikácie ako Kontakty, Správy, e-mailový klient.

Preklad aplikácie pre Android

Zdrojový Java kód je skompilovaný do Java byte kódu pomocou rovnakého kompilátoru ako pri preklade Java aplikácií. Následne sa Java byte kód prekompiluje pomocou Dalvik kompilátora a výsledný Dlavik byte kód sa spustí na DVM. Všetky spustené Android aplikácie majú svoj vlastný proces s vlastnou inštanciou DVM.

2. One time password

Motivácia

Chceme zabrániť aby sa po odhalení prístupového hesla mohol útočník prihlásiť do aplikácie neobmedzene veľa krát, prípadne systém meniť ak je požadované potvrdenie zmien heslom.

Jednorázové heslo je použité iba pre jedno prihlásenie, transakciu. Tým sa vyhýba problémom , ktoré nastávajú pri odhalení klasického statického hesla. Pri jednorázovom hesle sa po jeho prelomení útočník môže prihlásiť iba raz, prípadne vykonať iba jednu zmenu v systéme alebo transakciu

Generovanie OTP

Algoritmy generujúce OTP zvyčajne využívajú pseudonáhodou, náhodu alebo hash funkcie. Cieľom je aby bolo nemožné z jedného uniknutého otp hesla zistiť nasledujúce. Jednotlivé algoritmy sa výrazne líšia v implementácii, no existuje niekoľko základných spôsobov ako otp heslá generovať.

Spôsoby generovania otp

Synchronizácia na základe času.

V tomto prípade musí byť zhodný čas na serveri aplikácie na ktorú sa klient chce prihlasovať a zariadením u klienta z ktorého sa prihlasuje. Z času sa následne generuje heslo. Kvôli implementácii sa používa isté časové rozmedzie.

Použitie matematickej funkcie

Nové heslo sa generuje na základe predošlého. Heslá sa musia používať v správnom poradí.

Challenge-response

Užívateľ musí serveru zadať správnu odpoveď (heslo) na výzvu , ktorú od neho dostane. Heslo sa počíta pomocou funkcie , ktorá je známa serveru aj užívateľovi.

TOTP – time based done time password

Heslo je generované z aktuálneho času. Klient musí mať na svojom zariadení čas synchronizovaní s časom lokálneho overovacieho servera. Predchádzajúce heslo nemá v tomto prípade žiaden význam. Väčšina metód doručovania jednorazových hesiel používa časovú synchronizáciu

Použitie matematickej funkcie

Každé nové heslo je vytvorené z predošlého použitého. Systém využíva napríklad jednosmernú funkcie f. Prvé heslo je f(x) , druhé f(f(x)) a tak ďalej. Týchto hesiel ja vygenerovaných dostatočné množstvo. Každé heslo je následne použité v opačnom poradí, teda posledné použité heslo bude f(x). Ak sa tieto hesla vyčerpajú, je vygenerovaná nová postupnosť hesiel s iným zadaným x. Útočník, ktorý získa jedno heslo ho môže použiť k jednému prihláseniu sa, k jednej transakcii. Pre získanie ďalšieho hesla by ale potreboval nájsť funkciu inverznú k funkcii f, tá je ale navrhnutá ako jednosmerná, teda nájsť inverznú funkciu je zložité. Ak je ako funkcia f použitá hešovacia kryptografická funkcia , tak ide o výpočtovo nezvládnuteľnú operáciu.

Challenge-response

Overovací server poskytne otázku (challenge), aby klientovi umožnil prístup musí dostať zodpovedajúcu odpoveď (response). Otázkou môže byť reťazec, odpoveďou ten istý reťazec upravený funkciou známou iba serveru a klientovi.

3. HMAC – keyed hash message autehntication code

Kryptografický algoritmus pre výpočet autentifikačného kódu správ (MAC) pomocou kryptogafickej hash funkcie s tajným kryptografickým kľúčom. Definícia a analýza HMAC algoritmu bola prvykrát publikovaná v roku 1996 autormi Mihir Bella, Ran Canetti a Hugo Krawczyk, ktorý tiež vydal RFC 2104. Podobne ako iné MAC algoritmy môže byť použitý zároveň na overenie integrity dát ako aj autentifikáciu správy. Algoritmus môže využívať akúkoľvek hashovaciu funkciu (napríklad SHA-1, MD5). Výsledný algoritmus sa označuje ako HMAC-SHA1 alebo HMAC-MD5. Kryptografická sila výsledného algoritmu závisí na sile hash-funkcie, veľkosti výstupu tejto funkcie , veľkosti a kvalite kľúča. Iteratívna hash-funkcia rozdelí správu do blokov o fixnej veľkosti . Na tieto bloky je potom použitá jednosmerná kompresná funkcia. MD5, SHA1 a SHA2 využívajú Merkle-Damgård funkciu. Veľkosť výstupu HMAC je rovnaká ako veľkosť výstupu použitej hash funkcie. Algoritmus ma podobné časové nároky pre dlhé správy ako použitá hašovacia funkcia.

Definícia HMAC:

HMAC (K,m) = H ((K ⊕ opad) ‖ H ((K ⊕ ipad) ‖ m))

H - kryptografická hašovacia funkcia

K – tajný kľúč

m – správa

⊕ - logická spojka xor

ipad – B krát opakujúci sa bajt 0x36 (B je veľkosť bloku do ktorých je rozdelená správa)

opad- B krát opakujúci sa bajt 0x5C

Výhody HMAC:

V softvérovej implementácii je rýchlejší ako blokové šifry (napr. DES).

Voľne dostupné knižnice hašovacích funkcií.

Hašovacie funkcie nie sú na rozdiel od symetrických šifier limitované.

HMAC-SHA1

Pri použití hašovacej funkcie SHA-1 sa používajú bloky o veľkosti 64 bytov, výstupné bloky majú veľkosť 160 bitov. Autentifikačný kľúč K môže mať maximálnu veľkosť 64 bytov. Ak je väčší ja naňho najprv použitá hašovacia funkcia až potom sa jej výstup použije ako kľúč.

Postup vykonávania HMAC-SHA1

1. Pridanie núl na koniec tajného kľúča tak aby sa vytvoril blok dlhý 64 bytov.
2. Vykonanie bitovej operácie XOR – medzi dátovými blokmi z kroku 1 s doplnkovou funkciou ipad.
3. Pripojenie správy m.
4. Spustiť hašovaciu funkciu SHA1 na dáta z kroku 3.
5. Xor medzi dátovými blokmi z kroku 1 s doplnkovou funkciou opad.
6. Pripojiť hašovací kód z kroku 4 k dátam z kroku 5.
7. Spustiť hašovaciu funkciu na dáta z kroku 6.

HMAC-SHA2

SHA-2 je následníkom SHA-1.Dôvodom vzniku bolo, že v roku 2005 kryptoanalýza preukázala ,že útok na SHA-1 môže byť úspešný, od roku 2010 začal byť odporúčaný prechod na SHA-2. Najrozšírenejšie webové prehliadače končia s podporou SHA-1 na konci roka 2016. Hlavným cieľom pri vývoji SHA2 bolo odstrániť možné rizika SHA1. Pri použití SHA2 sa využije jedna zo 6 verzií tohto algoritmu. Líšia sa dĺžkou výstupu. Existujú funkcie SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, kde číslo označuje počet bitov na výstupe. SHA-256 a 512 operujú s 32, resp. 64-bitovými slovami. SHA-224 a 384 sú iba ich skrátenými verziami. SHA-2 je implementované vo viacerých rozšírených bezpečnostných protokoloch, ako napríklad TLS, SSL a SSH.

Útoky na SHA

Existujú 3 rôzne druhy útoku:

1. Nájdenie kolízie – pre 2 rôzne vstupy vygeneruje SHA ten istý hash. Bol to hlavný problém SHA-0, ktorý nebol odolný voči kolíziam. Pri SHA-1 hrozilo, že zo stúpajím výpočtovým výkonom počítačov bude možné nájsť kolíziu, čo sa nakoniec aj podarilo Bolo nato potrebných 2^{69} operácii. Ďalej sa podarilo toto číslo znížiť na 2^{35} čo predstavovalo bezpečnostné riziko. SHA-2 tento problém odstraňuje.
2. Preimage attack – útok pomocou predobrazu. Útočník sa snaží zistiť vstup x ktorý bude mať po hašovaní hodnotu h(x)=y. SHA funkcie majú tomuto útoku odolať, teda z y nemá byť možné vyrátať pôvodné x.
3.Secondary preimage attack – Útočník pozná x, snaží sa nájsť y rôzny od x, pričom h(x) =h(y). SHA funkcie sa snažia aby nebolo možné v reálnom čase vypočítať dve takéto rôzne x, y.

4. QR kód

QR, skratka Quick Response Code, je označením pre dvojrozmerný čiarový kód.Skladá sa z bielych a tmavých štvorcových oblastí v štvorcovej matici, veľkosť tejto matice je od 21x21 do 177x177 bodov. Vyvynutý bol japonskou spoločnosťou Denso-Wave, pôvodne pre automobilový priemysel. Na kód sa nevzťahuje patentová ochrana . Rozšíreným sa stal vďaka možnosti ho ľahko a rýchlo prečítať, tiež vďaka možnosti zakódovať doň väčšie množstvo dát ako do klasického čiarového kódu. Kód pozostáva z tmavých oblastí v štvorcovej mriežke na bielom pozadí. Tento obrazec je prečítaný dekódovacím zariadením, napríklad fotoaparát alebo skener. Na rozdiel od jedno rozmerného čiarového kódu , ktorý je skenovaný 1 rozmerným pásom svetla, je QR kód čítaný 2 rozmerným digitálnym obrazovým senzorom. Algoritmus lokalizuje 3 švorce v rohoch QR kódu, pričom používa menší štvorec (alebo viac menších štvorcov) v štvrtom rohu aby zistil otočenie a uhol pohľadu na QR kód. Zvyšné oblasti obsahujú dáta. Skenovacie zariadenia používa Reed-Solomon korekciu zosnímaného obrazu až pokiaľ sa dáta nedajú dekódovať.

Využitie

Napriek tomu, že pôvodne sa kód používal v automobilovom priemysle na označovanie časti vo výrobe sa QR kód stal populárnym v reklame. Zvyčajne sa ako skener použije mobilný telefón, ktorý z kódu prečíta napríklad url adresu. QR kódy sa využívajú takisto na poli kryptografických mien, napríklad Bitcoin a jemu podobných. Adresát platby, kryptografické kľúče a informácie o transakcii sa takisto môžu v internetovom bankovníctve zdieľať pomocou QR kódov. Ďalším z využití môže byť prihlasovanie sa na webové stránky. Kódované QR kódy sú využívané napríklad v Japonskom imigračnom systéme. Android aplikácia ktorá QR kód zašifruje a následne odšifruje využíva DES algoritmus.

Dáta v QR kóde

Množstvo uložených dát závisí na type dát, verzie kódu a úrovne korekcie chýb. Kód verzie 40-L (nízka korekcia chýb) dokáže uchovať 2953 8-bitových znakov, teda 2953 znakov podľa normy ISO 8859-1.

Korekcia chýb

Používa sa Reed-Solomon algoritmus. Samotné kódy majú 4 úrovne korekcie chýb. Nízka (L)- 7% údajov môže byť poškodených. Stredná (M) - 15%. Štvrtinová (Q) - 25%. Vysoká (H) - 30%. Vo väčších QR kódoch sa správa rozdelí na viac blokov kontrolovaných Reed-Solomon algoritmom , tak aby v jednom bloku mohlo byť opravených najviac 15 chýb. To limituje náročnosť vykonania opravného algoritmu.

Riziká

Jedinou formou ako môže kód obsahovať spustiteľné dáta je cez url adresu. Adresa môže obsahovať potencionálne škodlivý softvér.

5. Zxing

Pre skenovanie QR kódov sme sa rozhodli použiť knižnicu Zxing. Je to open source projekt pre skenovanie 1D alebo 2D čiarových kódov. Delí sa na niekoľko modulov.
Core- jadro knižnice dekodujucej obraz, testovací kód.
Javase –kód JavaSe špecifický klient.
Android- Barcode Scanner aplikácia pre android, umiestnené aj na Google play.
Androidtest- Android test app, ZXing Test.
Android-integration – podporuje integráciu aplikácie Barcode Scanner do iných aplikácii cez Intent.
Android-core- kód vzťahujúci sa k androidu , zdieľaný prvkami android , android-test, glass.
Glass – aplikácia pre Google Glass.
zxingorg – zdroj stojaci za zxing.org.
zxing.appspot.com – zdroj za barcode generátorom nachádzajúcom sa na zxing.appspot.com.

Táto knižnica v predošlích vydaniach obsahovala niekoľko portov pre iné jazyky, ako napríklad: C++, objektové C, JRuby.

V prototype sme využili triedy IntentIntegrator a IntentResult. Tie sú volané priamo z triedy MainActivity.