UNIVERZITA KOMENSKÉHO V BRATISLAVE FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Bezpečná autorizácia pomocou mobilnej aplikácie

2016

Juraj Mašlej

UNIVERZITA KOMENSKÉHO V BRATISLAVE FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Bezpečná autorizácia pomocou mobilnej aplikácie

Bakalárska práca

Študijný program : Aplikovaná informatika

Študijný odbor: 2511 Aplikovaná informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: RNDr. Peter Borovanský, PhD.

Bratislava 2016

Juraj Mašlej

Obsah

1 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, bezpečné riešenie pre daný problém. Pre prenos tajného kľúča medzi serverom a mobilným zariadením sme zvolili skenovanie qr kódu.

Abstract

Idea of this bachelor project was inspired by proposed project of my ex-classmate in which he needed to solve login in web application. There was demand for two way authentification system. This means need to generate one time passwords. System similar to those used in internet banking, that means sending every otp to client in sms was out of question because of costs. Key aspects of this demanded solution were, security and zero operation costs. With that in mind we decided to choose scanning qr code as a solution to share secret key between server and client.

2 Úvod

Samotný problém zabezpečenia ochrany dát a prístupu k nim v prípade prelomenia bezpečnosti hesla existuje od zavedenia prvého hesla. Cieľom tejto bakalárskej práce je vyvinúť mobilnú aplikáciu pre bezpečné prihlasovanie pomocou generovania jednorazového hesla. Na prihlasovanie sa do webovej aplikácie sa prostredníctvom mobilnej aplikácie vygeneruje z predtým načítaného QR kódu jednorazové heslo. Je potrebné porovnať HOTP, TOTP a challenge response, a zistiť ktorý spôsob je pre danú aplikáciu najvýhodnejší.

Webová aplikácia musí dokázať vygenerovať otp také isté ako mobilná aplikácia. Citlivé údaje sú zdieľané, teda ide o symetrické šifrovanie. Mobilná aplikácia má pracovať v off-line režime. V nasledujúcich kapitolách postupne približujeme najprv teoretické základy, neskôr implementačnú stránku vyvíjanej aplikácie.

3 Prehľad problematiky

3.1 Slovník pojmov

otp – one time password, jednorazové heslo
totp – time based otp, jednorazové heslo generované na základe času
hotp – hmac based otp, jednorazové heslo generované z zdieľaného tajného kľúča

3.2 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 pre 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.

3.2.1 Architektúra systému

Operačný systém je rozdelený na 5 základných vrstiev. 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.

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.

3.2.2 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. Základná sada služieb obsahuje najmä:

4.1. Sada prvkov View - prvky použité pre zostavovanie užívateľského 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 - umožňuje 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.

3.2.3 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 kompiluje pomocou Dalvik kompilátora a výsledný Dlavik byte kód sa spustí na DVM. Všetky spustené Android aplikácie majú svoj vlastný proces s vlastnou inštanciou DVM.

3.3 2. One time password

3.3.1 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.

3.3.2 Generovanie OTP

Algoritmy generujúce OTP zvyčajne využívajú pseudo-náhodu, náhodu alebo hašovacie 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.

3.3.3 Spôsoby generovania otp

3.3.3.1 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.

3.3.3.2 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.

3.3.3.3 TOTP - time based one 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.

3.3.3.4 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.

3.3.3.5 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.

3.3.4 3. HMAC - keyed hash message autehntication code

Kryptografický algoritmus pre výpočet autentifikačného kódu správ (MAC) pomocou kryptografickej hešovacej funkcie s tajným kľúčom. Definícia a analýza HMAC algoritmu bola prvýkrát 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 hešovaciu funkciu (napríklad SHA-1, MD5). Výsledný algoritmus sa označuje ako HMAC-SHA1 alebo HMAC-MD5. Kryptografická sila výsledného algoritmu závisí na sile hešovacej funkcie, veľkosti výstupu tejto funkcie , veľkosti a kvalite kľúča. Iteratívna hešovacia 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 hešovacej funkcie. Algoritmus ma podobné časové nároky pre dlhé správy ako použitá hešovacia funkcia.

3.3.4.1 Definícia HMAC:

HMAC (K,m) = H ((K ⊕ opad) ‖ H ((K ⊕ ipad) ‖m))
H - kryptografická hešovacia 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 bajt0x5C

3.3.4.2 Výhody HMAC:

V softvérovej implementácii je rýchlejší ako blokové šifry (napr. DES). Voľne dostupné knižnice hešovacích funkcií.
Hešovacie funkcie nie sú na rozdiel od symetrických šifier limitované.

3.3.4.3 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.

3.3.4.4 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
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.

3.3.4.5 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.

3.3.5 Útoky na SHA

Existujú 3 rôzne druhy útoku:

1. Nájdenie kolízie - pre 2 rôzne vstupy vygeneruje SHA ten istý heš. Bol to hlavný problém SHA-0, ktorý nebol odolný voči kolíziám. Pri SHA-1 hrozilo, že zo stúpajúcim 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.

3.4 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. Vyvinutý 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 štvorce 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.

3.4.1 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 rozšifruje využíva DES algoritmus.

3.4.2 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.

3.4.3 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.

3.4.4 Riziká

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

3.5 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.

3.5.1 Moduly

Core - jadro knižnice dekódujúcej 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.

Táto knižnica v predošlých 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.

3.6 Flask

Dynamický webový micro-framework založený na pythone využívaný od roku 2010. Ako micro-framework sa zameriava hlavne na malé aplikácie s jednoduchšími požiadavkami. Znamená to tiež, že celá aplikácia môže byť v jednom python súbore. Filozofiou Flasku je udržiavať jadro aplikácie čo najjednoduchšie, no rozšíriteľné.

3.6.1 Porovnanie s frameworkami Django a Pyramid

3.6.1.1 Zameranie

Najstarší z týchto frameworkov je Pyramid, vydaný v roku 2005 ešte pod názvom Pylon. Django nasledovalo v roku 2006. Najmladší je Flask s rokom vydania 2010. K tomu sa viaže aj podpora na fórach, kde vedie jednoznačne Django s viac ako 50 000 otázkami na portáli stack-overflow.

Django aj Pyramid sú zamerané hlavne na väčšie aplikácie,majú ale iný prístup k flexibilite a rozšíriteľnosti. Pyramid sa zameriava na flexibilitu a necháva viac rozhodnutí na vývojára, ako napríklad výber databázy, štruktúry URL a iné. Django ponúka databázový systém, URL schému a inú funkcionalitu pripravenú. Všetky ponúkajú administrátorské rozhranie.

3.6.1.2 Bootstraping

Djnago aj Pyramid obsahujú integrované nástroje pre bootstrap. Flask ich neobsahuje pretože v malých aplikáciach sa s bootstrapom neráta ako s nevyhnutnou súčasťou. Náhrada bootstrapingu vo Flasku pre aplikácie , ktoré potrebujú väčšie rozdelenie medzi komponentami sa používa Blueprint. Ide o odtlačky aplikácie, nie samostatné nové aplikácie. Je to súbor operácií, ktorý je registrovateľný na pôvodnej aplikácii. Pyramid využíva svoju súčasť pcreate. Narozdiel od Flask-blueprints už vytvára väčšiu kostru projektu s konfiguračnými súbormi, skúšobnou template a súbormi pre zbalenie aplikácie a jej nahranie do Python Package Index. Ako pri celom frameworku aj tu

Pyramid kladie dôraz na flexibilitu. Django využíva built in nástroj django-admin. Rozdeľuje projekt do viacerých aplikácií, narozdiel od Pyramid a Flask, ktoré predpokladajú ,že projekt je jedna aplikácia s niekoľkými views a models. Tento druh rozdelenia projektu na viac aplikácií, sa dá vytvoriť aj pri použití Flask alebo Pyramid, no nie je predvolený. Django necháva na vývojára voľbu ako distribuovať aplikácie , čo spôsobuje problémy najmä kvôli chýbajúcej uniformite distribuovania aplikácií.

4 Existujúce riešenia

Zabezpečenia dát proti neoprávnenému prístupu v prípade prelomenia konvenčného hesla nie je nový problém, preto sme zvolili prieskum už existujúcich riešení. Najrozšírenejším a aj medzi laickou verejnosťou najznámejším riešením tohto problému je zasielanie jednorázového hesla v sms, pre našu aplikáciu ale túto službu nemáme k dispozícii, preto sme museli nájsť iné riešenie. Zvolili sme formu prenosu tajného kľúča skenovaním qr kódu. Najrozšírenejším riešením tohto problému je google authenticator. Na prenos kľúča využíva qr kód, ponúka ale aj možnosť zobrazenia kľúča ako textu. Problém synchronizácie mobilnej aplikácie rieši vyzvaním užívateľa aby v nastaveniach mobilu povolil nastavenie času podľa siete. V priebehu prieskumu existujúcich riešení sme stále narážali na problém, že existujúce riešenia nevyhovovali naším požiadavkám prevažne z dvoch hlavných dôvodov. Prvým bola požiadavka na prístup mobilnej aplikácie k internetu, našou požiadavkou je aby mobilná aplikácia bola schopná generovania validných hesiel aj bez internetového pripojenia. Druhým bolo ponechanie riešenia problému nesynchronizovaného času na mobilnej a webovej aplikácii na časový interval pre ktorý je heslo validné. Toto riešenie ale nie je funkčné v prípade ak časový rozdiel medzi mobilnou časťou aplikácie a serverom je väčší ako časový interval platnosti hesla. Tieto dve problémy obsahovali nami skúmané riešenia od spoločností Onelogin a SourceForge. V druhom prípade ide o staršie riešenie vyžadujúce si pomerne komplikovanú inštaláciu na server. Návrh blízky tomu v tejto aplikácii je opísaný v článku [zdroj z lit.]

5 Návrh

5.1 Cieľ

Výsledná aplikácia má poskytovať dvoj-faktorový autentifikačný systém. Užívateľ má skenovať qr kód len pri prvom prihlásení, pri ďalších prihláseniach má už iba generovať nové heslo. Heslo má byť validné v určenom časovom intervale, dostatočnom nato aby ho užívateľ prepísal z obrazovky mobilného zariadenia do webovej aplikácie. V prípade uplynutia tohto intervalu má byť užívateľovi ponúknutá možnosť generovať nové heslo. Aplikácia má fungovať aj keď je rozdiel v systémových časoch medzi serverom a mobilnou aplikáciou väčší ako interval platnosti hesla. Podmienkou je jednoduché ovládanie užívateľom a beh mobilnej aplikácie aj bez internetového pripojenia.

5.2 Use case

5.2.1 Prvá registrácie užívateľa

Pri prvom prihlásení sa do aplikácie užívateľ zadá svoje prihlasovacie údaje, ktoré bude vopred poznať. Následne bude presmerovaný na stránku ktorá mu dá možnosť vybrať si medzi naskenovaním nového QR kódu, alebo zadaním hesla generovaným mobilnou aplikáciou. Návod na stránke ho navedie vybrať si možnosť pre nové skenovanie. Ak by užívateľ aj napriek tomu klikol na možnosť zadania hesla z mobilnej aplikácie bude sa musieť následne vrátiť na stránku s možnosťami, keďže jeho mobilná aplikácia ešte neoskenovala žiaden Qr kód. Po zadaní kódu z mobilnej aplikácie je overený a v prípade správnosti presmerovaný na konečnú stránku. Ak kód správny nie je, užívateľ je vyzvaný aby ho zadal opätovne.

5.2.2 Bežné prihlásenie

Užívateľ sa prihlási svojím menom a heslom, následne je usmernený na možnosť predpokladajúcu, že qr kód má už oskenovaný v mobilnej aplikácii. Na ďalšej stránke zadá heslo vygenerované mobilnou aplikáciou.

5.2.3 Use case mobilnej aplikácie

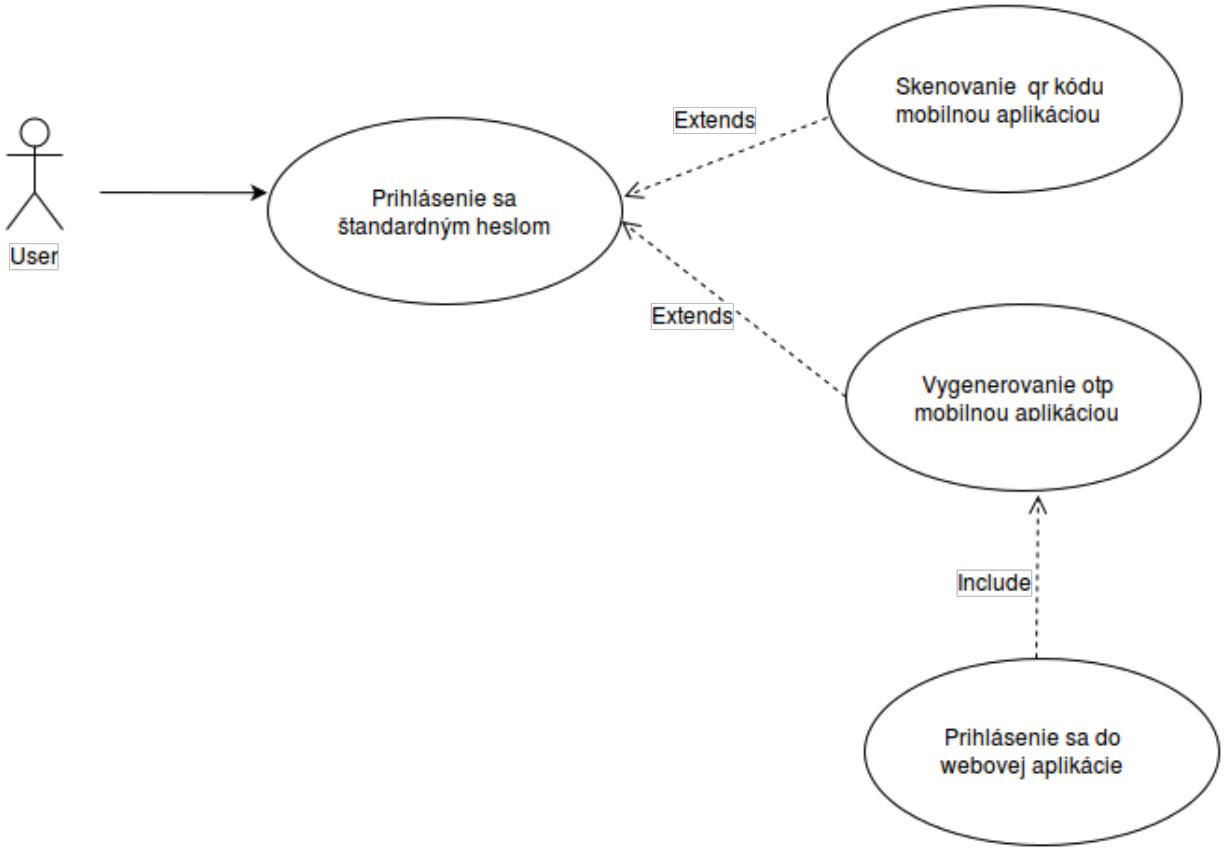
5.2.3.1 Prvé použitie

Užívateľ zvolí možnosť pre skenovanie nového qr kódu, načo sa otvorí skenovanie qr kódu. Po úspešnom skenovaní sa zobrazí heslo pre prihlásenie sa do webovej aplikácie.

5.2.3.2 Bežné použitie

Užívateľ zvolí možnosť pre vytvorenie nového hesla, následne sa otvorí obrazovka s heslom a možnosťou požiadať o ďalšie heslo ak by platnosť pôvodne vytvoreného hesla už vypršala.

5.2.3.3 Use case diagram



5.2.4 Návrh mobilnej aplikácie

Aplikácia je vyvíjaná pre mobilné telefóny s operačným systémom android, nakoľko sa jedna o v lokálnych podmienkach najrozšírenejší mobilný operačný systém.

5.2.4.1 Základné obrazovky

Intro – zobrazí sa pri spustení aplikácie, užívateľ sa ďalej dostane klikom na logo, v kóde je to trieda Intro.
Hlavná obrazovka – prechádza sa na ňu z triedy Intro, obsahuje možnosť voľby medzi skenovaním nového Qr kódu , alebo generovaním nového hesla z už oskenovaného kódu. V programe je to trieda MainActivity.
Obrazovka s heslom – zobrazí sa po zvolení možnosti generovať nové heslo na hlavnej obrazovke, alebo po úspešnom skenovaní.

5.2.4.2 Skenovanie Qr kódu

Hlavnou funkcionalitou mobilnej aplikácie je generovanie nového hesla, pre ktoré ale musí najprv poznať kľúč, ktorý pozná aj server, aby sme vedeli zabezpečiť overenie hesla. Preto som sa rozhodli pre prenos formou skenovania Qr kódu. Rozhodli sme sa využiť knižnicu zxing. V projekte sú z nej implementované 2 triedy IntentIntegrator a IntentResult.

5.2.4.3 Generovanie hesla

Mobilná aplikácia aj server má kľúč z ktorého sa má generovať heslo. Po štúdiu odbornej literatúry sme sa rozhodli využiť hešovací algoritmus SHA-512 , ktorý v súčasnej dobe ešte nie je prelomený. V mobilnej aplikácii sme vytvorili triedu Hash, ktorá vytvára heslo.

5.2.5 Návrh webovej aplikácie

Aplikácia je postavená na frameworku Flask. Návrhový vzor implementovaný týmto frameworkom je model-view-controller.

5.2.5.1 Stránky zobrazované užívateľovi

Prihlasovacia stránka – obsahuje prihlásenie sa cez meno a heslo používateľa a možnosť vytvoriť účet pre nového používateľa.
Vytváranie nového používateľa – užívateľ vypĺňa položky login, email, password, first name, last name.
Hlavná stránka – voľba medzi možnosťami vytvoriť nový Qr kód, ktorý sa má následne skenovať mobilnou aplikáciou, alebo zobrazením formulára pre vyplnenie hesla z mobilnej aplikácie.
Scan stránka- obsahuje Qr kód a formulár do ktorého užívateľ vyplní heslo vytvorene mobilnou aplikáciou.
Vyplnenie hesla – stránka neobsahuje nový Qr kód, iba formulár pre heslo.

5.2.5.2 Generovanie Qr kódu

Využili sme python knižnice ako PIL a qr. Knižnica qr qeneruje priamo qr kód. Ako údaj je posielaný systémový čas zaokrúhlený na stovky sekúnd. PNG súbor je ukladaný do priečinku static.

5.2.5.3 Overenie užívateľom zadaného OTP

Udalosť je obsluhovaná pomocou view scanned. Z tohto view je volaná metóda validate_otp formuláru OneTimeLoginForm. Validate_otp volá metódu generate_passwd generujúcu heslo, ktoré sa má zhodovať s heslom generovaným mobilnou aplikáciou.

5.2.5.4 Administrátorske rozhranie

Framework ponúka prednastavené administrátorské prostredie, ktoré sme v tomto projekte využili. Administrátor môže pridávať a zmazávať bežných užívateľov.

5.3 Generovanie hesla

V priebehu návrhu aplikácie sme sa rozhodovali medzi generovaním hesla pomocou algoritmov TOTP – time based one time password alebo HOTP – hmac based one time password. Obidve algoritmy sme presnejšie opísali v teoretickej časti práce. Pri využití algoritmu HOTP sme ale narazili na potenciálne problémovú situáciu, keby užívateľ klikol generovať nové heslo na mobilnom zariadení viackrát ako by sa prihlasoval do webovej aplikácie, tj. Po jednom alebo viacerých vygenerovaných heslách by sa neprihlásil. To by viedlo k tomu, že mobilná aplikácia by sa dostala o niekoľko hešov vopred oproti webovej aplikácii. Bolo jednou z podmienok, aby mobil užívateľa nemusel byť pre prihlásenie pripojený k internetu, teda mobil nemá ako poslať serveru dáta oznamujúce ,že už je o niekoľko otp vopred. Z tohto dôvodu sme sa rozhodli využiť algoritmus TOTP. Pri tomto algoritme sa ako seed pre hešovaciu funkciu využíva aktuálny čas plus tajný kľúč zdieľaný bezpečným spôsobom medzi užívateľom a serverom.

5.3.1 Synchornizácia času na serveri a mobilnom zariadení

Čas na servery a mobile používateľa sa môže, a v reálnom živote aj bude líšiť. Teda heslá generované aplikáciou v mobile a serverom by sa nezhodovali. Tento problém sme vyriešili nasledovne. Dáta v qr kóde obsahujú aktuány čas, kedy bol qr kód na servery vytvorený a id užívateľa, id je pridávané kvôli zabezpečeniu unikátnosti hesla pre každého užívateľa. Mobilná aplikácia naskenuje qr kód a uloží jeho dáta, k ním pridá svoj aktuálny systémový čas. Potom ako je užívateľom vyžiadané nové heslo sa čas na servery vyráta mobilnou aplikáciou nasledovne. Aktuálny systémový čas mínus systémový čas v momente skenovania qr kódu, táto hodnota je prirátaná k času, ktorý sme získali oskenovaním qr kódu. Týmto výpočtom získame aktuálny čas na servery bez toho aby mobilná aplikácia musela so serverom komunikovať. Perióda pre platnosť jedného hesla je nastavená na 100 sekúnd.

6 Implementácia

6.1 Mobilná aplikácia

Aplikácia je vyvíjaná pre operačný systém android. Minimálna verzia sdk pre beh aplikácie je 10, cieľová je 23. Android manifest aplikácie obsahuje povolenia pre použitie kamery kvôli skenovaniu.

6.1.1 Štruktúra aplikácie

Aplikácia je rozdelená do 3 aktivít, Intro, MainActivity a MainActivity2. Intro je úvodná obrazovka s logom, z ktorej sa užívateľ kliknutím dostane na MainActivity. Tá obsahuje výber medzi skenovaním nového kódu, alebo len generovaním nového hesla. MainActivity2 je zodpovedná za zobrazenie nového hesla, takisto ponúka možnosť po uplynutí časového limitu pre jedno heslo generovať nové heslo. MainActivity a MainActivity2 spolu komunikujú cez intent.

6.1.2 Triedy aplikácie

MainActivity – poskytuje možnosť voľby medzi skenovaním nového qr kódu, alebo iba generovaním nového hesla. Takisto spracováva skenovanie nového qr kódu. Pomocou intentu zaháji nový sken, ak je sken úspešný ukladá dáta v oskenovanom qr kóde, ak je sken neúspešný, alebo ho užívateľ ukončí, vráti sa naspäť na obrazovku s možnosťami. Pre ukladanie dát sú využívané SharedPreferences. Nebola potreba využívať databázu, nakoľko aplikácia potrebuje uložiť iba čas na servery a aktuálny čas v zariadení. Čas na servery je v qr kóde spolu s id užívateľa , ktorý vytvoril skenovaný qr kód. Trieda obsahuje metódy ktoré tieto dáta ukladajú a následne aj metódy k dátam pristupujúce. Ďalšie metódy obsluhujú udalosti nastávajúce v aplikácii, a to vytvorenie triedy (a teda aktivity), klikanie na tlačidlá, ukončenie skenovania, zatvorenie aplikácie. Iniciácia skenovania je obsluhovaná v metóde reagujúcej na stlačenie príslušného tlačidla. Metóda countServerTime vypočíta aktuálny serverový čas.

MainActivity2 – trieda zodpovedná za zobrazenie OTP a možnosť vytvorenia nového OTP, v prípade ak užívateľ zistí, že pôvodne už nie je validné. Pri vytvorení tejto triedy sa do textView zobrazujúceho heslo vloží hodnota získaná priamo z intentu, ktorým

bola táto trieda zavolaná z triedy MainActivity. V prípade požiadavky na nové heslo, je OTP vyrátané priamo v metóde onClick príslušného tlačidla.

Hash – úlohou triedy je poskytnúť zakódované heslo. V konštruktory dostane ako parametre dáta z oskenovaného qr kódu a čas na mobilnom zariadení v čase skenovania. Metóda get_sha_512_SecurePassword vráti vstupný string zakódovaný hešovacou funkciou sha512. Je využitá java knižnica MessageDigest.

IntentIntegrator- trieda integrujúca skenovanie do aplikácie. Inštancia sa vytvára v prípade ak je odkliknutá možnosť pre nový sken (trieda MainActivity). Trieda je poskytovaná knižnicou zxing.

IntentResult- odchytáva výsledok skenovania, obsahuje dáta zo skenovania. Trieda je poskytovaná knižnicou

6.2 Webová aplikácia

Využili sme framework flask. Návrhový vzor využívaný týmto framework-om je model-view-controller.

6.2.1 Model

User- jediný model potrebný pre túto aplikáciu. User ma parametre ako id, meno, login, heslo a iné. Jediná potrebná metóda vracia id užívateľa.

6.2.2 Forms

RegistrationForm- registrácia nového užívateľa.
LoginForm- formulár pre prihlasovanie užívateľa pomocou ním zvoleného mena a hesla.
OneTimeLoginForm- obsahuje pole pre zadanie hesla, metódu na overenie správnosti zadaného hesla validate_otp, metódu generate_passwd pre vytvorenie hesla serverom , táto metóda sa volá z validate_otp.

6.2.3 Views

MyAdminIndexView- spracuváva prihlasovanie , vytváranie nového účtu, odhlasovanie užívateľa.
Scan- vytára nový qr kód pomocou python knižnice qrcode. Vytvára inštanciu OneTimeLoginForm pre prihlásenie sa hneď po skenovaní qr kódu bez potreby preklikávať sa na inú stránku. Po zadaní hesla užívateľom kontroluje jeho správnosť volaním metódy validate_otp z triedy OneTimeLoginForm.
Scanned- vytvorí inštanciu formuláru OneTimeLoginForm, kontroluje platnosť zadaného hesla.

6.2.4 Vytváranie qr kódu

MakeQr - vytvára qr kód použitím makeqr python knižnice, ako dáta mu posiela systémový čas na servery a id užívateľa v tomto poradí, hodnoty sú rozdelené špeciálnym znakom pre ľahšie spracovanie.

6.2.5 Html templates

Base.html – prihlásenie užívateľa, možnosť vytvoriť nový účet. Po prihlásení možnosť výberu medzi skenovaním nového kódu, alebo iba zadaním hesla z mobilnej aplikácie.
BaseUser.html – základná template pre skenovanie nového qr kódu, alebo len zadanie OTP. Z tejto template dedia user.html a userScanned.html.
User.html – template pre sken nového qr kódu, obsahuje tiež form pre zadanie OTP, dedí z baseUser.html
UserScanned.html – template pre zadanie hesla z mobilnej aplikácie bez skenovanie qr kódu. Dedí z baseUser.htm.

6.3 Implementácia riešenia synchronizácie mobilnej a webovej aplikácie

Dáta posielané do qr kódu:

```
def makeQr(user_id):
    qr = qrcode.QRCode(
        version=1,
        error_correction=qrcode.constants.ERROR_CORRECT_L,
        box_size=10,
        border=4,
    )
    time1 = time.time()
    time1 = round(time1)
    time1 = str(time1)
    time1 = time1[:10] ##sekundova presnost
    qr.add_data(time1 + "#" +str(user_id)) ##aktualny cas a unikatne id uzivatela
    qr.make(fit=True)
    return qr.make_image()
```

Iniciácia skenovania qr kódu v mobilnej aplikácii:

```
@Override
public void onClick(View v) {
  if(v.getId()==R.id.scan_button){ //scan
    IntentIntegrator scanIntegrator = new IntentIntegrator(this);
    scanIntegrator.initiateScan();
    Log.d("cas", "current system time while scanning " +
                String.valueOf(this.roundSystemTime()));
  }
      // odchytavanie udalosti kliknutia na ine tlacidlo
```

Získanie výsledku skenovania:

```
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
   //retrieve scan result
        IntentResult scanningResult =
        IntentIntegrator.parseActivityResult(requestCode, resultCode, intent);
        if (scanningResult.getContents() != null) {
           // v pripade if (scanningResult != null) nastane
                nullPointerException ak qr kod nebol naskenovany a uzivatel ukoncil
                kenovanie
      //we have a result
        String scanContent = scanningResult.getContents();
        this.saveQr(scanContent); // ukladanie dat
        this.saveMobileTime(this.roundSystemTime());
        } else {
                Toast toast = Toast.makeText(getApplicationContext(),
                "No scan data received!", Toast.LENGTH_SHORT);
                toast.show();
   }
}
```

Metóda pre vypočítanie aktuálneho času na servery:

```
private long countServerTime(){
        long timePassed = this.roundSystemTime() - this.getSavedMobileTime();
                //time passed from first scan

        Long serverTime = Long.valueOf(this.getSavedQr().substring(0,
        this.getSavedQr().lastIndexOf("#")))  +  timePassed;
        //time in QR code + timePassed
        return serverTime;
}
Spracovanie požiadavky na nové heslo
@Override
public void onClick(View v) {

   if(v.getId()==R.id.scanned){
     String id =
        this.getSavedQr().substring(this.getSavedQr().lastIndexOf("#") + 1);

      this.hash_QR_Time(id + Long.toString (this.countServerTime())
        .substring(0,8));
   }
}
```

Odoslanie dát do hešovacej funkcie, vytvorenie novej aktivity MainActivity2

```
public String hash_QR_Time(String myTime){
        Hash hash = new Hash(this.getSavedQr(), this.getSavedMobileTime());
        hashed = hash.get_SHA_512_SecurePassword(myTime);
        Hash newPasswd = new Hash(this.getSavedQr(), this.getSavedMobileTime());
        Intent intent = new Intent(this, Main2Activity.class);
        intent.putExtra(long_passwd , hashed);
        intent.putExtra(qr, this.getSavedQr());
        intent.putExtra(mobileTime, this.getSavedMobileTime());
        startActivity(intent);
        return  hashed;
}
```

Metóda triedy Hash, vráti heslo vytvorené pomocou sha-512 šifry

```
public String get_SHA_512_SecurePassword(String passwordToHash)
{
   String generatedPassword = null;
   try {
      MessageDigest md = MessageDigest.getInstance("SHA-512");
      byte[] bytes = new byte[0];
      bytes = md.digest(passwordToHash.getBytes
         (java.nio.charset.Charset.forName("UTF-8")));
      StringBuilder sb = new StringBuilder();
      for(int i=0; i< bytes.length ;i++)
      {
         sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).
         substring(1));
      }
      generatedPassword = sb.toString();
   }
   catch (NoSuchAlgorithmException e)
   {
      e.printStackTrace();
   }
   return generatedPassword;
}
```

6.4 Obmedzenia súčasného riešenia

Metóda generovania otp využitá v tejto aplikácii nie je funkčná ak užívateľ po naskenovaní qr kódu zmení čas na svojom mobilnom zariadení. Na výpočet času na serveri sa totiž používa rozdiel v čase v mobilnom zariadení počas skenovania qr kódu a v momente požiadavky na vytvorenie nového otp. Tento rozdiel ale nebude správny ak užívateľ zmení čas na svojom mobilnom zariadení. V skúmaných už existujúcich riešeniach sme sa nestretli s riešením tohto problému, nakoľko niektoré z nich vôbec nezohľadňovali nesynchronizovaný čas medzi serverom a mobilom. Ponúkané riešenie ale funguje aj keď je server v inom časovom pásme ako mobil užívateľa.

6.5 Testovanie

Aplikácia bola testovaná pre rôzne časové rozostupy medzi serverom a mobilnou aplikáciou. Vytvorené metóda fungovala. Potvrdil sa predpoklad vyjadrený v kapitole obmedzenia súčasného riešenia, teda generovanie správneho otp nefunguje v prípade zmeny času užívateľom v mobilnom zariadení.

7 Záver

nalyzovali sme dve rôzne prístupy k vytváraniu otp, HOTP a TOTP. Ako vhodný pre túto aplikáciu sa ukázal TOTP, nakoľko pri HOTP sme nevedeli zabezpečiť zhodu hesiel na strane serveru a mobilnej aplikácie po chybe užívateľa. Po naštudovaní v súčastnosti využívaných hešovacích funkcií sme zvolili funkciu sha-512. Bolo cieľom využiť pre webovú aplikáciu framework ľahko spustiteľný na serveri, preto sme sa po preskúmaní možností rozhodli pre Flask. Ako operačný systém pre mobilnú aplikáciu bol zvolený android, pretože je v lokálnych podmienkach najrozšírenejší.

7.1 Prínos práce

Hlavný prínos práce vidíme v ponúknutí riešenia ako v mobilnej aplikácii zistiť čas na servery len pomocou prvotného skenovania qr kódu, bez nutnosti pripájať mobilné zariadenie na internet. Osobný prínos vidím v oboznámení sa s vytváraním web aplikácie pomocou frameworku

7.2 Možné rozšírenie súčasnej aplikácie

Je potrebné vyriešiť problém s nesprávnym generovaním otp v prípade ak užívateľ posunie čas na mobilnom zariadení. V aktuálnom stave aplikácie je užívateľ upozornení na potrebu znova naskenovať qr kód. Ďalším možným rozšírením je vytvorenie mobilnej aplikácie pre iné operačné systémy ako android, teda iOS a WindowsPhone.