LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

MOBILO LIETOTŅU DATU DROŠĪBA *iOS* VIDĒ

MAĢISTRA DARBS

Autors: **Maksims Moisja**

Stud. apl. Nr. mm13072

Darba vadītājs: asociētais profesors

Dr. Sc. Comp. Edgars Celms

RĪGA 2015

ANOTĀCIJA

Maģistra darba pamatā ir autora novērota problēma, ka daudzi viedtālruņu lietotāji glabā dažādus sensitīvus datus savos telefonos, nedomājot, ka tas nav droši. Autors ir izpētījis, kā *iOS* sistēma nodrošina lietotāja datu drošību, aprakstījis ievainojamības, kuras var ietekmēt gan izstrādātājus, gan lietotājus, ir apskatījis veidus, kā lietotājs un lietotnes izstrādātājs var realizēt datu drošību. Darba autors ir veicis *iOS* lietotņu analīzi, kuras rezultātā tika atrastas vairākas, kas, pēc autora domām, neievēro lietotāja datu drošību tiešā vai netiešā veidā, kā arī veicis ielaušanās testēšanu ar mērķi izpētīt lietotņu darbību no drošības aspekta.

Atslēgas vārdi: *iOS* operētājsistēma, datu drošība, lietotāja dati, lietotne, ielaušanās testēšana, ievainojamība.

ABSTRACT

Paper is based on observation made by the author, that many smartphone users save their sensitive data without thinking about data security and potential risks. Author has researched how iOS operating system secures user data, has described vulnerabilities, which can affect users and developers. Author also has described the ways how user and application developer can secure data. Research author has analysed multiple iOS application, as result some of them in author's opinion fail in securing user data. Author has done penetration testing to find how applications work from security perspective.

Keywords: iOS operating system, data security, application, penetration testing, vulnerability.

AUTOREFERĀTS

Maģistra darba tēmas aktualitāte ir saistīta ar to, mobilo telefonu lietotnes ir kļuvušas ļoti populāras. Tās piedāvā dažāda veida pakalpojumus, lietotāji tās izmanto, lai pirktu ceļojumu biļetes, apskatītu sava bankas konta atlikumu, lasītu e-pastu, līdz ar to, ir jādomā par datu drošību – gan lietotājiem, gan lietotņu izstrādātājiem. Darba autors vēlējās pārbaudīt, kādā veidā un cik droši lietotnes glabā lietotāja datus, kādas ir iespējas, lai lietotņu izstrādātāji un lietotāji varētu pasargāt sensitīvus datus.

Darba ietvaros izmantoti avoti ir tīmekļa vietnes, jo mobilās sistēmas strauji attīstās, un pārsvarā dažādi raksti par mobilo operētājsistēmu jaunākajām ievainojamībām parādās tieši tīmekļa vietnēs. Autors ir izmantojis oficiālo *Apple* vietni, lai gūtu informāciju par *iOS* operētājsistēmas drošības arhitektūru, dažādu drošības firmu, žurnālu un pētnieku vietnes un tīmekļa dienasgrāmatas, lai iegūtu jaunāko informāciju par operētājsistēmas vai lietotņu ievainojamībām. Darba praktiskā daļa ir balstīta uz autora zināšanām, kas ir iegūtas, apmeklējot dažādus kursus, piemēram, „*SecurityTube iPhone Security course and certification*", „*EC-Council Certified Ethical Hacker v8*", kuri bija saistīta ar *iOS* lietotņu drošību.

Darbam tika veltīts pietiekams stundu skaits – ap 200 stundām, daudz laika tika patērēts, meklējot lietotnes, kuras, pēc autora domām, neievēro lietotāja datu drošību. Darba rezultātā tika atrastas vairākas lietotnes, kuras, pēc autora domām, nerūpējās datu drošību, kā arī divām lietotnēm tika veikta ielaušanās testēšana. Darba gaita ir detalizēti aprakstīta; dažās vietās, lai saprotamāk izskaidrotu veiktās darbības, ir pievienoti ekrānuzņēmumi.

Darba kvalitātes nodrošināšanai tikai izmantota programmatūra, kas nodrošina gramatikas pareizrakstību. Maģistra darbs ir noformēts atbilstoši Latvijas Universitātes Datorikas fakultātes Maģistra darba izstrādes un aizstāvēšanas metodisko norādījumu 3. nodaļas prasībām.

Ievainojamību apraksti, attēli un cita veida informācija, kas ir aizgūta no citiem autoriem, ir atzīmēta kā atsauces. Vietās, kur ir izmantots autoru teiktais, tika atzīmēts kā citāts.

SATURS

APZĪMĒJUMU SARAKSTS

Termins vai saīsinājums angļu valodā	Tulkojums latviešu valodā	Skaidrojums
App Sandbox	Lietotnes glabātuve	*iOS* sistēmā norobežota no apkārtējās vides datu glabātuve.
Apple ID	Apple ID	Lietotāja vārds, kas ir izveidots, lai varētu iegādāties dažādus *Apple* piedāvātos pakalpojumus.
Apple TV	Apple TV	Digitālais multivides atskaņotājs, kuru ražo *Apple.*
AppStore	AppStore	*Apple* pakalpojums, kas ļauj nopirkt un lejupielādēt lietotnes *iOS* sistēmai.
Cydia	Cydia	Lietotne, kas tiek instalēta, veicot *jailbreak* procesu, kurā iespējams lejupielādēt dažādas lietotnes vai servisus.
Crypto Engine	Šifrēšanas dzinējs	Šifrēšanas programma, kuru izmanto aparātprogrammatūrā.
CVSS – common vulnerability scoring system	CVSS	Standarts, kurš palīdz novērtēt sistēmas drošības ievainojamības nozīmīgumu [1].
DoS – Denial of Service	Pakalpojumatteices uzbrukums	Uzbrukums, kura mērķis ir liegt lietotājam izmantot pakalpojumu un kurš tiek realizēts, izmantojot ievainojamības tīklā vai sistēmā [2].
Effaceable Storage	Dzēšama glabātuve	Datu glabātuve, kuru ir iespējams dzēst noteiktos gadījumos.

iOS	iOS	Kompānijas *Apple* izstrādāta operētājsistēma, kas ir paredzēta ierīcēm *iPhone*, *iPad* un *iPod*.
iPad	iPad	Planšete, kuru ražo kompānija *Apple*.
iPhone	iPhone	Viedtālrunis, kuru ražo kompānija *Apple*.
iPod touch	iPod touch	Mūzikas atskaņotājs, kuru ražo kompānija *Apple.*
Jailbreak	Jailbreak	Process, kura laikā tiek noņemti dažādi ierobežojumi *Apple* operētājsistēmai *iOS.*
Mac OS X	Mac OS X	Kompānijas *Appl"* izstrādātā operētājsistēma, kas ir paredzēta *Apple* ražotiem datoriem.
MITM – Man-in-the-Middle	MITM	Termins kriptogrāfijā, ar kuru raksturo uzbrukumu, kura laikā uzbrucējs pārtver datus un tos modificē [3].
Passcode	Ieejas kods	Kods, kas tiek pieprasīts, lai atbloķētu *iOS* ierīci.
RADIUS	RADIUS	Protokols, kas ļauj realizēt autentifikāciju, autorizāciju un informācijas iegūšanu par izmantotiem resursiem, kas ir izstrādāti sakaru realizācijai starp galveno platformu un ierīci [4].
Root	Saknes lietotājs	Lietotājs, kuram ir augstākās tiesības sistēmā, administrators.
Secure Element	Drošais elements	Mikroshēma, kurā tiek

		uzglabāta kredītkartes ģenerētā informācija, kas tiek izmantota, veicot pirkumu ar *iPhone.*
Secure Enclave	Drošības anklāvs	Procesors, kas apstrādā datus no pirksta nolasīšanas procesa.
SSH	SSH	Tīkla protokols, kurš ļauj drošu datu pārraidi starp divām ierīcēm tīklā.
Touch ID	Touch ID	Pirksta nospieduma sensors, kurš ir pieejams ierīcēs *iPhone 5s*, *iPhone 6* un *iPhone 6+*.
UID	Unikāls identifikators	Simbolu kombinācija, kas palīdz precīzi identificēt katru ierīci
UITextField	Teksta lauciņš	Grafisks elements, kas ir paredzēts teksta ievadei
WPA2 Enterprise	WPA2 Enterprise	Drošības metode, kas ir paredzēta bezvadu tīkliem, tā šifrē datus, kas tiek pārraidīti, izmantojot bezvadu tīklu, kā arī tiek izmantota, lai autentificētu lietotājus [5]
Zero-day	Nulltās dienas ievainojamība	Ievainojamība, kuru izstrādātājs neapzinās, tas ir, ievainojamība nav publiskota [6].

IEVADS

Mūsdienās mobilo telefonu lietotņu daudzveidība ir ļoti liela. Lietotnes piedāvā dažāda veida pakalpojumus, piemērām, apskatīt bankas kontu, iegādāties dažādas preces, sarakstīties ar draugiem, pierakstīt svarīgu informāciju. Cilvēki uzticas savai ierīcei, glabājot tur svarīgu un sensitīvo informāciju. Bet cik tas ir droši? Cik droša ir ierīce, operētājsistēma, un lietotnes, kuras tiek izmantotas ikdienā?

Maģistra darba mērķis ir izpētīt operētājsistēmas *iOS* datu drošību, kā arī veikt lietotņu testēšanu, pārbaudot, vai lietotāji dati tiek droši glabāti.

Maģistra darba uzdevumi:

- izpētīt un aprakstīt operētājsistēmas jaunākās ievainojamības;
- izpētīt un aprakstīt ievainojamības lietotnēs;
- veikt ielaušanās testēšanu lietotnēm, ar mērķi atrast tajā iespējamās ievainojamības;
- iepazīties ar veidiem, kā lietotņu izstrādātājs un pats lietotājs var pasargāt savus un lietotāja datus

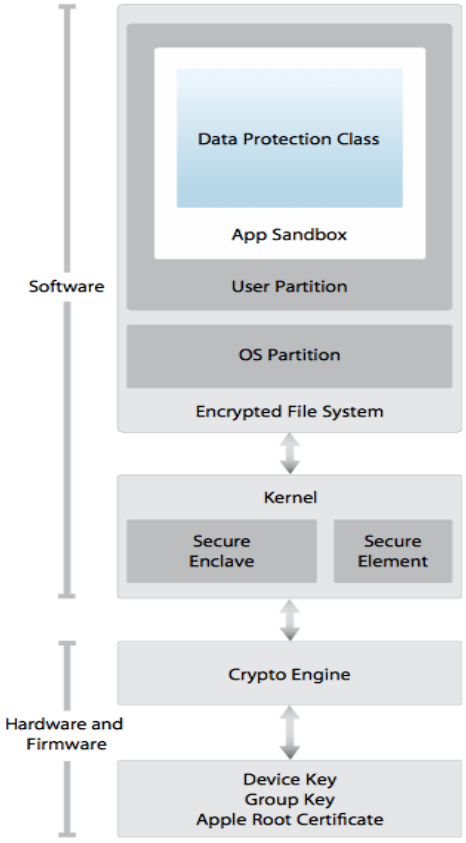
1. DATU DROŠĪBA OPERĒTĀJSISTĒMĀ *iOS*

Apple iOS operētājsistēmas drošība tiek realizēta gan programmatūras līmenī, gan aparatūras līmenī. *iOS* sistēma piedāvā iespējas gan lietotājam, gan izstrādātājam rūpēties par datu drošību. Lietotņu izstrādātājam ir iespēja izmantot dažādas klases, kuras piedāvā vairākas ar drošību saistītas funkcijas. *Apple* ierīcēs lietotājam ir piedāvāti dažādi iestatījumi, kuri ir saistīti ar datu drošību.

1.1 *iOS* drošības arhitektūras apraksts

iOS platformas pamatā ir drošība. *iOS* drošība tiek kontrolēta gan programmatūras līmenī, gan aparatūras līmenī. *iOS* aizsargā ne tikai ierīci un datus tajā, bet arī visu sistēmu, ieskaitot visu, ko lietotājs veica lokāli, tīklā un ar tīmekļa servisiem.

iOS drošības arhitektūras diagramma ir redzama attēlā 1.1.



1.1 att. ***iOS* drošības arhitektūra [7]**

Data Protection Class (*iOS* sistēmas klase, kas ir atbildīga par datu drošību) — kad jauns fails ir izveidots uz ierīces ar *iOS* operētājsistēmu, lietotne, kas ir izveidojusi failu, tam pievieno klasi. Katra klase izmanto dažādas politikas, lai noteiktu, kad dati ir pieejami.

Pamata klases un politikas veidi [7]:

1) pilnīga aizsardzība (*NSFileProtectionComplete* klase) – ja lietotājs nobloķē ierīci, dati

vairs nav pieejami. Lai datus padarītu pieejamus, ierīce ir jāatbloķē, ievadot atbloķēšanas kodu vai izmantojot *Touch ID* sensoru;

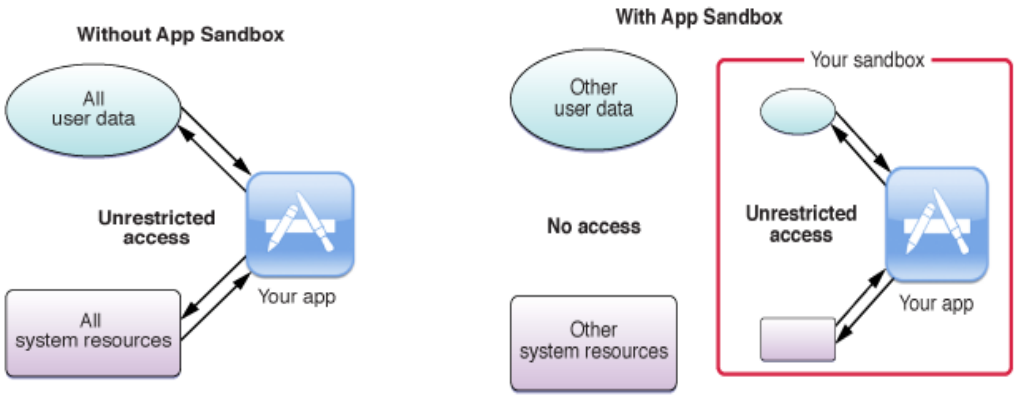
2) aizsargāt, ja nav atvērts (*NSFileProtectionCompleteUnlessOpen* klase) – klase ir jāizmanto, ja ir nepieciešams rakstīt failā, kamēr ierīce ir nobloķēta. Piemēram, ja ir nepieciešams lejupielādēt e-pasta pielikumu fonā (*Background mode*);

3) aizsargāt līdz pirmajai lietotāja autentifikācijai (*NSFileProtectionCompleteUntil UserAuthentication* klase) – klase strādā līdzīgi *NSFileProtectionComplete* klasei. Vienīgā atšķirība ir tāda, ka klases atslēga netiek dzēsta no atmiņas, kad lietotājs ir nobloķējis ierīci. Tā tiek lietota pēc noklusējuma visiem trešās personas lietotņu datiem;

4) nekad neaizsargāt (*NSFileProctionNone* klase) – klases atslēga tiek aizsargāta ar *UID* (*Unique Identifier*) un tiek turēta dzēšamajā glabātuvē (*Effaceable Storage*). Ja failam netiek piesaistīta datu aizsardzības klase, tas joprojām tiek glabāts šifrētā veidā.

App Sandbox — iOS sistēmā katrai lietotnei ir piešķirta sava noslēgta vide. Tas nozīmē, ka viena lietotne nezina par otru, tātad nevar izmantot otras lietotnes datus.

Attēlā 1.21 (a) ir parādīts, kā izskatās sistēma bez *App Sandbox*, savukārt attēlā 1.2 (b) var redzēt sistēmu ar *App Sanbox*.



1.2 att. **Lietotnes datu glabātuve [7]:**

(a) glabātuve bez *App Sandbox;* (b) glabātuve ar *App Sandbox*

Diska nodalījums (*User partition)* un operētājsistēmas nodalījums (*OS partition)* — divi nodalījumi, kuri ir paredzēti atbilstoši lietotāja datiem — trešo pušu lietotnēm, mūzikai, bildēm, u.c. — un *iOS* sistēmas failiem. Kā var redzēt attēlā 1.1, lietotāja datu nodalījums ir atdalīts no sistēmas nodalījuma, kas nodrošina, ka lietotne nevar modificēt sistēmas failus.

Drošības anklāvs (*Secure Enclave)* ir papildus procesors, kas ir iebūvēts *Apple A7* vai vēlākos *A* tipa procesoros. Tas izmanto pastāvīgo drošo ielādi un atdalītu no lietotņu procesora programmatūras atjaunināšanu. Drošības anklāva galvenais uzdevums ir apstrādāt pirksta nospieduma datus no *Touch ID* sensora un noteikt, vai tas atbilst sistēmā reģistrētiem pirksta nospiedumiem.

Drošais elements (*Secure Element*) — mikroshēma, kura parādījās, sākot ar *iPhone 6* un *iPhone 6+* izlaišanu, kas atbalsta maksāšanas iespējas. Tas ir paredzēts, lai glabātu kredītkartes informāciju.

Šifrēšanas dzinējs (*Crypto Engine*) — katrā *iOS* ierīcē ir iebūvēta *AES* 256 šifrēšanas programma, kura atrodas atmiņas tiešpiekļuves (*DMA – Direct Memory Access*) posmā starp zibatmiņas glabātuvi un galveno sistēmu atmiņu, padarot failu šifrēšanu ļoti ātru un resursietaupīgu.

1.2 Lietotnes koda parakstīšana

Tiklīdz *iOS* kodols ir ielādēts, tas kontrolē, kuri lietotāja darbinātie procesi un lietotnes var tikt palaistas. Lai nodrošinātu, ka visas lietotnes nāk no zināma un apstiprināta avota un ka ar tām nav notikušas modifikācijas, *iOS* pieprasa, lai viss izpildāmais kods būtu parakstīts ar sertifikātu, kuru ir izsniedzis *Apple* [7]. Lietotnes, kuras nāk kopā ar ierīci, piemēram, *"Mail"* un *"Safari"*, ir parakstījis *Apple*. Trešo pušu lietotnēm ir jābūt validētam un parakstītam, izmantojot *Apple* izsniegtu sertifikātu. Obligātā parakstīšana nodrošina operētājsistēmas uzticību lietotnēm un aizliedz trešo pušu lietotnēm ielādēt neparakstītu koda resursus vai modificēt kodu.

Lai izstrādātu un instalētu lietotnes uz *iOS* ierīcēm, izstrādātājiem ir jāreģistrē *Apple* kontu un jāpievienojas maksas *iOS* izstrādātāju programmai (*iOS Developer Program*). Katru izstrādātāja identitāti – fiziska vai juridiska persona – *Apple* verificē pirms tiek izsniegts sertifikāts. Šis sertifikāts ļauj izstrādātājiem parakstīt lietotnes un iesniegt tās *AppStore* izplatīšanai. Tā rezultātā visas lietotnes, kuras ir atrodamas *AppStore*, ir iesniegušas identificējamas personas vai organizācijas, kas kalpo kā aizsardzības pasākums pret ļaunprātīgām lietotnēm. *Apple* arī veic lietotņu recenziju, nodrošinot, ka lietotnē nav kļūdu un ka tā veic darbības, kuras ir tās aprakstā.

Atšķirībā no citām mobilajām platformām, *iOS* neļauj lietotājiem instalēt potenciāli ļaunprātīgas lietotnes no tīkla vietnēm vai laist neuzticamu — neparakstītu — kodu. Izpildlaikā tiek pārbaudīts, vai lietotne nav modificēta kopš tā tika uzinstalēta vai atjaunināta [7].

1.3 Drošība izpildlaikā

Tiklīdz lietotne ir verificēta un ir apstiprināts, ka tā nāk no droša avota, *iOS* īsteno aizsardzības pasākumus, kuri ir paredzēti, lai lietotne nevarētu sabojāt citas lietotnes vai pārējo sistēmas daļu [7].

Visas trešās puses lietotnes tiek novietotas norobežotā vidē — *Sandbox*, tas ir, tām ir liegta pieeja failiem, kuri pieder citām lietotnēm, kā arī nav ļauts veikt izmaiņas ierīces darbībā. *Sandbox* liedz lietotnēm iegūt vai modificēt informāciju, kuru glabā citas lietotnes. Katrai lietotnei ir

izveidota unikāla mājas direktorija tās failiem, kura (tās adrese) ir gadījuma veidā piešķirta, kad tā tiek uzinstalēta. Ja trešās puses lietotnei ir nepieciešama papildus informācijas piekļuve, tad to nodrošina *iOS* sistēma ar nodefinētu servisu palīdzību.

Sistēmas faili un resursi arī ir norobežoti no trešo pušu lietotnēm. Lielāka daļa *iOS* procesu strādā kā nepriviliģēts lietotājs – *"mobile"* –, tāpat strādā arī visas trešo pušu lietotnes. Visa operētājsistēmas nodalījums ir ielādēta tikai ar lasīšanas tiesībām (*read-only*) [7]. Nevajadzīgās programmas, piemēram, attālinātās piekļuves servisi, nav iekļauti sistēmas programmatūrā. Lietojumprogrammas interfeiss (*API – Application Program Interface*) neļauj lietotnēm palielināt savas privilēģijas, lai modificētu citas lietotnes vai pašu operētājsistēmu.

1.4 Tīkla drošība

iOS izmanto un piedāvā daudzas tīkla drošības iespējas, lai nodrošinātu informācijas aizsardzību, kad tā ceļo no un uz *iOS* ierīci.

Mobilo telefonu lietotāji slēdzas klāt tīkliem visā pasaulē, tāpēc ir ļoti svarīgi nodrošināt, ka viņi ir autorizēti un viņu dati ir aizsargāti pārraides laikā. *iOS* izmanto un dod izstrādātājiem piekļuvi standarta tīkla protokoliem, to autentifikācijai, autorizācijai un šifrētai datu pārraidei. Lai nodrošinātu minētos aizsardzības uzdevumus, *iOS* izmanto akceptētās tehnoloģijas un jaunākos standartus bezvadu tīkla un datu pārraides pieslēgumos.

Citās platformās ir nepieciešama ugunsmūra programmatūra, lai aizsargātu atvērtos komunikācijas portus pret ielaušanos. *iOS* sistēmā nav uzstādīts ugunsmūris, jo klausīšanas portu skaits ir ierobežots, nevajadzīgas tīkla programmas, piemēram, *"Telnet"*, čaulas (*shells*), tīkla serveri, nav uzstādīti uz *iOS* ierīcēm.

iOS atbalsta drošligzdu slāņa v3 (*SSL – Secure Socket Layer*) protokolu, transporta slāņa drošības *TLS v1.0, TLS v1.1, TLS v1.2* (*TLS – Transport Layer Security*) protokolu un datagrammas transporta slāņa drošības (*DTLS – Datagram Transport Layer Security*) protokolu [7]. Lietotnes "*Safari*", "*Calendar*", "*Mail*" un citas tīmekļa lietotnes automātiski izmanto šos mehānismus, lai padarītu iespējamu šifrētu datu pārraides kanālu starp ierīci un tīkla servisiem.

iOS ierīces atbalsta drošus tīkla servisus, piemēram, virtuālos, privātos tīklos *(VPN – Virtual Private Network*). Parasti ir nepieciešama neliela konfigurācija ierīces ietvaros. *iOS* ierīcēs strādā ar *VPN* serveriem, kuri atbalsta tādus protokolus un autentifikācijas metodes [7]:

- *Cisco, Open VPN*, izmantojot atbilstošo lietotni no *AppStore*;
- *Cisco IPSec* ar lietotāja autentifikāciju, izmantojot paroli, *RSA SecureID* vai *CRYPTOCard*;
- *L2TP/IPSec* ar lietotāja autentifikāciju, izmantojot *MS-CHAPV2* paroli, *RSA SecureID* vai *CRYPTOCard*;

- *PPTP* ar lietotāja autentifikāciju, izmantojot *MS-CHAPV2* paroli, *RSA SecureID* vai *CRYPTOCard*.

iOS atbalsta *Wi-Fi* standartu protokolus, ieskaitot *WPA2 Enterprise*, kas nodrošina autentificētu piekļuvi korporatīvo bezvadu tīkliem. *WPA2 Enterprise* izmanto 128 bitu *AES* šifrēšanu, kas nodrošina lietotājiem pārliecību, ka viņu dati ir aizsargāti to pārraidīšanas un saņemšanas laikā, izmantojot bezvadu tīklu. *iOS* ierīces atbalsta arī 802.1X standartu, kas nodrošina iespēju pieslēgties dažādu *RADIUS* (*Remote Authentication in Dial-In User Service*) vides autentifikācijai. 802.1X bezvadu autentifikācijas metodes, kuras atbalsta *iPhone* un *iPad*, iekļauj: *EAP-TLS, EAP-TTLS, EAP-FAST, EAP-SIM, PEAPv0, PEAPv1* un *LEAP* [7].

2. *iOS* IEVAINOJAMĪBAS

Darba autors maģistra darba ietvaros izpētīja šādas ievainojamības, kas ir saistītas ar *iOS* operētājsistēmu, bibliotēkām un lietotnēm:

1. *jailbreak*;
2. *iOS* operētājsistēmas ievainojamības:
    - "*No iOS Zone*";
    - "*Masque Attack*";
3. ievainojamība bibliotēkā "*AFNetworking*";
4. ievainojamības *iOS* lietotnēs:
    - „*WiFi Drive Pro v1.2 iOS*",
    - „*Photo Manager Pro 4.4.0 iOS*";
    - „*Folder Plus 2.5.1 iOS*";
    - „*Grindr 2.1. iOS*;
    - „*PayPal Inc iOS 4.6*"

2.1 *Jailbreak*

2.1.1 Jailbreak apraksts

Jailbreak (izkļūšana no cietuma) ir process, kura laikā tiek noņemti dažādi ierobežojumi *Apple* operētājsistēmai *iOS*. Process tiek veikts, izmantojot programmatūras un aparatūras ievainojamības. *Jailbreak* tiek veikts dažādām *Apple* ierīcēm — *iPhone*, *iPad*, otrās *paaudzes Apple TV*. Kad uz ierīces ir uzstādīts *jailbreak*, lietotājam ir *root* piekļuve *iOS* failu sistēmai, kas ļauj lejupielādēt papildus lietotnes, papildinājumus, tēmas, kuras nav pieejams, izmantojot oficiālo *Apple* lietotņu veikalu — *AppStore*.

Jailbreak ir sava veida privilēģiju paaugstināšana sistēmā, kas ir līdzīga citām operētājsistēmām, piemēram, *Windows* un *Ubuntu*. Ierīces, kurām ir uzstādīts *jailbreak*, var joprojām lejupielādēt lietotnes no *App Store*, lietot *Apple* programmas, piemēram, „*iTunes*", un izmantot visas ierīces pamatfunkcijas.

Viens no iemesliem, kāpēc lietotājs gribētu lietot *jailbreak* ierīci, ir tāda, ka *jailbreak* ļauj lietot telefonus, kuri ir slēgti kādam noteiktajam mobilo sakaru operatoram, ar citiem mobilo tīklu operatoriem, piemēram, ja lietotājam *iPhone* ir paredzēts lietot tikai LMT operatoru, tad pēc *jailbreak* ierīci būtu iespējams lietot ar Tele2 vai Bite *SIM* kartēm.

Vēl viens iemesls ir tāds, ka *jailbreak* ierīci ir iespējams modificēt, pārvaldīt daudz vairāk. Piemēram, ir iespējams mainīt lietotāja saskarni, uzstādīt lietotnes, kas ļauj pārvaldīt failu sistēmu u.c.

Jailbreak process ir legāls vairākās valstīs, piemēram, ASV, kur atrodas *Apple* galvenā ēka. Taču, veicot ierīcei *jailbreak*, tiek zaudēta garantija, ierīce var tikt nobloķēta (*bricked*), tāpēc katram lietotājam pašam ir jāizvēlas, vai viņš vēlās veikt *jailbreak* vai nē.

2.1.2 Jailbreak vēsture

Pirmā *jailbreak* ierīce parādījās 2007. gadā 24. augustā, kad septiņpadsmit gadu vecs hakeris Džordžs Hotzs (*George Hotz*) ar iesauku *GeoHot* veiksmīgi noņēma bloķēšanu, kas ļāva ASV telekomunikācijas kompānijai „*AT&T*" būt vienīgajai, kuru tīklos strādāja *Apple* pirmais *iPhone* [8]. Bloķēšanas noņemšana ļāva izmantot *iPhone* telefonu ar jebkuru *SIM* karti ar jebkuru mobilo sakaru operatoru visā pasaulē.

2007. gada 11. septembrī komanda „*Phone DevTeam*" sāka piedāvāt divas bezmaksas atbloķēšanas programmas ar grafisko saskarni — „*AnySim*" un „*iUnlock Reloaded*" [9]. Tajā pašā gadā 24. septembrī *Apple* nāca ar paziņojumu, ka atbloķēšanas programmas var sabojāt *iPhone* programmatūru, un kā arī šīs ierīces var tikt padarīt par nesaderīgām ar turpmākiem operētājsistēmas atjauninājumiem [9].

Jailbreak sāka piesaistīt plašu cilvēku auditoriju 2008. gada vasarā, kad lietotne „*Cydia*" tika iekļauta komandas „*iPhone Dev Team*" *jailbreak* versijā, kas bija paredzēta *iPhone 3G* ar *iOS 2.0* versiju [8]. Džejs Frīmens (*Jay Freeman*), kurš ir zināms kā *Saurik*, tiek uzskatīts par *jailbreak* dibinātāju, jo izstrādāja *Cydia* — lietotni, ar kuras palīdzību var uzinstalēt lietotnes un papildinājumus *jailbreak* ierīcēm. *Cydia* var uzskatīt kā *Apple AppStore*, kas ir paredzēta *jailbreak* aplikācijām, tomēr *Cydia* piedāvā daudz vairāk: ir iespējams uzstādīt dažādus operētājsistēmas uzlabojumus, sīkrīkus (*gadget*), papildinājumus.

Uz darba rakstīšanas brīdi iOS versija, kurai ir iespējams veikt *jailbreak*, ir 8.1. Katru gadu, kad iznāk jauna *iOS* versija, kādu laiku vēlāk parādās arī atbilstošā *jailbreak* versija. Tas liecina par *jailbreak* popularitāti, līdz ar to ir jāpadomā kādi dati tiek glabāti uz *Apple* ierīcēm.

2.1.3 Argumenti pret jailbreak

iOS operētājsistēma tika izstrādā kā droša un uzticama vide. Iebūvētie drošības mehānismi pasargā no ļaunprogrammatūras vīrusiem un palīdz nodrošināt pieeju privātai informācijas un uzņēmuma datiem. Neautorizētie uzlabojumi, tā saucamais *jailbreak*, „tiek garām" drošības mehānismiem *iOS* sistēmā, kas var izraisīt dažādas problēmas uzlauztajai *iPhone, iPad* vai *iPod touch* ierīcei, ieskaitot [10]:

1. drošības ievainojamības. *Jailbreak* procesa rezultātā ierīcei tiek likvidēti drošības līmeņi, kuri tika izveidoti, lai pasargātu lietotāja personālo informāciju un pašu *iOS* ierīci. Ja šī drošība ir likvidēta, uzbrucējs var nolaupīt lietotāja datus no ierīces, sabojāt ierīci, uzbrukt tīklam, kurā atrodas ierīce vai ieviest ļaunprogrammatūru, vīrusus, spiegprogrammatūru;
2. nestabilitāte. Bieža un negaidīta ierīces darba pārtraukšana, iebūvēto un trešo pušu

lietojumprogrammatūru avārijas un iesaldēšana (*freeze*) un datu pazušana;

3. saīsināta akumulatora darbība. Lietojumprogrammatūra, kas ir derīga tikai *jailbreak* ierīcei, paātrina akumulatora izlādēšanas procesu un saīsina *iPhone, iPad* un *iPod touch* darbību no uzlādēšanas procesa;
4. neuzticama balss un datu pārraides procesi. Pārtraukti zvani, lēni un neuzticami datu pārraidīšanas, iekavēta un neprecīza atrašanās vietas aprēķināšana;
5. pakalpojumu pārtraukšana. Tādu lietojumprogrammu darbība kā „*Visual Voicemail*", „*Weather*", „*Stock*" tiek pārtraukta vai arī vairs nav pieejama. Papildus trešo pušu lietojumprogrammām, kuras izmanto pakalpojumu „*Apple Push Notification Service*", ir reģistrētas problēmas ar šī pakalpojuma ģenerēto paziņojumu saņemšanu vai tādu paziņojumu saņemšana, kuri tika paredzēti citai uzlauztajai ierīcei. Citiem pakalpojumiem, kā piemēram, „*iCloud*" un „*Exchange*", ir bijušas problēmas ar datu sinhronizēšanu ar atbilstošajiem serveriem;
6. nepieejamība turpmāko lietojumprogrammatūras atjauninājumiem. Dažas neautorizētas modifikācijas ir izraisījušas nelabojamus bojājumus *iOS* ierīcēm. Rezultātā uzlauztajām *iPhone, iPad* un *iPod touch* ierīcēm vairs nav iespējams uzinstalēt *Apple* piedāvātos programmatūras atjauninājumus.

Kompānija *Apple* iesaka ar uzmanību attiekties pret programmatūras instalēšanu, kuras rezultātā tiek uzlauzta ierīce. Ir svarīgi pieminēt, ka jebkura neautorizēta *iOS* operētājsistēmas modifikācija ir gala lietotāja programmatūras licences vienošanās pārkāpums, un, līdz ar to, *Apple* var atteikt pakalpojumus *iPhone, iPad* un *iPod touch* ierīcēm, uz kurām ir uzinstalēta neautorizēta lietojumprogrammatūra [10].

2.2 *iOS* operētājsistēmas ievainojamības

2.2.1 "No iOS Zone" ievainojamība

Drošības speciālisti atklāja *zero-day* ievainojamību operētājsistēmā *iOS 8,* kuras dēļ lietojumprogrammatūras uz *iPhone, iPad* un *iPod* ierīcēm pārstāj darbību (*crash*), kad tās izmanto ļaunprātīgi izveidotu *Wi-Fi* tīklāju (*hotspot*). Ievainojamība izpaužas kā *DoS* uzbrukums uz *Apple iOS* ierīcēm, kā rezultātā pārstāj darboties lietojumprogrammas vai pat pati ierīce.

Adi Šarbani (*Adi Sharbani*) un Jairs Amits (*Yair Amit*) no Mobilo telefonu drošības uzņēmuma „*Skycure*" prezentēja 2015. gada aprīlī savu pētījumu ar nosaukumu „*No iOS Zone*" *RSA* drošības konferencē San Francisko [11]. Viņi parādīja, ka uzbrucējam ir iespējams izveidot ļaunprātīgus bezvada tīklus ar nolūku apstādināt tuvumā esošo lietotāju mobilās ierīces ar ļoti lielu precizitāti. Šī uzbrukuma dēļ ir pilnīgi neiespējami lietot ierīci ar *iOS* operētājsistēmu, jo tiek

nepārtraukti izsaukta ierīces restartēšana. „*No iOS Zone*" uzbrukums ir nekas cits kā *DoS* uzbrukums dažādām mājaslapām un serveriem, kura dēļ šie pakalpojumi kļūst nepieejami.

Šobrīd vienīgais veids, kā izvairīties no šī uzbrukuma, ir pamest zonu, kurā darbojās ļaunprātīgais *Wi-Fi* tīklājs. Vēl viens veids kā izvairīties no uzbrukumu ir neizmantot bezmaksas bezvada tīklus, kur ir nepazīstami un brīvi pieejami.

Uzbrukuma darbības princips ir tāds, ka vienīgais, kas uzbrucējam ir nepieciešams, ir izveidot ļaunprātīgu bezvadu tīklu, kas izmanto *Wi-Fi* pieslēgumu, lai manipulētu ar *SSL* sertifikātiem, kuri tiek nosūtīti ierīcēm ar *iOS* operētājsistēmu. Tiklīdz ierīce ir pieslēgta ļaunprātīgajam tīklam, uzbrucējs var palaist ļaunprātīgu kodu, kurš veic *DoS* uzbrukumu, kura dēļ gan lietotnes, gan pati ierīce var pārstāt savu darbu. Drošības pētnieki izveidoja video, kurā tiek demonstrēts *DoS* uzbrukumus *iOS* ierīcei [12].

Apple tika informēts par šo ievainojamību, taču nav skaidrs, vai tā uz doto brīdi ir izlabota vai nē. Līdz ar to A. Šarbani un J. Amits ir izlēmuši neizpaust nekādas papildus tehniskas detaļas sakarā ar šo ievainojamību, lai nodrošinātu, ka *iOS* sistēmas lietotāji nav brīvi pieejami problēmām, kas varētu būt saistīties ar „*No iOS Zone*" uzbrukumu [11].

2.2.2 "Masque Attack" ievainojamība

2014. gada jūlijā, mobilās drošības kompānijas „*FireEye*" pētnieki atklāja, ka *iOS* lietotne, kura tiek uzinstalēta, izmantojot uzņēmuma/neformālu (*ad-hoc*) nodrošinājuma profilu (*provisioning profile*), var aizvietot citas „īstās" lietotnes, kuras tika instalētas no *AppStore*, ja abām lietotnēm sakrīt pakotnes identifikators *(bundle identifier*) [13]. Neformālā lietotne var attēlot neobligātu paziņojuma logu, piemēram, „Jauna draugiem.lv lietotne", kas vilina lietotāju, lai to instalētu, bet īstenībā lietotne aizvieto oriģinālo lietotni pēc veiksmīgas instalēšanas. Tādā veidā ir iespējams aizvietot visas lietotnes, izņemot jau instalētās pēc noklusējuma lietotnes, tas ir, lietotnes, kuras nāk kopā ar jaunu *iOS* ierīci, piemēram, lietotne „*Safari*". Ievainojamība rodas tāpēc, ka *iOS* neveic pārbaudi, vai sertifikāti sakrīt lietotnēm ar vienādiem pakotnes identifikatoriem. Kompānija identificēja ievainojamību ierīcēm ar *iOS* versijām 7.1.1, 7.1.2, 8.0, 8.1 un 8.1.1 *beta* gan uz ierīcēm, kam ir uzstādīts *jaibreak,* gan uz ierīcēm, kuram nav. Uzbrucējs var izmantot šo ievainojamību, lietojot gan bezvada tīklu, gan *USB*. Uzbrukums tika nosaukts par „*Masque Attack*", tā demonstrācija ir pieejama tiešsaistē [14].

Uzbrukums "*Masque Attack*" ir ļoti bīstams – tas var aizvietot tādas „īstās" lietotnes kā banku lietotnes, e-pasta lietotnes, izmantojot uzbrucēja ļaunprogrammatūru caur internetu. Tas nozīmē, ka uzbrucējs var iegūt lietotāja bankas lietotājvārdu, paroli. Interesanti, ka „viltus" lietotnei ir piekļuve oriģinālās lietotnes lokāliem datiem, kuri netika dzēsti, kad tika uzinstalēta neīstā lietotne. Lokālie dati var saturēt saglabātās e-pasta vēstules, lietotājvārdu, paroles un citu sensitīvu informāciju.

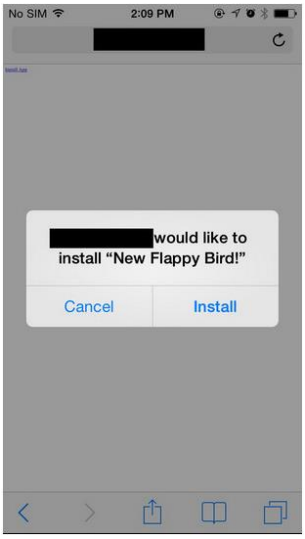
Ievainojamībai var būt šādas sekas:

- uzbrucēji var imitēt orģinālās lietotnes ielogošanas logu, tādējādi iegūstot lietotājvārdu un paroli, un augšupielādēt to uz savu serveri;
- oriģinālās lietotnes lokālie dati tiek saglabāti, kad pa virsu tiek uzinstalēta „viltus" lietotne, līdz ar to sensitīvi dati var tikt augšupielādēti uz uzbrucēja serveri;
- uzbrucējs ar „viltus" lietotnes palīdzību, var „tikt garām" lietotnes ierobežotai videi (*sandbox*) un tad mēģina iegūt sev *root* tiesības, izmantojot zināmās *iOS* ievainojamības.

Lietotnes, kuras izmanto neformālo nodrošinājuma profilu, netiek pārbaudītas ar *Apple* apskati. Līdz ar to uzbrucējs var izmantot privātos *API*, lai īstenotu spēcīgus uzbrukumus, piemēram, fona novērošana, vai atdarināt lietotnes „*iCloud*" lietotāja saskarni, lai nozagtu lietotāja *Apple ID* un paroli [15].

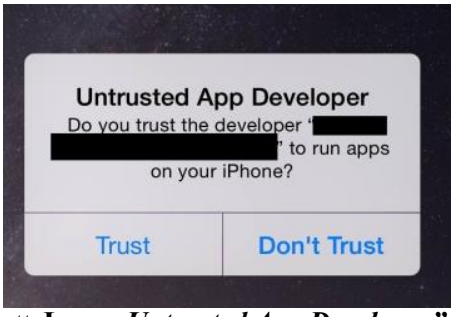
iOS ierīču lietotāji var pasargāt sevi no „*Masque Attack*" uzbrukuma, ievērojot šādus soļus:

- instalēt trešo pušu lietotnes tikai no *AppStore* vai izmantot lietotāja uzņēmuma resursus;
- nespiest pogu „*Install*" uznirstošajā logā, kurš parādās, apmeklējot trešo pušu vietni, kā tas ir parādīts attēlā 2.1.



2.1 att. **Uznirstošais logs „*Install*" [13]**

- ja, atverot lietotni, parādās logs par neuzticamu lietotnes izstrādātāju (*Untrusted App Developer*), kā tas ir parādīts attēlā 2.2, jānospiež poga ar uzrakstu „*Don't Trust*" un jāizdzēš lietotne (ja, protams, tā nav paša lietotāja izstrādātā lietotne).



2.2 att. **Logs „*Untrusted App Developer*" [13]**

2.3 Ievainojamība bibliotēkā "*AFNetworking*"

Daudzi *iOS* izstrādātāji izmanto trešo pušu izstrādās bibliotēkas, lai atvieglotu savu darbu. Taču ir jāapzinās, ka pastāv iespēja, ka bibliotēkā ir kļūda, kuras dēļ var tikt apdraudēti lietotāju dati. Tā tas notika ar bibliotēku „*AFNetworking*".

Bibliotēka „*AFNetwotking*" ir tīkla bibliotēka, kas ir paredzēta operētājsistēmām *iOS* un *Mac OS X*. Šo bibliotēku izmanto tādas lietotnes, kā „*Zagat*" un „*AdWords Express*", kuras ir izstrādājusi kompānija „*Google*", „*Microsoft Health*", kuru izstrādājusi kompānija „*Microsoft*" u.c. Tika atklāts, ka bibliotēkas 2.5.1. versijā ir kļūda, kuras dēļ tiek palaista garām *HTTPS* sertifikāta verificēšana. Attēlā 2.3 ir iezīmēta koda daļa, kuras dēļ rodas ievainojamība.

```
245 - (BOOL)evaluateServerTrust:(SecTrustRef)serverTrust
246                   forDomain:(NSString *)domain
247 {
248     NSMutableArray *policies = [NSMutableArray array];
249     if (self.validatesDomainName) {
250         [policies addObject:(__bridge_transfer id)SecPolicyCreateSSL(true, (__bridge CFStringRef)domain)];
251     } else {
252         [policies addObject:(__bridge_transfer id)SecPolicyCreateBasicX509()];
253     }
254
255     SecTrustSetPolicies(serverTrust, (__bridge CFArrayRef)policies);
256
257     if (self.SSLPinningMode != AFSSLPinningModeNone && !AFServerTrustIsValid(serverTrust) && !self.allowInvalidCertificates) {
258         return NO;
259     }
260
261     NSArray *serverCertificates = AFCertificateTrustChainForServerTrust(serverTrust);
262     switch (self.SSLPinningMode) {
263         case AFSSLPinningModeNone:
264             return YES;
265         case AFSSLPinningModeCertificate: {
266             NSMutableArray *pinnedCertificates = [NSMutableArray array];
```

2.3 att. **Bibliotēkas kļūdas vieta [16]**

Apmēram 1500 *iPhone* un *iPad* lietotnes satur *HTTPS* ievainojamību, kas ļauj uzbrucējiem viegli pārtvert šifrētas paroles, banku kontu numurus un citu sensitīvu informāciju [16]. Lai gan atbildīgie par bibliotēku ir novērsuši ievainojamību, izlaižot versiju 2.5.2., vismaz 1500 *iOS* lietotnes joprojām paliek ievainojamas, jo tās izmanto versiju 2.5.1.

„Problēma rodas, kad mobilā lietotne pieprasa bibliotēkai veikt pārbaudi servera validācijai, izmantojot *SSL* sertifikātu," sava tīmekļa dienasgrāmatā raksta pētnieki Simone Bovi (*Simone Bovi*)

un Mauro Džentile (*Mauto Gentile*) [17]. Viņi turpina: „Mēs notestējām lietotni uz reālas ierīces un konstatējām, ka viss *SSL* trafiks var tikt regulāri pārtverts, izmantojot starpniekservera programmu, piemēram, „*Burp*”.”

Lai īstenotu ievainojamību, uzbrucējam ir nepieciešams atrasties kāda vietā ar publisku, brīvu pieejamu bezvada tīklu, kur varētu atrasties kāda ierīce ar ievainojamu lietotni. Tad ir nepieciešams izveidot viltus *SSL* sertifikātu un nomainīt to pret īsto ievainojamai lietotnei. Parasti akreditācijas dati tiktu identificēti kā viltus un pieslēgums tiktu atteikts. Bet loģiskās koda kļūdas dēļ versijā 2.5.1. validācijas pārbaude netiek īstenota un viltus sertifikāts tiek apstiprināts kā īsts.

Kompānija „*SourceDNA*” identificēja ievainojamās lietotnes, skenējot 5000 populārākās bezmaksas lietotnes, kuras ir pieejamas lejupielādēšanai no *AppStore,* un analizējot bināro kodu katrai lietotnei. Tas ļāva kompānijas darbiniekiem identificēt ievainotās lietotnes, balstoties uz to darbības principiem un kādas bibliotēkas tās izmanto. 1500 ievainotās lietotnes ir tādas, kuras izmanto „*AFNetworking*” bibliotēkas, implementē *HTTPS* un neizmanto tādu iespēju kā sertifikāta piespraušana (*certificate pinning*), kas nodrošina, ka lietotnes izmanto tikai noteiktu sertifikātu *HTTPS* autentificēšanai un šifrēšanai. Pēc noklusējuma sertifikāta piespraušana ir atslēgta bibliotēkā [18].

Lai pārbaudītu, vai lietotne ir ievainojuma bibliotēkas kļūdas dēļ, ir jāapmeklē kompānijas „*SourceDNA*” vietne un jāveic lietotnes meklēšana [19].

Jauna „*AFNetworking*” bibliotēkas ievainojamība tika atrasta visās 2.*x*.*x* versijās, kuras tikai izlaistas pirms 2.5.3. Ievainojamības dēļ uzbrucēji var iegūt, pārraudzīt vai pat modificēt informāciju, pat ja tā ir aizsargāta ar *SSL* sertifikātu. Ievainojamību var izmantot, lietojot jebkuru korektu *SSL* sertifikātu jebkuram domēna vārdam, ja digitālas akreditēšanas datus izdeva pārlūka uzticamas sertificēšanas iestādes (*CA – Certificate Autority*).

„Rezultāts ir tāds, ka, ja uzbrucējam ir korekts sertifikāts, tad viņš var slepus noklausīties (*eavesdrop*) vai modificēt *SSL* sesiju, kuru ir izveidojusi lietotne, kurai ir šī ievainotā bibliotēka,” stāsta Neits Lausons (*Nate Lawson*), drošības analītikas kompānijas „*SourceDNA*” dibinātājs. Viņš turpina: „Problēma ir sekojoša – domēna vārds netiek pārbaudīts sertifikātā, kaut gan pats sertifikāts tiek pārbaudīts, vai to ir izdevusi validēta sertificēšanas iestāde. Piemēram, es varu izlikties, ka esmu „*microsoft.com*”, izmantojot korektu sertifikātu „*sourcedna.com*” domēnam.” [20]

Ievainojamība rodas bibliotēkā „*AFNetworking*”, jo netiek veikta pārbaude, vai domēna vārds, kuru satur sertifikāts, sakrīt ar *HTTPS* servera domēna vārdu, kuru tas sargā. Līdz ar to, ikviens, kas atrodas *Man-in-the-Middle* uzbrukuma pozīcijā, piemēram, uzbrucējs nedrošajā *Wi-Fi* tīklājā, neapmierināts vai viltus darbinieks (*rogue employee*), kurš izmanto iekšējo tīklu vai virtuālo privāto tīklu, var iesniegt savu sertificētās iestādes izsniegto sertifikātu un tad lasīt vai

modificēt „drošo" datu pārraidi.

Ir jāatceras viena lieta – kad tiek iesniegta *iOS* lietotnes jaunā versija, tās apskate var aizņemt līdz 5 darba dienām. Līdz ar to, pat, ja izstrādātāji ir lejupielādējuši jauno „*AFNetworking*" bibliotēkas versiju, tas ir, versiju 2.5.3, var paiet 5 darba dienas, kamēr jaunā lietotnes versija parādīsies *AppStore*. Vajag piebilst, ka kļūda bibliotēkā padara ievainojamu tikai pašu lietotni, kurā tā tiek izmantota. *iOS* ierīces un citas lietotnes, interneta pārlūkus ieskaitot, nav ietekmētas ar bibliotēkas ievainojamību, un tās var droši izmantot.

Apple pārbauda visas lietotnes, kuras tiek iesniegtas *AppStore*. Var secināt, ka daļu vainas, ka tik daudz lietotnes var būt ievainotas, ir jāuzņemas arī *Apple.* Iespējams, ka uz doto brīdi kompānijas testa metodes nav tik ekstensīvas, lai atrastu šāda tipa kļūdas.

Iespējams, ir vēl viens risinājums, kā varētu minimizēt šādas ievainojamības. Risinājums ir anti-vīrusu lietojumprogrammatūra, bet *Apple* aizliedz visas lietotnes, kas pat minimālā mērā imitē anti-vīrusu programmatūru funkcionalitāti.

2.4 Ievainojamības *iOS* lietotnēs

„*WiFi Drive Pro v1.2 iOS*" – *iOS* lietotne ļauj izmantot *iPhone, iPad* vai *iPod touch* ierīci kā bezvadu *USB* atmiņu, ar kuru ir iespējams lejupielādēt, saglabāt un apskatīt dokumentus un failus. Izmantojot šo lietotni, ir iespējams pārnest failus no datora ar bezvada tīkla vai ar *USB* porta palīdzību, to var lejupielādēt no *AppStore.* Dotajai lietotnei ir atklāta ievainojamība, kas ir saistīta ar lokāla faila pievienošanu tīmeklim. Šī ievainojamība ļauj attālinātam uzbrucējam iekļaut pieprasījumus vai sistēmai specifiskas komandas, lai kompromitētu tīmekļa lietotni. Ievainojamība atrodas vērtībā „*filename*" modulī „*file upload*". Uzbrucēji var pievienot savus failus ar ļaunprātīgiem „*filename*" vērtībām „*file upload*" *POST* metodes pieprasījumā, lai kompromitētu mobilo tīmekļa lietotni. Drošības risks dotajai ievainojamībai ir novērtēts kā augsts ar *CVSS* vērtējumu 6.4. Ievainojamības izmantošanā nav vajadzīgas lietotāja darbības vai priviliģētas tīmekļa lietotāja konts. Rezultātā tiek kompromitēta mobilā lietotne [21].

„*Photo Manager Pro 4.4.0 iOS*" – *iOS* lietotne palīdz pārvaldīt fotogrāfijas un video. Ar tās palīdzību ir iespējams pārsūtīt fotogrāfijas un video, izmantojot bezvada tīklu, *USB* pieslēgumu, programmas „*iTunes"* failu koplietošanu, lietotni „*Photos*". Ir iespējams failus apskatīt arī pārlūkā. Lietotne ir pieejama *AppStore*. Dotajai lietotnei tika atklāt patvaļīga koda izpildes ievainojamība, kas ļauj uzbrucējiem izpildīt ļaunprātīgu kodu lietotnes pusē un kompromitēt mobilo ierīci. Ievainojamība atrodas modulī „*newfolder.action*" mainīgajā „*folderName*". Attālinātie uzbrucēji var manipulēt ar vērtību *POST* metodes pieprasījumā „*index.html#?w=300*", lai kompromitētu lietotni, lietotāja sesijas informāciju vai pašu telefonu. Drošības risks dotajai ievainojamībai ir novērtēts kā augsts ar *CVSS* vērtējumu 8.6. Ievainojamības izmantošanai nav vajadzīgas lietotāja darbības vai privileģēts tīmekļa lietotnes lietotāja konts. Veiksmīgas ievainojamības izmantošanas

rezultātā var notikt sesijas nolaupīšana (*session hijacking*), pastāvīgā pikšķerēšana (*phishing*) u.c. [22]

„*Folder Plus 2.5.1 iOS*" – *iOS* lietotne ļauj viegli un ērti pārvaldīt, apskatīt un modificēt failus. Lietotne ir pieejama *AppStore.* Dotajai lietotnei tika atklāta tīmekļa ievades pārbaudes ievainojamība. Kļūda ļauj uzbrucējam injicēt savu skripta kodu kā vērtumu (*payload*) lietotnes pusē ievainotajā modulī. Ievainojamība atrodas *Wi-Fi* saskarnes attēlošanas modulī izdzēsto ziņojumu kontekstā. Uzbrucējs injicē skripta koda vērtumu un gaida, kad lietotājs ar augstākām tiesībām vai serviss izdzēš vienību, kā rezultātā tiek palaists skripts. Drošības risks dotajai ievainojamībai ir novērtēts kā vidējs ar *CVSS* vērtējumu 2.5. Ievainojamības izmantošanai ir nepieciešams tīmekļa lietotnes lietotāja konts ar zemām tiesībām un neliela lietotāja mijiedarbība. Veiksmīgas ievainojamības izmantošanas rezultātā var notikt sesijas nolaupīšana, pastāvīgā pikšķerēšana, pastāvīga novirzīšana uz ļaunprātīgām tīmekļa lapām u.c. [23]

„*Grindr 2.1. iOS*" – lielākā un populārākā uz atrašanās vietas balstītā sociālā *iOS* lietotne vīriešiem [24]. Ar tās palīdzību var atrast draugu, paziņu. Lietotne ir pieejama *AppStore*. Lokāla un attālinātā pakalpojumatteices ievainojamība ir atrasta versijā 2.1.1 *iOS* mobilajā tīmekļa lietotnē. Lietotājs injicē skripta koda tagu vai vairākas izbeigšanas burtu virknes (*termination strings*) – *%00%20%00%20%00* – lietotāja vārda ievades laukā modulī „*Edit Profile*". Kad ļaunprātīgā burtu virkne tiek saglabāta, kāds nejaušs lietotājs uzspiež uz „ļaunprātīgo" kontakta profilu, lai apskatītu informāciju sociālos tīklos *„Twitter*", „*Facebook*". Ja lietotājs vēlās nokopēt vietnes adresi, notiek mobilās lietotnes darbības beigšana. Drošības risks dotajai ievainojamībai ir novērtēts kā vidējs ar *CVSS* vērtējumu 3.3. Lai izmantotu doto ievainojamību, ir nepieciešama neliela lietotāja mijiedarbība [25].

„*PayPal Inc iOS 4.6*" – *iOS* lietotne, kas piedāvā veikt maksājumus tīmeklī dažādiem pakalpojumiem dažādās tiešsaistēs, piemēram, http://www.ebay.com, un naudas pārskaitīšanu. Lietotne ir pieejama *AppStore*. Lietotnei un tās *API* sastāvdaļai ir atklāta autentifikācijas ierobežojuma apiešanas ievainojamība . Tā ļauj apiet filtru ierobežojumus tīmekļa pakalpojumam, lai iegūtu neautorizētu pakalpojuma „*PayPal*" konta pieeju. Ievainojamība atrodas „*PayPal*" tīmekļa pakalpojuma mobilajā *API* autentifikācijas procedūrā. Mobilās lietotnes *API* neveic pārbaudi jau ierobežotajiem vai bloķētiem lietotnes kontiem. Attālinātie uzbrucēji vair pieslēgties, izmantojot mobilās lietotnes *API,* un iegūt pieeju konta informāciju vai darboties ar to pašu kontu. Drošības risks dotajai ievainojamībai ir novērtēts kā augsts, ar *CVSS* vērtējumu 6.2. Lai izmantotu doto ievainojamību, ir nepieciešams ierobežots vai bloķēts „*Paypal*" lietotāja konts un nav vajadzīga nekāda lietotāja mijiedarbība. Veiksmīga ievainojamības izmantošanas rezultātā uzbrucējs var apiet autentifikācijas ierobežojumu dotajai lietotnei, izmantojot lietotnes *API* [26].

3. LIETOTŅU DROŠĪBAS PĀRBAUDE

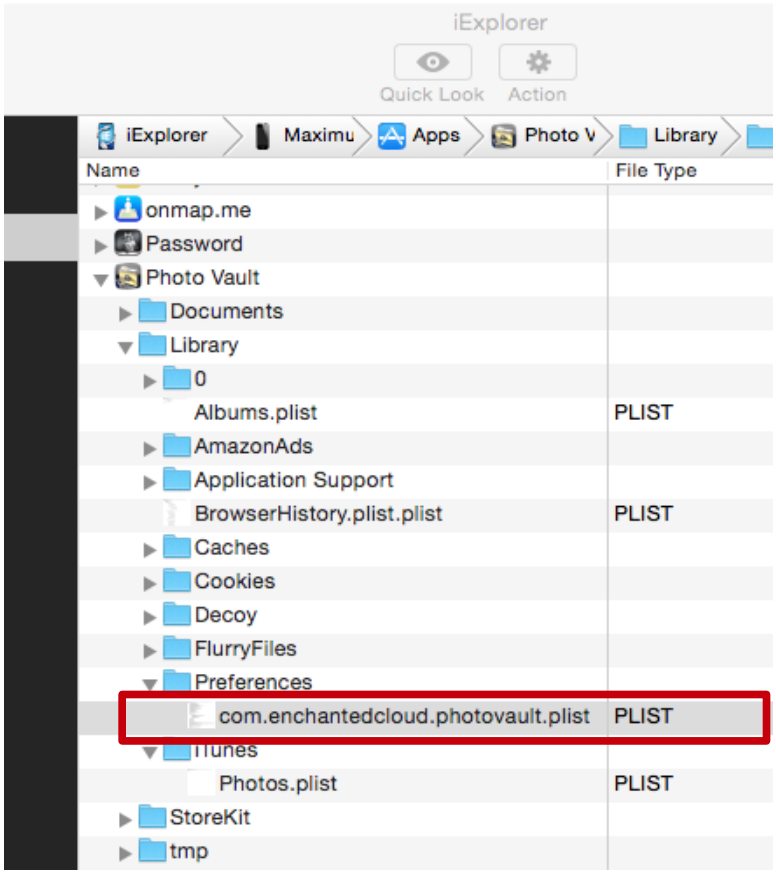
Maģistra darba ietvaros tika atrastas trīs lietotnes, kuras, pēc autora domām, nerealizē lietotāja datu drošību. Šīs lietotnes ir *"Photo Vault", "My Apps Lite", "WhatsApp"*. Datu drošība tika pārbaudīta ar vienkāršu programmu *"iExplorer",* kas attēlo katras lietotnes failu sistēmā izveidotos failus. Veicot drošības pārbaudi, autors neizmantoja *jailbreak* ierīci. Tas nozīmē, ko šāda veida pārbaudi var veikt jebkurai *iOS* ierīcei.

3.1 Lietotnes *"Photo Vault"* drošības pārbaude

Lietotne *"Photo Vault"* ļauj lietotājam droši uzglabāt fotogrāfijas. Lietotne piedāvā uzstādīt kodu, kas tiek pieprasīts, lai apskatītu fotogrāfijas. Attēlā 3.1 parādīts, kā izskatās koda ievades ekrāns.

3.1. att. **Koda ievades ekrāns**

Attēlā 3.2 var redzēt lietotnes failu sistēmu. Uzmanība ir jāpievērš izceltajam failam, kurš, spriežot pēc mapes nosaukuma, satur lietotnes konfigurācijas datus.



3.2. att. **Lietotnes *"Photo Vault"* failu sistēma**

Atverot „*com.enchantedcloud.photovault.plist*" failu ar jebkuru teksta redaktoru, var redzēt dažādus konfigurācijas datus. Attēlā 3.3 skaidri redzams kods, kas tiek lietots, lai atbloķētu lietotni. Kā redzams, tas ir saglabāts vienkāršā tekstā.

```
<key>OpenUDID</key>
<dict>
        <key>OpenUDID</key>
        <string>484a3bfc964debd7da6f69078868bf8807dd551e</string>
        <key>OpenUDID_appUID</key>
        <string>2614DE7F-B261-4CF2-8843-B149295BE014</string>
        <key>OpenUDID_createdTS</key>
        <date>2015-01-27T18:03:25Z</date>
        <key>OpenUDID_slot</key>
        <string>org.OpenUDID.slot.0</string>
</dict>
<key>PIN</key>
<string>1234</string>
<key>PathNaming</key>
<integer>1</integer>
<key>Request Pin</key>
<false/>
<key>RequirePin</key>
<true/>
```

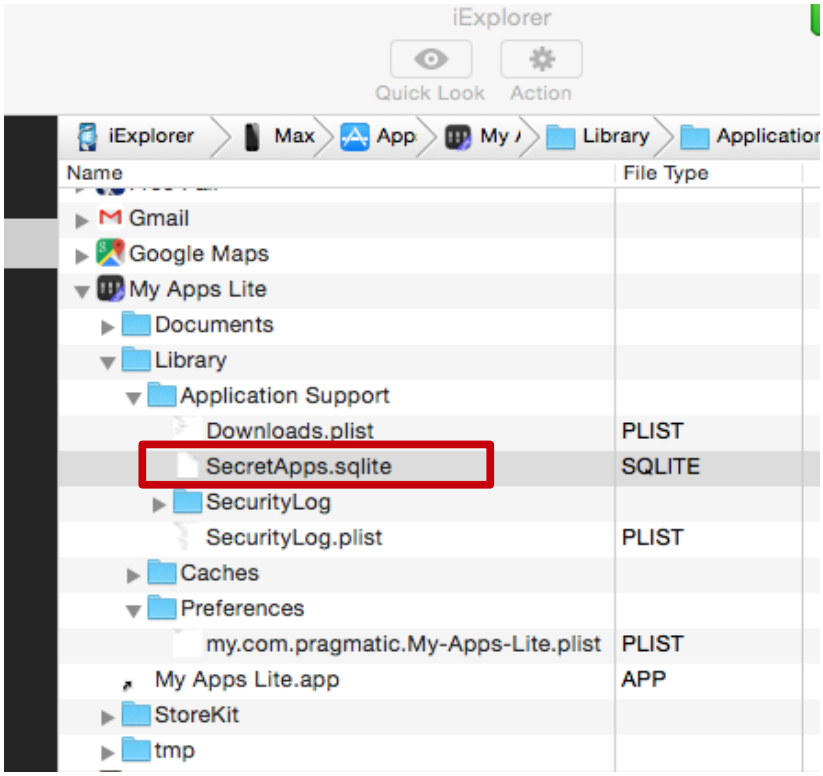
3.3. att. **Lietotnes konfigurācijas fails**

3.2 Lietotnes *"My Apps Lite"* drošības pārbaude

Lietotne *"My Apps Lite"* piedāvā veikt dažādus pierakstus, saglabāt fotogrāfijas, pievienot kontaktus. Šie dati tiek pasargāti, uzstādot lietotnei kodu, kurš jāievada katru reizi, kad lietotne kļūst aktīva. Attēlā 3.4 ir redzams, ka ir pievienots viens ieraksts.

3.4. att. **Pievienotais ieraksts lietotnē**

Izmantojot programmu "*iExplorer*", ir iespējams atrast lietotnes failu sistēmā datubāzes failu, kas ir redzams attēlā 3.5.



3.5. att. **Lietotnes *"My Apps Lite"* failu sistēma**

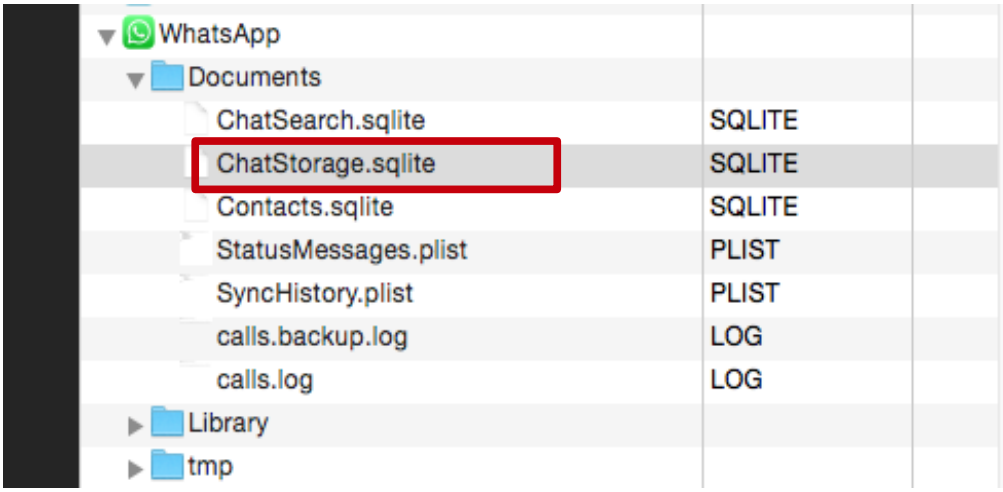
Izmantojot programmu *"Liya"*, datubāzes fails tiek atvērts, un tajā var redzēt ierakstu, kas atbilst veiktajam lietotnē *"My Apps Lite"* – attēls 3.6.

3.6. att. **Lietotnes datubāzes saturs**

Līdz ar to, nav pat nepieciešams atbloķēt lietotni, lai piekļūtu tās "drošībā" glabātiem datiem.

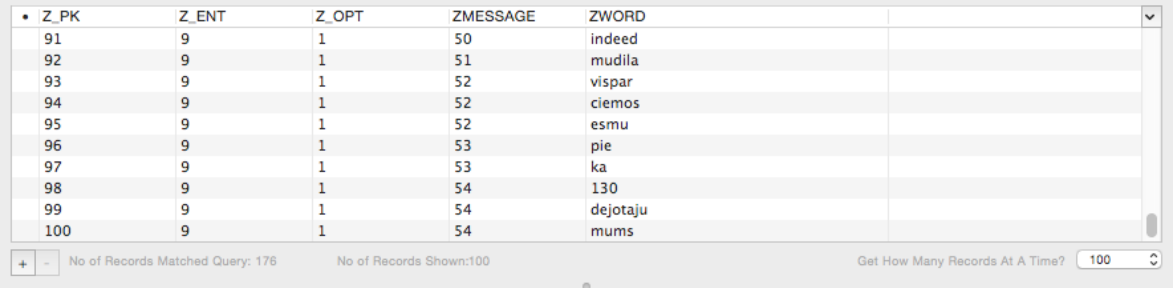
3.3 Lietotnes "*WhatsApp*" drošības pārbaude

Lietotne „*WhatsApp*" ļauj lietotājiem ērti sazināties, izmantojot bezvadu tīklu. Attēlā 3.7 var redzēt populāras sarakstes lietotnes *"WhatsApp"* failu sistēma.



3.7. att. **"*WhatsApp*" failu sistēma**

Kā var redzēt attēlā, failu sistēmā ir vairāki datubāzu faili, pēc nosaukumiem var aptuveni pateikt, kādus datus katra datubāze uzglabā. Attēlā 3.8 var redzēt faila *"ChatStorage.sqlite"* saturu.



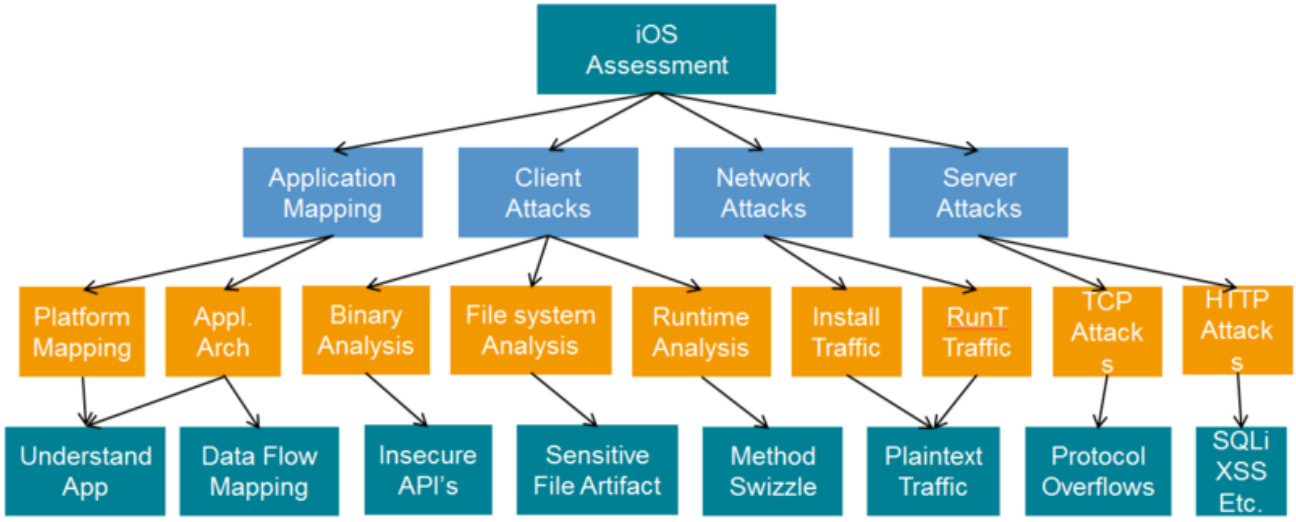
3.8. att. **"*WhatsApp*"** datubāzes ieraksti

3.8 attēlā ir redzams sarakstes teksts.

4. IELAUŠANĀS TESTĒŠANA

4.1 Ielaušanās testēšanas komponentes

Ielaušanās testēšana (*penetration testing*) ir testēšanas veids, kurā lietotne, sistēma, tīkls tiek testēts no uzbrucēja puses, tas ir, tiek simulēts uzbrukšanas process. *iOS* ielaušanas testēšanas laikā uzbrukšanas vektori jeb pieejas ir redzamas attēlā 4.1.



4.1 att. **Drošības pārbaudes shēma [27]**

Kā var redzēt 4.1 attēlā, *iOS* lietotnes drošības pārbaudē izdala četras lielas komponentes, kuras ir jāpārbauda. Tās ir:

- lietotnes kartēšana (*mapping*);
- uzbrukumi klientam (*iOS* lietotnei);
- uzbrukumi tīklam;

Lietotnes kartēšanas komponentes testēšanas laikā ir jāmēģina izprast, kā strādā lietotne, kādus datus tā izmanto. Komponente tiek sadalīta divās daļās:

- platformas kartēšana;
- lietotnes arhitektūra.

Platformas kartēšanas daļā ir jāizprot lietotnes darbības princips, kādi ekrāni ir lietotnē, kādas komponentes tiek izmantotas. Lietotnes arhitektūras daļā ir jāmēģina izprast, kā dati tiek pārvietoti lietotnes darbības procesā, ja ir iespējams, jāuzzīmē datu plūsmas diagrammu, lai izprastu lietotnes darbības principus.

Tālāk tiek veikta ielaušanās testēšana lietotnei. Uzdevums ir atrast ievainojamības, kas varētu būt saistītas ar bibliotēkām, sensitīvo datu nekorektu glabāšanu u.c. Fāze „Uzbrukumi lietotnei" tiek iedalīta trijās daļās:

- binārā analīze;

- failu sistēmas analīze;
- lietotnes izpildlaika analīze.

Binārās analīzes daļā tiek pārbaudītas bibliotēkas, kādas lietotne izmanto, tiek apskatīti funkciju nosaukumi – pēc tiem var dažreiz viegli izprast, kāds ir funkcijas uzdevums. Failu sistēmas analīzes daļā jāpārbauda, kādus failus lietotne izveido, vai pēc failu nosaukumiem var izprast, kas tajos tiek glabāts, kāda informācija tiek glabāta, vai faili ir šifrēti vai nē. Izpildlaika analīzes daļā var pārbaudīt, kādas metodes tiek izsauktas kādā konkrētā brīdī, piemēram, kad lietotājs ir nospiedis kādu pogu lietotnē. Šajā daļa ir iespējams veikt funkciju apmaiņu (*method swizzle*), kuras laikā īstā metode tiek aizstāta ar uzbrucēja definēto metodi.

Ir jāveic uzbrukumi tīklam, ja tas ir nepieciešams lietotnes darbībai. Uzdevums ir mēģināt apskatīties, kādi dati tiek sūtīti, cik bieži, kādā veidā tie ir aizsargāti. Uzbrukumi tīklam sastāv no divām daļām:

- lietotnes instalācijas datplūsma;
- lietotnes darbības datplūsma.

Gan lietotnes instalācijas, gan darbības laikā ir jāmēģina pārtvert datplūsma, jāanalizē tā, jāpārbauda, vai dati tiek pārsūtīti šifrētā vai nešifrētā veidā. Var mēģināt veikt sertifikāta pārtveršanu, *Man-in-the-Middle* uzbrukumu.

Ja ir iespēja, tad ir jātestē serveris, ar kuru lietotnei notiek komunikācija. Uzdevums ir gūt pieeju serverim, mēģināt izgūt jebkādu informāciju, iegūt piekļuvi datubāzei, kas atrodas uz servera. Uzbrukumi serverim sastāv no divām daļām:

- *TCP* uzbrukumi;
- *HTTP* uzbrukumi.

TCP uzbrukumos ir jāmēģina pārpludināt dažādi protokoli ar mērķi izraisīt *DoS* uzbrukumu. *HTTP* uzbrukumos var izmantot *SQL* injekcijas (*injections*), lai iegūtu datus, kas glabājas datubāzē, starpvietņu skriptptēšanas (*XSS – cross-site-scripting*) uzbrukumus, lai mēģinātu iegūt informāciju par sesijas sīkfailiem, vairāku vietņu pieprasījuma viltošanas (*CSRF – cross-site request forgery*) uzbrukumus, lai viltotu „īstā" lietotāja pieprasījumus.

4.2 Ielaušanās testēšanas gaita

Darba autors *iOS* ielaušanās testēšanu ir iedalījis četros soļos:

- plānošana – šajā solī jāsagatavo ierīces un programmatūra, lai varētu veiksmīgi sākt *iOS* lietotnes drošības pārbaudi;
- informācijas ievākšana – šajā solī ir jāievāc informācija par lietotni, jāapskatās lietotnes tvērums;
- statiskā analīze – šajā solī tiek analizēts mobilās lietotnes pirmkods, ja tas nav pieejams,

tad ir jāveic koda dekompilēšana vai disasamblēšana

- dinamiskā analīze – šajā solī ir jādarbina lietotne un jāveic tās darbības analīze reālajā laikā.

4.2.1 Plānošanas fāze

Šajā fāzē ir nepieciešams sagatavot vajadzīgo aparatūru, programmatūru, lai varētu uzsākt ielaušanās testēšanu. Testētājam ir jābūt sagatavotai *iOS* ierīcei – *iPad, iPhone, iPod touch*. Vēlams, lai ir vairākas ierīces ar dažādām operētājsistēmas versijām, jo ir lietotnes, kuras ir pieejams tikai, piemēram, *iOS* 8 versijām, vai arī lietotne var atšķirties katrai *iOS* versijai. Katrai ierīcei ir jābūt uzstādītam *jailbreak*, kas ļauj veiksmīgāk veikt lietotnes drošības vērtēšanu. Testētājām ir jāuzinstalē nepieciešama programmatūra gan uz *iOS* ierīcēm, gan uz datora – jāsagatavo ielaušanās testēšanas programmatūras nodrošinājums.

Kad visi plānošanas un sagatavošanas darbi ir pabeigti, var pāriet pie nākamās fāzes – informācijas ievākšana par lietotni.

4.2.2 Informācijas ievākšanas fāze

Informācijas ievākšanas fāze ir ļoti svarīga. Tā palīdz testētājam izprast lietotnes darbības principus. Ir svarīgi izpētīt, kā lietotne strādā, kādas tehnoloģijas tiek izmantotas. Ja testētājs neizpētīs, kā lietotne strādā, tad nevarēs identificēt momentus, kad lietotne strādā „nepareizi", kas ir svarīgs moments ielaušanās testēšanas procesā. Jo vairāk testētājs ievāks informācijas par lietotni un tās darbības principiem, jo vieglāk būs viņam strādāt nākamajās fāzēs.

Šī fāze ir diezgan apjomīga. Informācijas ievākšanas fāzē var veikt sekojošas darbības, lai iegūtu informāciju par lietotni [28]:

- apskatīt datu pārraidi starp lietotni un serveri vai *API*, saglabāt to turpmākai izpētei;
- reģistrēt vairākus testa kontus, lai varētu testēt privilēģiju palielināšanas uzbrukuma vektorus;
- manuāli iziet cauri visai lietotnei, lai izprastu tās pamata darbības principus un darbplūsmu;
- pārbaudīt, kādus tīkla protokolus lietotne izmanto – vai tie ir droši vai nē, un vai ir iespējams pāriet uz mazāk drošiem protokoliem;
- vai lietotnē tiek izmantoti maksāšanas pakalpojumi – darbības ar kredītkartes datiem, iekšējie lietotnes maksājumi (*In-App purchases*);
- izpētīt, kādas ierīces komponentes lietotne izmanto – mikrofonu, kameru, *GPS*, *Bluetooth* u.c.;
- jāmēģina atrast, kādas bibliotēkas vai programmatūras izstrādātāja rīkkopas (*SDK – Software Development Kit*) lietotne izmanto, un tad izpētīt, vai nav reģistrētas kādas ievainojamības šajās bibliotēkās;
- izpētīt, vai lietotne strādā vai izmanto citas lietotnes, piemēram, „*iCloud*", „*Dropbox*", „*Twitter*", „*Facebook*", vai kādus citus pakalpojumus, piemēram, *SMS,* kontaktu grāmatiņu,

e-pastu;

- vai ir iespējams atrast informāciju, kāda ir servera puses vide, ar kuru strādā lietotne, piemēram, viesošanās (*hosting*) pakalpojumu sniedzēju – *„AWS*", „*Heroku*", „*Azure*" u.c., kādā vidē vai valodā ir izstrādāta servera puse – „*Rails*", „*Java*", „*Django*" u.c., vai lietotne izmanto autentifikācijas *API* vai vienoto pierakstīšanos (*single sign on*)
- izpētīt pieprasījumus un atbildes no servera, lai identificētu noderīgus datus, vai lietotnes uzvedību, kad tiek saņemti dati;
- izpētīt dažādus lietotnes stāvokļus – kā tie strādā, kad atrodas fona režīmā, pārējā no priekšplāna (*foreground*) uz fona stāvokli;
- atšifrēt *AppStore* bināros failus;
- izpētīt, kādai arhitektūrai lietotne tika kompilēta;
- iegūt informāciju par funkcijām, klasēm un metodēm, kas tiek izmantotas lietotnē u.c.

Kad visa šī informācija ir iegūta, apskatīta un pierakstīta, testētājs var pāriet pie nākamās ielaušanās testēšanas fāzes – statiskās analīzes.

4.2.3 Statiskās analīzes fāze

Pārsvarā ir divi veidi, kā statiskā analīze var tikt veikta mobilajai lietotnei [28]:

- analizēt pirmkodu, kurš tika iegūts no izstrādātāju komandas (ieteicamais veids);
- uzmantojot sakompilēto bināro failu.

Gadījumos, kad galvenais mērķis ir programmēšanas kļūdas, kuru dēļ rodas drošības draudi, ir vēlams veikt pirmkoda analīzi, nevis reverso inženieriju lietotnei. Lai veiktu pirmkoda apskati, ir vēlams, lai testētājam būtu piekļuve vai nu testēšanas, vai produkcijas tīmekļa servisa instancei.

Statisko analīzi var sadalīt vairākos posmos [28]:

- darba sākšana;
- autentifikācijas testēšana;
- autorizācijas testēšana;
- sesiju pārvaldes testēšana;
- datu glabātuves testēšana;
- transporta līmeņa drošības testēšana;
- informācijas drošības testēšana;
- tīmekļa lietotņu drošības testēšana.

Darba sākšanas posmā, ja lietotnes pirmkods nav pieejams, ir jāveic lietotnes dekompilēšana. Tālāk ir jāapskata, kādas atļaujas lietotne pieprasa, piemēram, mikrofona vai kameras lietošana. Ir jāapskata, kādas bibliotēkas lietotne izmanto un vai tās ir pašas jaunākās versijas bibliotēkas, vai tām ir atrastas ievainojamības. Testētājam ir jāmēģina atrast statiski definētos tekstus, kas varētu kalpot kā lietotājvārdi vai paroles serverim, jānosaka visi datu ieejas

punkti, kuri tiek izmantoti lietotnē, piemēram, datu ielasīšana no faila vai no datubāzes.

Autentifikācijas testēšanas posmā ir jāmēģina atrast kods, kas ir atbildīgs par lietotāja autentifikāciju, izmantojot lietotāja saskarni. Ir jāizvērtē iespējamās metodes, lai attēlotu „īsto" lietotāju, piemēram, parametru mainīšana, atkārtošanas uzbrukumi un pārlases (*brute-force*) uzbrukumi. Testētājam jānosaka, vai lietotne izmanto ne tikai lietotājvārdu un paroli, bet vēl kādu citu informāciju, piemēram, lietotāja atrašanās vietu, sertifikātus, piekļuves pilnvaras (*access tokens*). Galvenais ir izpētīt pēc iespējas dažādas autentifikācijas pieejas un kuras no tām tiek izmantotas lietotnē.

Autorizācijas testēšanas posmā ir jāpārbauda tiesības failiem, kuri tiek izveidoti izpildlaikā, jānosaka, vai ir iespējams piekļūt funkcionalitātei, kas nav paredzēta lietotāja lomai, piemēram, vai lietotnē pieejamā funkcionalitāte ir sadalīta, balstoties uz lietotāja tiesībām, atrast vietas lietotnē, kurām nav paredzēta piekļuve, ja neizpildās kāds konkrēts nosacījums vai darbību plūsma. Ir jāpārbauda licencēšana, tas ir, vai licencēšanas pārbaudes – vai ir samaksāts par papildus pakalpojumiem, ko lietotne sniedz, – var tikt lokāli apietas, vai licencēti pakalpojumi jau ir iestrādāti lietotnē, un tie nav pieejami tikai caur lietotāja saskarni, vai licencēšanas pārbaude notiek servera pusē vai lietotnes pusē, kā lietotne nosaka un rīkojas, ja tiek atklātas, ka ar dati tika modificēti, – vai informācija tiek nosūtīta izstrādātājam, vai lietotne vairs nevar uzsākt darbu, vai lietotnes dati tiek dzēsti.

Sesiju pārvaldīšanas testēšanas posmā ir jāpārbauda, vai sesija laiks tiek kontrolēts lokāli un servera pusē un vai sensitīvā informācija tiek dzēsta no lietotnes atmiņas, kad sesijas laiks ir beidzies.

Datu glabātuves testēšanas posmā ir jāpārbauda šifrēšana – kādi algoritmi tiek izmantoti, lai šifrētu datus, vai tie ir populārākie un labākie, vai arī tiem ir zināmas ievainojamības, kur tiek glabātas šifrēšanas atslēgas. Tālāk testētājam vajadzētu noteikt, kur lietotne glabā datus, piemēram, vai faili glabājas vietās, kurām tiek veikta rezerves kopija, izmantojot programmu „*iTunes*", vai faili tiek sinhronizēti ar mākonī bāzētiem glabāšanas pakalpojumiem, piemēram, „*Dropbox*" vai „*Google Drive*". Vēl ir jāpārbauda, vai lietotne kādā brīdī raksta sensitīvos datus failu sistēmā, piemēram, lietotājvārdus vai paroles, *API* piekļuves datus, maksājuma informāciju.

Transporta līmeņa drošības testēšanas posmā testētājam ir jāpārbauda, vai lietotne veic korektu sertifikāta piesaisti, vai sertifikāti tiek validēti, lai noteiktu, vai sertifikātam nav beidzies derīguma termiņš, vai sertifikātu izsniedza korekta sertifikātu izsniegšanas aģentūra, vai gala mērķis atbilst informācijas, kas ir rakstīta sertifikātā.

Informācijas drošības testēšanas posmā ir jāpārliecinās, vai teksta izvadīšana žurnālfailos ir ieslēgti vai izslēgti, un, ja ir ieslēgti, vai tajos netiek rakstīta sensitīva informācija, un, vai informācija, kas tiek rakstīta, nepārkāpj privātuma tiesības. Ir jāpārbauda kešfails, vai tajā netiek

glabāta atrašanās vietas informācija vai pārlūka informācija, fatālos izņēmumus (*exceptions*) – vai sensitīva informācija netiek izpausta lietotnes darba beigšanas žurnālfailos. Kā arī ir jāveic trešo pušu bibliotēku un *API* pārbaude – kādas tiesības tiek pieprasītas, vai tiek pārraidīta sensitīva informācija, vai to darbība izpildlaikā var pakļaut lietotājus dažādiem drošības riskiem, piemēram, privātuma pārkāpumi vai neatļautā sekošana.

Tīmekļu lietotņu drošības testēšanas posmā ir jātestē, vai lietotnei ir iespējams veikt *HTML* un starpvietņu skritptēšanas injekcijas, komandu injekcijas, veikt vairāku vietņu pieprasījuma viltošanas uzbrukumus, *SQL* injekcijas, uzbrukumus sīkfailiem un *HTML5*.

4.2.4 Dinamiskās analīzes fāze

Dinamiskās analīzes posmā jāveic testēšana lietotnes darbības laikā. Šajā posmā ir nepieciešamas disasamblēt lietotni, lai iegūtu pēc iespējas vairāk informācijas. Dažas no darbībām, ko var veikt dinamiskās analīzes laikā, ir [27]:

- analizēt failu un sistēmas mijiedarbību;
- ar programmu „*class-dump-z*" iegūt informāciju no *.h* failiem (tajos glabājas publisko metožu nosaukumi, galvenes fails) par metodēm, lai varētu mēģināt izveidot metožu apmaiņu;
- izpētīt klases „*CFStream*" un „*NSStream*", ja tādas tiek izmantotas lietotnē;
- veikt buferu/atmiņas pārpildes (*buffer/memory overflow*) un atmiņas bojāšanas (*corruption*) testēšanu;
- veikt izpildlaika injekcijas;
- izsaukt metodes, ievietojot sev vēlamos parametrus, lai apietu ierasto lietotnes darbības plūsmu, piemēram, paslēptu autentifikācijas ekrānu.

Pēc autora domām, ja ielaušanās testēšanas laikā tiek iziets cauri augstāk apskatītiem posmiem, var iegūt pietiekami daudz kvalitatīvas informācijas par to, vai testējamā lietotne ir droša vai tomēr nē.

5. *iOS* LIETOTŅU IELAUŠANĀS TESTĒŠANA

Darba autors ir veicis ielaušanās testēšanu dažām *iOS* lietotnēm, lai pārbaudītu to drošību. Parasti, lai veiktu šāda tipa testēšanu, ir nepieciešama *jailbreak* ierīce.

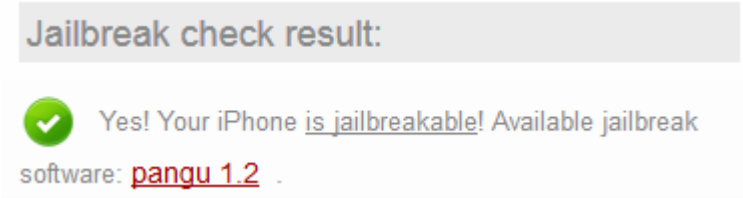
5.1 Ielaušanās testēšanas plānošana

Ielaušanās testēšanas pirmā fāze ir plānošana. Šajā fāzē ir jāsagatavo visu nepieciešamu programmatūru un aparatūru, lai varētu veikt testēšanu. No aparatūras darba autors testēšanā izmanto *"MacBook Pro"* datoru, *iPhone 5* telefonu ar *iOS* versiju 7.1.2.

Pirmais solis ir veikt *jailbrek* procesu ierīcei. Sākumā ir jāpārbauda, vai vispār ierīcei var veikt *jailbreak* – to ir iespējams apskatīties vietnē http://jailbreak-me.info/index.php[1]. Vietnē ir jāievada savas ierīces parametri, kā tas ir redzams attēlā 5.1.

5.1 att. ***Jailbreak*** **iespējas pārbaude**

Uzspiežot pogu ar uzrakstu *"Check your iDevice"*, tiek pārbaudīta ierīce konfigurācija, tas ir, vai ir iespējams tai veikt *jailbreak.* Attēlā 5.2 var redzēt, ka autora izvēlētai ierīcei *jailbreak* ir iespējams.



5.2 att. **Pārbaude rezultāts**

Darba autors sekoja atrastajām instrukcijām, lai uzstādītu ierīcei *jailbreak* [29]. Obligāti ir jāveic datu rezerves kopija ierīcei. Interesants ir tas, ka ierīcei vajag ieslēgt lidmašīnas režīmu, noņemt automātisko laika un datuma iestatīšanu un uzlikt datumu uz 2014. gada 2. jūniju. *Jailbreak* uzstādīšanas procesā telefons dažas reizes restartējas, ir nepieciešams atvērt lietotni

[1] Skatīts 2015. gada 4. maijā

"*Pangu*", kas tika instalēta *jailbreak* procesa sākumā. Pēc veiksmīgas *jailbreak* uzstādīšanas, *iPhone* ierīcei parādās lietotne "*Cydia*". Tagad ir jāinstalē nepieciešamās lietotnes, bibliotēkas un citu programmatūru, lai varētu uzsākt ielaušanās testēšanu.

Lai uzsāktu ielaušanās testēšanu, ir jānodrošina piekļuve ierīcei. Šim nolūkam darba autors izmanto pakalpojumu "*Open SSH* Lietotni ir iespējams lejupielādēt, izmantojot *Cydia* – vajag apakšā jāizvēlas "*Search*" opcija, tad var ievadīt meklējamās komponentes nosaukumu un instalēt to.

Pakalpojums ļauj attālināti pieslēgties ierīcei, izmantojot divus veidus:

- bezvadu tīklu;
- *USB*.

Lai pieslēgtos ierīcei caur bezvada tīklu, programmā „*Terminal*" (*Mac OS X* vide) ir jāievada sekojoša komanda – „*ssh root@ip*", kur *ip* ir ierīces *IP* adrese. Parole ir „*alpine*". Kad tas ir izdarīts ir ļoti ieteicams nomainīt paroli, tas ir izdarāms ar komandu „*passwd root*", tālāk tiks prasīts ievadīt jaunu paroli lietotājam *root*. Aprakstītās darbības ir redzamas attēlā 5.3.

```
Max:~ maximus$ ssh root@192.168.2.2
root@192.168.2.2's password:
oldPhone:~ root#
oldPhone:~ root#
oldPhone:~ root# passwd root
Changing password for root.
New password:
Retype new password:
oldPhone:~ root#
```

5.3 att. **Paroles maiņa**

Lai pieslēgtos telefonam, izmantojot *USB* pieslēgumu, ir jālejupielādē faili un jāpalaiž fails ar nosaukumu "*tcprelay.py*", kas atrodas mapē ar nosaukumu "*usbmyx-python-client*" [30]. Gala komanda izskatās šādi – „*sudo python tcprelay.py -t 22:22*". Kad skripts ir palaists, tad telefonam var pieslēgties, izmantojot jebkuru programmu, kas nodrošina darba ar protokolu *SFTP*. Darba autors izmanto programmu "*CyberDuck*". Pieslēdzoties jānorāda lietotājs *root*, adrese ir „*localhost*" un parole ir ierīces parole, pēc noklusējumu „*alpine*". Ja pieslēgšanās ir veiksmīga, tad ir redzama *iPhone* failu sistēma.

Kad iespēja pieslēgties ierīcei ir notestēta, ir nepieciešams lejupielādēt pārējās lietotnes, programmatūru, pakalpojumus un bibliotēkas, lai varētu veikt ielaušanās testēšanu. Bet pirms tam vēl ir nepieciešams instalēt vienu servisu, lai varētu lietot "*apt-*" komandu uz *iPhone*, kas ļauj lejupielādēt un uzstādīt dažādas programmas un servisus.

Darba autors testēšanā izmantoja vairākas komponentes

1. Programmas, kas tika lejupielādētas no *Cydia*, ir:

- *Erica Utilities* – lejuplādējot šo lietotnei, tiek uzstādīta *plutil* programma;
- *unzip;*
- *adv-cmds;*
- *cycript.*

2. Lietotnes, kas tika lejuplādētas terminālī, izmantojot komandu *apt-get install*, ir:
   - *wget;*
   - *sqlite3;*
   - *vim;*
   - *file;*
   - *python.*
3. Programmas, kas tika lejupielādētas no dažādām vietnēm, ir
   - *class-dump-z* [31];
   - *clutch* [32];
   - *burp suite* [33];
   - *keychain-dumper* [34].

5.2 Ielaušanās testēšanas sākums

Ielaušanās testēšana tiek veikta lietotnēm, pieslēdzoties *iOS* ierīcei caur *SSH* piekļuvi. Kad pieslēgums ierīcei ir nodibināts, nākamais solis ir atrast, kur glabājās lietotnes. *iOS* operētājsistēma lietotnes glabā 2 dažādās vietās:

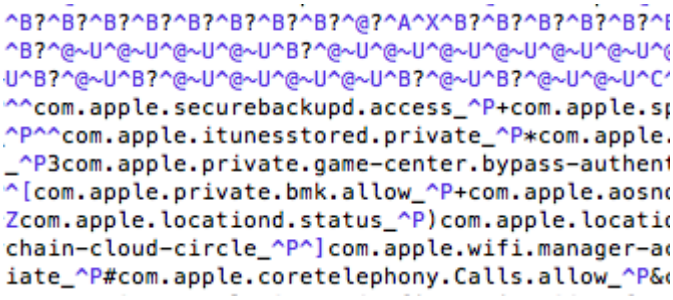
- lietotnes, kuras jau ir instalētas, kad ierīce tiek nopirkta, respektīvi jau instalētās *Apple* lietotnes, piemēram, „*Safari*", „*Calculator*", tiek glabātas mapē ar ceļu „*/Applications/";*
- lietotnes, kuras ir uzinstalētas no *AppStore,* tiek glabātas mapē ar ceļu *„/private/var/mobile/Applications*/".

Ja ierīcē ir ļoti daudz lietotņu, ir grūti atrast īsto, jo mapes nosaukums, kurās tiek glabāti visi lietotnes dati, tiek nejauši ģenerēts un sastāv no cipariem un burtiem, piemēram, „3002F825-6CE1-4C72-A7FB-A6E3159B6271". Viens veids, kā atrast vajadzīgo mapi, ir uzrakstīt komandu „*ls **" – šī komanda attēlo mapes, kuras atrodas vienu līmeni dziļāk, tas ir, ja ir mape ar nosaukumu "mape1", mape satur vēl divas mapes ar nosaukumiem "viens" un "divi", tad, uzrakstot komandu „*ls **", tiks attēlotas "viens" un "divi" mapes. Tas tomēr nav ieteicamas variants, jo, ja uz telefona ir 20 lietotnes, katra lietotnes mape ar nesaprotamo nosaukumu satur vēl piecas mapes, tad atrast vajadzīgo mapi būs grūti – būs pārāk daudz informācijas.

Otrs veids ir, pēc autora domām, ir vienkāršāks un ērtāks. *iOS* operētājsistēmā fails ar nosaukumu „*com.apple.mobile.installation.plist*". Tas ir fails, kurā tiek glabāta katras lietotnes

atrašanās vieta. Šo failu var atrast, rakstot meklēšanas komandu „*find / -name com.apple.mobile.installation.plist*”. Kad fails ir atrasts, tad vēlams viņu kopēt, piemēram, mapē ar nosaukumu „*tmp*”, jo būs nepieciešams to modificēt.

Kad konfigurācijas fails ir pārkopēts, tad tas jāatver – tam var izmantot jebkādu teksta redaktoru, piemēram, „*vim*” vai „*nano*”. Taču, atverot failu, dati tajā ir attēloti nesaprotamā formā, tas ir parādīts attēlā 5.4.



5.4 att. **Konfigurācijas fails**

Šādā veidā dati tiek attēloti visos „*plist*” konfigurācijas failos. Ir nepieciešama konvertācija un to var paveikt ar programmu „*plutil*”. Komanda izskatās šādi „*plutil -convert xml1 faila_nosaukums*”, kur dotajā gadījumā faila nosaukums ir „*com.apple.mobile.installation.plist*”. Izpildot šo komandu un atverot atkal to pašu failu, dati ir attēloti daudz saprotamāk, kā tas ir redzams attēlā 5.5.

5.5 att. **Lietotnes adrese**

Attēlā 5.5 ir izcelts mapes nosaukums, kur atrodas lietotnes „*Gmail*” dati.

5.3 Lietotnes „Latvijas Universitāte” testēšana

iOS lietotne „Latvijas Universitāte” ļauj Latvijas Universitātes studentam pēc autorizēšanās apskatīt studiju informāciju, savas sekmes, lekciju sarakstu. Ja aplikācijas lietotājs nav autorizējies lietotnē, tad viņam ir pieejama vispārīga informācija par Latvijas Universitāti un universitātes kontakti. Lietotni ir iespējams lejupielādēt *AppStore*.

Izpētot lietotni, darba autors pamanīja, ka tā piedāvā lietotājam saglabāt autorizācijas datus – lietotājvārdu un paroli. Tas nozīmē, ka dati tiek saglabāt pašā ierīcē, tāpēc darba autors nolēma izpētīt, kur un kādā veidā šie dati tiek glabāti.

Apskatot lietotnes failu sistēmu, darba autors pamanīja, ka tiek izmantota datubāze. To var apskatīt, izmantojot programmu „*sqlite3*”, komanda izskatās šādi „*sqlite3 LUIS.sqlite*”. Datubāze

satur vairākas tabulas, interesantākā tabula ir ar nosaukumu "*ZAPPDATA*" ("Z" burts tabulas priekšā nozīmē, ka datubāze tika veidota ar "*CoreData*" satvaru). Darba autors vairākas reizes autorizējās lietotnē, kā arī ievadīja nepareizus datus. No datiem tabulā var izsecināt, ka viena vērtība, kas tur glabājās ir pēdējais ievadītājs lietotājvārds. Tabulā arī uzglabājās veiksmīgi autorizēts lietotājvārds. Parole, pēc autora domām, tiek šifrēta vai arhivēta un tad saglabāta datubāzē, līdz ar to izgūt to no datubāzes darba autoram nesanāca.

Datubāzē ir arī citas tabulas, kas darba autoram likās interesantas, piemēram, "*ZNEWS*" tabula, kurā tiek glabātas ziņas, kas ir nosūtītas lietotājam, vai tabula "*ZCOURSE*", kurā tiek glabātas lietotāja atzīmes. Taču jāatzīmē, kad lietotājs veic atteikšanos no lietotnes, tabulu dati tiek dzēsti, kas, pēc autora domām, ir pareizi no drošības viedokļa. Taču jāatzīmē, ka dati no tabulas "*ZAPPDATA*" saglabājās arī pēc tam, kad lietotājs veic atteikšanos no lietotnes.

Apple iOS izstrādātājiem iesaka klases, metodes un mainīgos saukt tādos nosaukumus, lai varētu vienkārši saprast, ko tā metode dara vai kādu informāciju mainīgais uzglabā. Līdz ar to nākamajā testēšanas fāzē darba autors izpētīja lietotnes koda struktūru, tas ir, kādas ir klases, kādas ir metodes un kādi mainīgie.

Lai iegūtu lietotnes koda sastāvdaļas, ir nepieciešams atšifrēt lietotni. Visas lietotnes, kuras tiek iesniegtas un vēlāk ir pieejamas *AppStore*, tiek šifrētas. Savukārt lietotnes, kuras ir instalētas pēc noklusējuma, netiek šifrētas. Lai atšifrētu lietotni, ir jāizmanto programma „*clutch*". Dotā procedūra ir samērā vienkārša:

1. jāpārvietojas uz mapi, kurā atrodas lietotne „Latvijas Universitāte";
2. jāstartē „*clutch*" programma un jāpadod kā parametrs lietotnes nosaukums, dotajā gadījumā – „lu"

Diemžēl autoram nezināmo iemeslu dēļ, ar „Latvijas Universitātes" lietotni programma nestrādāja, tāpēc bija nepieciešams iegūt informāciju citā veidā. Šim nolūkam noderēja lietotne „*Flex*", kuru ir iespējams lejupielādēt no *Cydia*. „*Flex*" lietotne piedāvā ne tikai apskatīt, kādas klases, funkcijas tiek izmantotas lietotnē, bet arī modificēt funkcijas argumentus un atgriežamās vērtības, kas ir aprakstīts nedaudz vēlāk darbā.

Izmantojot lietotni „*Flex*", darba autors uzzināja, ka lietotnē ir kontrolieris ar nosaukumu „*LoginViewController*". Pēc nosaukuma var secināt, ka šis kontrolieris ir atbildīgs par lietotāja autorizāciju, kā arī šajā klasē ir mainīgie ar nosaukumiem „*UsernameTextField*" un „*PasswordTextField*". Pēc nosaukumiem var secināt, ka tie ir lietotājvārda un paroles teksta ievades lauki.

Iegūtā informācija ir noderīga kopā ar programmu „*cycript*", kas ļauj apskatīt datus un manipulēt ar tiem lietotnes izpildes laikā. Lai palaistu „*cycript*", lietotnei ir jābūt aktīvai, tas ir, atrasties priekšfonā. Kad tas ir izdarīts, tad programmu var palaist ar komandu „*cycript -p*

lietotnes_nosaukums", kur lietotnes nosaukumus šajā gadījumā ir „lu". Ja viss tika veiksmīgi palaists, tad atveras „*cycript*" konsole.

Nākamais uzdevums ir iegūt un saglabāt mainīgajā „*LoginViewController*" instanci. Darbības gaita ir redzama attēlā 5.6.

```
cy# app = [UIApplication sharedApplication]
#"<UIApplication: 0x17e462a0>"
cy# rootVC = app.keyWindow.rootViewController
#"<SplashViewController: 0x17d6a000>"
cy# *rootVC
{isa:SplashViewController,_view:#"<UIView: 0x17e4d540; frame = (0 0; 320 568); autoresize = RM+BM; layer =
Item: 0x17d75350>",_toolbarItems:null,_title:null,_nibName:@"qtL-4f-ZCq-view-FPG-Lc-0Ta",_nibBundle:#"NSBu
p> (loaded)",_parentViewController:null,_childModalViewController:#"<UINavigationController: 0x17d85c30>",
ansitionView:null,_modalPreservedFirstResponder:null,_dimmingView:null,_dropShadowView:null,_currentAction
:@[#"<UIStoryboardModalSegueTemplate: 0x19059440>",#"<UIStoryboardModalSegueTemplate: 0x17e606f0>"],_exter
rd:null,_savedHeaderSuperview:null,_savedFooterSuperview:null,_editButtonItem:null,_searchDisplayControlle
aceOrientation:1,_popoverController:null,_containerViewInSheet:null,_contentSizeForViewInPopover:{width:0,
ll,_afterAppearance:null,_explicitAppearanceTransitionLevel:0,_keyCommands:null,_viewControllerFlags:@erro
:0,_storyboardIdentifier:@"UIViewController-qtL-4f-ZCq",_transitioningDelegate:null,_modalPresentationCapt
eractiveTransitionDuration:0,_customNavigationInteractiveTransitionPercentComplete:0,_customTransitioningV
ull,_bottomLayoutGuide:null,_topBarInsetGuideConstraint:null,_bottomBarInsetGuideConstraint:null,_sourceVi
ll,_presentedStatusBarViewController:#"<UINavigationController: 0x17d85c30>",_edgesForExtendedLayout:0,__e
ontentSize:{width:0,height:0},_navigationControllerContentInsetAdjustment:{top:0,left:0,bottom:0,right:0},
e:{origin:{x:0,y:0},size:{width:0,height:0}},loadingView:null,_appDelegate:null,_popupView:null}
cy# navC = rootVC.childModalViewController
#"<UINavigationController: 0x17d85c30>"
cy# navC.viewControllers
@[#"<LoginViewController: 0x17d751c0>"]
cy# loginVC = navC.viewControllers[0]
#"<LoginViewController: 0x17d751c0>"
```

5.6 att. **Lietotnes „Latvijas Universitāte" testēšana**

Kad instance ir iegūta un tā ir saglabāta mainīgajā, ir iespējams izvadīt visus instances mainīgos, tas ir izdarāms ar komandu „**loginVC*". Kā redzams attēlā 5.7, var redzēt mainīgo vērtības, kas atbilst ievadītām vērtībām teksta ievades laukos autorizācijas logā.

```
20 568); autoresize = W+H; layer = <CALayer: 0x17efb180>>",_appDelegate:null,_popupView:null,check
edImage:#"< UIResizableImage: 0x19081f70>",uncheckedImage:#"< UIResizableImage: 0x17f29dc0>",isChe
cked:1,UsernameTextField:#"<UITextField: 0x17d7f740; frame = (34 248; 266 31); text = 'mm1[redacted]
lipsToBounds = YES; opaque = NO; autoresize = RM+BM; gestureRecognizers = <NSArray: 0x19221d80>; l
ayer = <CALayer: 0x192a8440>>",PasswordTextField:#"<UITextField: 0x17d7e9d0; frame = (34 309; 266
31); text = 'M[redacted]ToBounds = YES; opaque = NO; autoresize = RM+BM; gestureRecognizers =
<NSArray: 0x191c5a40>; layer = <CALayer: 0x17dd8200>>",scrollView:#"<UIScrollView: 0x17f66f40; fra
me = (-6 0; 326 504); clipsToBounds = YES; autoresize = RM+H; gestureRecognizers = <NSArray: 0x192
```

5.7 att. **Lietotnes autorizācijas dati**

Ir jāpiebilst, kā šāda veida datu iegūšana ir iespējama, kad lietotājs ir jau autorizējies lietotnē. Līdz ar to ir nepieciešams lietotāju – LU studentu „piespiest" autorizēties lietotnē, kas ir instalēta uz uzbrucēja vai testēja *iOS* ierīces. Šāda veida uzbrukumu vai atvieglot jau augstāk minētā lietotne „*Flex*".

Kā jau tika minēts darbā, lietotne „*Flex*" spēj modificēt funkciju argumentus. Lietotnē „Latvijas Universitāte" autorizācijas logā ir iespējams izvēlēties, vai saglabāt lietotāja autorizācijas datus. Uzbrukuma scenārijs varētu būt šāds:

1. ar lietotnes „*Flex*" palīdzību modificēt metodi ar nosaukumu „*performLoginWithUsername:(id)password:(id)permanentSessionFlag:(BOOL)info:(id)*".

Pēc parametru nosaukumiem un to tipiem var secināt, ka mainīgais „*permanentSessionFlag*" atbild, vai saglabāt lietotāja autorizācijas datus vai nē. „*Flex*" lietotnē tiek uzstādīta vērtība „*TRUE*", kā rezultātā, pat ja lietotājs nebūs atzīmējis, ka vēlās saglabāt autorizācijas datus, tas tiks izdarīts, lietotājam nezinot;

2. uzbrucējs pienāk pie potenciālā „upura" un saka, ka viņam nestrādā LU lietotne un vai viņš nevar pārbaudīt ar saviem datiem;
3. uzbrucējs iedot savu ierīci ar LU lietotni, kurā izvēles rūtiņa „Atcerēties mani" nav ieķeksēta un ļauj savam „upurim" ievadīt datus;
4. kad dati ir ievadīti un autorizācija tika veikta, uzbrucējs lietotāja priekšā apstādina lietotnes darbu un pēc lietotāja domā viņš tika atteikts no lietotnes, bet īstenībā tā nav;
5. uzbrucējs tagad var iegūt lietotāja datus, pieslēgties LUIS sistēmai un iegūt papildus datus par savu „upuri", manipulēt ar tiem.

Pēdējais tests, ko darba autors veica lietotnei, bija saistīts ar pārbaudi, vai lietotne neizvada kaut kādus datus. Lai to izdarītu, bija nepieciešama integrētā izstrādes vide "*Xcode*". Ierīce tika savienota ar datoru, izmantojot *USB*, tika atvērta "*Xcode*" programmas konsoles sadaļa, kurā tiek izvadīta dažāda informācija, kas ir saistīta ar *iOS* ierīci. Autors secināja, ka pieprasījumi, kas tiek veikti uz serveri, tiek izvadīta, kā tas ir redzams attēlā 5.8.

5.8 att. **Pieprasījumu izvadīšana**

No vienas puses, šos datus ir iespējams iegūt tad, kad ierīce ir pieslēgta pie datora, tomēr ir iespējams redzēt kāda struktūra ir pieprasījumam, un tas var kalpot kā sākuma punkts, ja tiek plānota testēšana pašam serverim.

5.4 Lietotnes „Draugiem" testēšana

Lietotne „Draugiem" piedāvā gandrīz visas tās pašas iespējas, ko tiešsaiste http://draugiem.lv. Lietotni ir iespējams lejupielādēt *AppStore*.

Izpētot lietotnes darbības principus, darba autors pamanīja, ka lietotne, līdzīgi kā lietotne „Latvijas Universitāte", saglabā lietotāja autorizācijas datus. Atšķirība ir tāda, ka tas tiek darīts automātiski – lietotājam netiek piedāvāta izvēle saglabāt datus vai nē. Līdz ar to datus ir nepieciešams uzglabāt ierīcē.

Darba autors izpētīja lietotnes failu sistēmu un secināja, ka nav nekādu failu vai datubāzes, kas varētu liecināt, ka tajā tiek glabāt lietotāja piekļuves dati. Dati, kas tika uzglabāti failu sistēmā, bija saistīti ar kompānijas „*Google*" piedāvāto pakalpojumu – kartēm. Nākamais solis ielaušanās testēšanā bija izpētīt lietotnes kodu – klases, funkcijas, lai iegūtu pēc iespējas noderīgāku informāciju par lietotnes darbību.

Atšifrēšanas process „Draugiem" lietotnei ir veiksmīgs, kā rezultātā ir izveidots fails ar paplašinājumu „*ipa*". Tas ir vienkāršs arhīvs, un tā atarhivēšanai tiek izmantota lietotne "*unzip*". Nākamais solis pēc failu atarhivēšanas ir iegūt visu klašu, funkciju un mainīgo nosaukumus un tam ir paredzēta programma „*class-dump-z*". Dotā komanda ir jāpalaiž, atrodoties mapē „draugiem.app" un tā izskatās šādi: „*class-dump-z lietotnes_nosaukums > klases_informācijas_fails*", kur lietotnes nosaukums dotajā gadījumā ir „draugiem" un *klases_informācijas_fails* ir jebkāds nosaukums failam, kurš saturēs informāciju par lietotnes uzbūvi, dotajā gadījumā „draugiem-dump". Kad komanda ir izpildīta, jāmēģina atrast kaut kas interesents un noderīgs ielaušanās testēšanai. Neliels faila fragments ir redzams attēlā 5.9, kurā ir izcelta koda daļa, kas, pēc autora domām, ir interesanta no ielaušanās testēšanas viedokļa.

```
__attribute__((visibility("hidden")))
@interface DRLoginViewController : UIViewController <UITextFieldDelegate, DRKeyboardStateListenerDelega
te> {
        UIView* loginContentView;
        UITextField* userTextField;
        UITextField* passwordTextField;
        UIButton* loginButton;
        BOOL viewIsRotating;
}
@property(readonly, copy) NSString* debugDescription;
@property(readonly, copy) NSString* description;
@property(readonly, assign) Class superclass;
@property(readonly, assign) unsigned hash;
-(void).cxx_destruct;
-(void)shakeLoginContentViewToCenterWithDuration:(double)duration;
-(void)animateLoginContentViewToCenterWithDuration:(double)duration animationCurve:(int)curve;
-(void)tapGestureRecognized:(id)recognized;
-(void)removeKeyboard;
-(BOOL)textFieldShouldReturn:(id)textField;
-(void)DRKeyboardFrameWillChangeTo:(CGRect)drkeyboardFrame withDuration:(float)duration animationCurve:
(int)curve;
-(void)didRotateFromInterfaceOrientation:(int)interfaceOrientation;
-(void)willRotateToInterfaceOrientation:(int)interfaceOrientation duration:(double)duration;
-(BOOL)isDeveloperAuthentification;
-(void)loginButtonTapped:(id)tapped;
-(void)setUpLoginButtonAppearence;
-(void)viewWillAppear:(BOOL)view;
-(void)viewDidLoad;
-(id)initWithNibName:(id)nibName bundle:(id)bundle;
@end
```

5.9 att. **Lietotnes „Draugiem" klases struktūra**

Veicot izpēti, ir atrasta klase „*LoginViewController*", kurā ir daudz sološa metode ar nosaukumu „*loginButtonTapped:*". Pēc nosaukuma var secināt, ka šī metode tiek izsaukta, kad lietotājs vēlas autorizēties lietotnē un nospiež pogu „Ienākt". Darba autora nākamais testēšanas mērķis ir mēģināt aizvietot doto metodi ar autora definēto, tas ir, veikt funkciju apmaiņu.

Lai varētu veikt funkciju apmaiņu, ir nepieciešams palaist programmu „*cycript*", iegūt

„*LoginViewController*” klases instanci. Darbības gaita ir redzama attēlā 5.10.

```
oldPhone:~ root# cycript -p Draugiem
cy# app = [UIApplication sharedApplication]
#"<UIApplication: 0x17d8e500>"
cy# rootVC = app.keyWindow.rootViewController
#"<DRDrawerViewController: 0x17e9d4b0>"
cy# *rootVC
{isa:DRDrawerViewController,_view:#"<UIView: 0x17e9f850; frame = (0 0; 320 56
arItems:null, title:null, nibName:null, nibBundle:#"NSBundle </var/mobile/App
_childModalViewController:#"<DRLoginViewController: 0x17d9baa0>",_parentModal
tResponder:#"<DRNavigationController: 0x18919600>",_dimmingView:null,_dropSha
ableForViewLoading:null,_topLevelObjectsToKeepAliveFromStoryboard:null,_saved
,_modalTransitionStyle:0,_modalPresentationStyle:0,_lastKnownInterfaceOrienta
ight:0},_formSheetSize:{width:0,height:0},_recordedContentScrollView:null,_af
ror,_retainCount:2,_ignoreAppSupportedOrientations:0,_viewHostsLayoutEngine:0
nce:0,_childViewControllers:@[#"<DRNavigationController: 0x18919600>"],_custo
,_customTransitioningView:null,_navigationControllerContentOffsetAdjustment:0
eConstraint:null,_sourceViewControllerIfPresentedViaPopoverSegue:null,_modalS
_edgesForExtendedLayout:15,__embeddedView:null,__embeddingView:null,__embedde
t:{top:0,left:0,bottom:0,right:0},_contentOverlayInsets:{top:20,left:0,bottom
imitMenuViewSize:1,_liveBlur:1,_visible:0,_automaticSize:0,_panGestureRecogni
=panGestureRecognized:, target=<REFrostedContainerViewController 0x17db6320>)
rSaturationDeltaFactor:1.7999999523162842,_liveBlurBackgroundStyle:1,_delegat
ageViewWidth:0,_image:null,_imageView:null,_containerViewController:#"<REFros
th:280,height:0},_calculatedMenuViewSize:{width:0,height:0},_developerConsole
r:#"<DRNavigationController: 0x18919600>",_leftDrawerViewController:#"<DRLeft
cy# loginVC = rootVC.childModalViewController
#"<DRLoginViewController: 0x17d9baa0>"
```

5.10 att. **Lietotnes „Draugiem” testēšana**

Kad instance ir saglabāta mainīgajā, var mēģināt veikt funkciju apmaiņu. Procesa gaita ir redzama attēlā 5.11.

```
cy# username = loginVC.view.subviews[0].subviews[3].text
@"maximilian@inbox.lv"
cy# password = loginVC.view.subviews[0].subviews[4].text
@""
cy# loginVC->isa.messages["loginButtonTapped:"] = function() { t1 = [NSString stringWithContentsOfFile:
@"/tmp/draugiem.txt" encoding:4 error:nil]; text = [NSString stringWithFormat:@"%@\nUsername: %@; Passw
ord: %@",t1, username, password]; [text writeToFile:@"/tmp/draugiem.txt" atomically:YES encoding:4 erro
r:nil]; [loginVC shakeLoginContentViewToCenterWithDuration:1]; }
function () {t1=objc_msgSend(NSString,"stringWithContentsOfFile:encoding:error:",Instance.box("/tmp/dra
ugiem.txt"),4,nil),text=objc_msgSend(NSString,"stringWithFormat:",Instance.box("%@\nUsername: %@; Passw
ord: %@"),t1,username,password),objc_msgSend(text,"writeToFile:atomically:encoding:error:",Instance.box
("/tmp/draugiem.txt"),YES,4,nil),objc_msgSend(loginVC,"shakeLoginContentViewToCenterWithDuration:",1);}
cy# [loginVC loginButtonTapped:NULL]
```

5.11 att. **Metodes apmaiņa**

Funkciju apmainīšanas rezultāts ir šāds: lietotājs ievada savu lietotājvārdu un paroli, šie dati tiek ierakstīti “draugiem.txt” failā. Tālāk tiek simulēta nepareizo datu ievadīšana – tiek “pakustēti” ievades lauki, kas liek lietotājam domāt, ka viņš ir nepareizi ievadījis savus datus. Līdz ar to var teikt, ka ir izveidots arī *DoS* uzbrukums. Ideālais gadījums būtu, ja lietotne ļautu autorizēties, tomēr darba autoram to nesanāca panākt. Jāatzīmē, kā šāda vieta datu pārtveršana beigsies tiklīdz lietotne tiks restartēta, tas ir tāpēc, ka šis ir izpildlaika uzbrukums.

Tā kā lietotāja autorizācijas dati netika atrasti lietotnei paredzētā failu sistēmā, tad visdrīzāk tie tiek glabāti *keychain*. Izmantojot speciāli uzrakstītu skriptu, ir iespējams izgūt visus datus no

keychain glabātuves. Skripts tiek palaists ar komandu „*./keychain-dumper*”. Veicot meklēšanu izvadītajā informācijā, var atrast ierakstu, kas atbilst „Draugiem” lietotnei. Ieraksts ir redzams attēlā 5.12.

5.12 att. **Autentifikācijas dati**

Pēc attēlā 5.11 redzamās informācijas var veikt secinājumu, ka tikko lietotājs ir autorizējies „Draugiem.lv” sistēmā, viņam tiek piešķirts unikāls identifikators, *API* atslēga, ar kuras palīdzību tad tiek veikta turpmākā saziņa ar serveri. Jāpiemin, ka tikko lietotājs ir veicis atteikšanos no lietotnes, šis ieraksts tiek dzēst no *keychain* glabātuves, kas, pēc darba autora domām, ir pareizi.

5.5 *Keychain* analīze

Keychain ir *iOS* operētājsistēmas glabātuve, kurā lietotnes vai uzglabāt dažāda veida paroles, sertifikātus, citu svarīgu un sensitīvu informāciju. Tā tiek uzskatīta par diezgan drošu glabāšanas iespēju, tomēr, ja ierīcei nav uzstādīts *passcode*, tad ir diezgan viegli iegūt visu informāciju no glabātuve.

Lai iegūtu datus no *keychain*, ir jāpalaiž „*keychain-dumper*” skripts, kurš ir aprakstīts iepriekšējā nodaļā. Pēc autoram domām, vislielākās uzmanības vērti ir dati no lietotnes „*iTunes Connect*”. Lietotnei piedāvā pārvaldīt izstrādātāja izveidotās un publicētās lietotnes, pārvaldīt savas komandas lietotājus, rēķinus u.c. svarīgas lietas. Attēlā 5.13 var redzēt, ka lietotājvārds un paroli netiek šifrēti, kad tos saglabā *keychain* glabātuvē. Un, ja ierīcei nav uzstādīts *passcode*, tad uzbrucējam ir diezgan vienkārši iegūt šos datus.

```
Generic Password
----------------
Service: iTunes Connect Session
Account: AccountName
Entitlement Group: VT6C486PNU.com.apple.itunesconnect.mobile
Label: (null)
Generic Field: (null)
Keychain Data: [redacted]@gmail.com

Generic Password
----------------
Service: iTunes Connect Session
Account: AccountPassword
Entitlement Group: VT6C486PNU.com.apple.itunesconnect.mobile
Label: (null)
Generic Field: (null)
Keychain Data: d[redacted]
```

5.13 att. ***Keychain*** **testēšana**

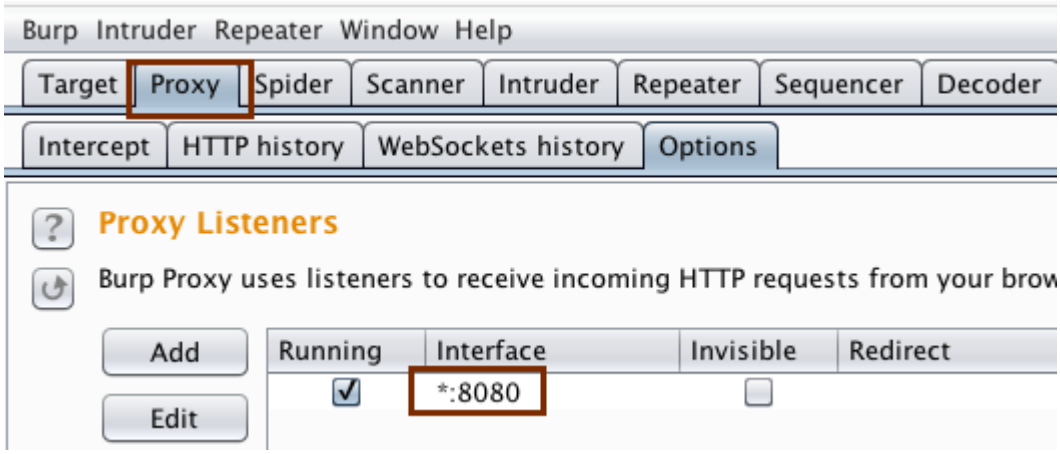
Skripta izvadītā informācija satur arī bezvadu tīklu paroles, citu lietotņu autorizācijas datus, piemēram, „*Gmail*”, „*9gag*”.

5.6 Lietotņu datu pārraides testēšana

Ļoti daudzas lietotnes sadarbojas ar serveri, lai atjauninātu informāciju, piedāvātu jaunus pakalpojumus, ielādētu jaunus līmeņus spēlēs, veiktu konta pārskatu vai maksāšanu, izmantojot bankas lietotnes. Darba autors izpētīja veidu, kā varētu pārtvert šo saziņu ar serveri, lai izpētītu, kādi dati tiek sūtīti, kāds ir pieprasījumu un atbildes forma.

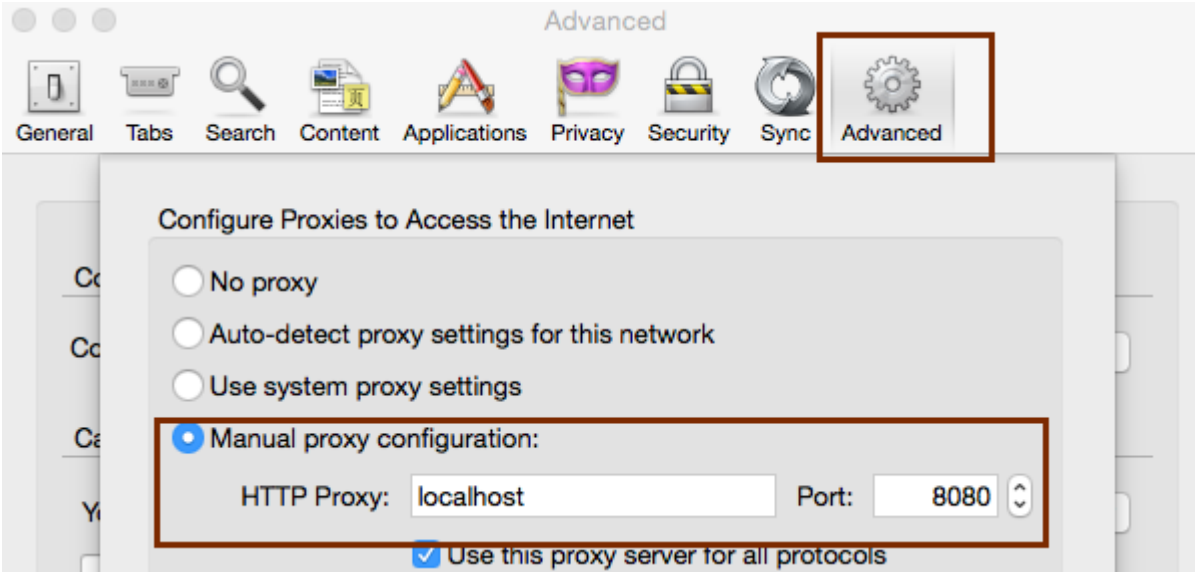
Lai veiktu testu, ir nepieciešama programma „*Burp suite*” (turpmāk „*burp*”). Šī programma darbojas kā starpniekserveris.

Pirmais solis, kas ir jāveic, ir jālejupielādē „*burp*” sertifikāts. Lai to izdarītu, ir jāstartē „*burp*” programma un jāveic neliela konfigurācija, kā tas ir redzams attēlā 5.14.



5.14 att. ***„Burp”*** **konfigurācija**

Tālāk ir jāatver jebkurš tīmekļa pārlūks un jānokonfigurē tā, lai tas izmantotu „*burp*" starpniekserveri. Darba autors konfigurāciju veica pārlūkā „*Mozilla Firefox*". Konfigurācijas parametri ir redzami attēlā 5.15.



5.15 att. ***„Mozilla Firefox" konfigurācija***

Kad iestatījumi ir saglabāti, tad ir jāapmeklē vietne, kura nodrošina *HTTPS* pieeju, piemēram, https://gmail.com. Kad tiek atvērta jebkura *HTTPS* vietne, parādās paziņojums par nedrošo savienojumu. Ir jāizvēlas „*I Uderstand the Risks*" opcija, tad ir jānospiež poga „*Add execption:*". Parādīsies vēl viens logs, kurā ir jānospiež poga „*View...*". Tālāk augšā ir jāizvēlas „*Details*" poga, „*Certificaty Hierchy*" sadaļā ir jāizvēlas ieraksts „*PortSwigger CA*", tad jānospiež poga „Export...", parādīsies failu saglabāšanas logs, kurā jānorāda, ka failu jāsaglabā ar paplašinājumu „*crt*". Kad tas izdarīts un sertifikāts ir saglabāts, nākamais solis ir šo sertifikātu saglabāt *iOS* ierīcē.

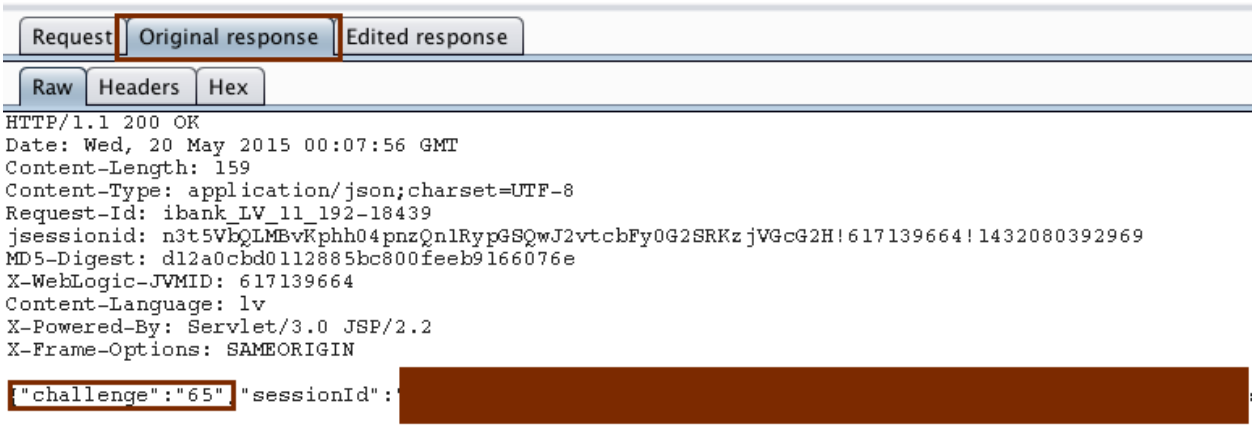
Ir vairāki veidi, kā var nosūtīt sertifikātu uz *iOS* ierīci. Viena iespēja ir nosūtīt e-pastu, tad to atvērt uz *iPhone.* Cita iespēja ir palaist tīmekļa serveri, ko arī izvēlējās darba autors. Serveris tika startēts "*python*" programmēšanas valodā, un, lai to izdarītu, ir jāuzraksta sekojoša komanda programmā „*Terminal*": „*python -m SimpleHTTPServer 8000*".

Tālāk caur jebkuru pārlūka lietotni uz *iOS* ierīces pieslēdzas datoram (jāievada datora *IP* adrese un ports 8000), uz kura tika palaists serveris un jālejupielādē sertifikāts. Kad sertifikāts ir lejupielādēts, nepieciešams to apstiprināt. Lai to izdarītu, jāiet uz lietotni „*Settings*", tālāk uz sadaļu „*General*", tad „*Profiles*", tad jāizvēlas sertifikātu un jāakceptē tas.

Trešais solis ir veikt konfigurāciju *iOS* ierīcē, lai tā izmantotu „*burp*" starpniekserveri. Lai to izdarītu, jāatver „*Settings*" lietotne, tad uz sadaļu „*Wi-Fi*", tad jānospiež poga ar „i" burtu tīklam, kuram pašreiz ierīce ir pieslēgta. Atvērsies konfigurācijas logs, kurā ir jāveic „*Http proxy*"

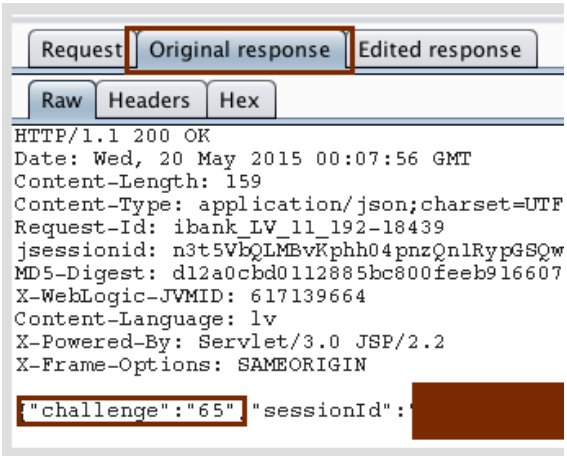
konfigurācija – jāizvēlas „*Manual*" sadaļa, „*Server*" šūnā jāievada datora *IP* adrese, uz kura ir palaista programma „*burp*", ports pēc noklusējuma ir 8080. Kad tas ir izdarīts, datus saglabā. Ar to konfigurācija beidzās, un ir iespēja tagad apskatīt datu pārraidi starp lietotni un serveri.

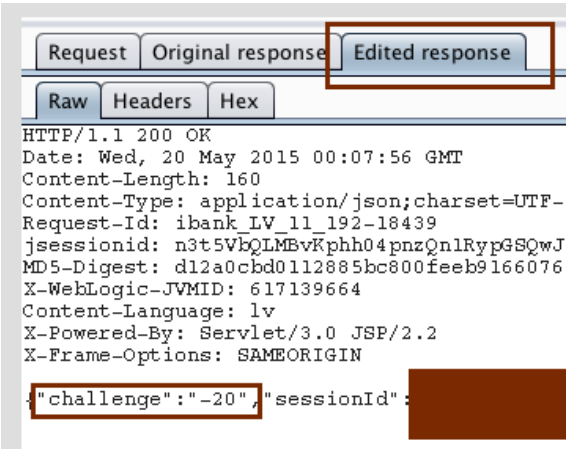
Darba autors izvēlējies lietotni „*Swedbank*". Tā nodrošina „*Swebank*" bankas klientiem gandrīz tādus pašus pakalpojums kā internetbanka. Tā ir pieejama *AppStore*, Sākot darbu ar lietotni, var pamanīt, ka pieprasījumi neizpildās, tas ir tāpēc ka programma „*burp*" to pārtver pēc noklusējuma. Tad uzbrucējs vai testētājs var izvēlēties, ko ar šo pieprasījumu darīt – modificēt, laist cauri nemodificētu vai vispār pārtraukt. Autorizēšanās procesa atbilde ir redzams attēlā 5.16.



5.16 att. **Autorizēšanas procesa atbilde**

Viens no *DoS* iespējamiem uzbrukumiem ir, kad uzbrucējs maina lietotāja *ID*, neļaujot pieslēgties internetbankai. „*Burp*" programma ļauj modificēt arī atbildi no servera, piemēram, ja lietotājam no servera puses atnāk paziņojums, ka no banku koda kartes jāievada 12. kods, tad uzbrucējs var modificēt šo atbildi tā, lai lietotājam lietotne attēlotu, ka jāievada 23. kods, šāda veida uzbrukums ir parādīts attēlos 5.17 (a) un 5.17 (b).

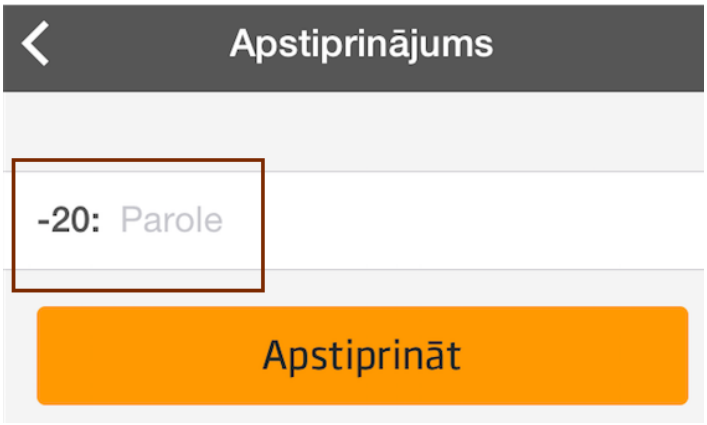




5.17 att. **Izmaiņas servera atbildē**

(a) oriģinālā atbilde (b) modificētā atbilde

Attēlā 5.18 ir redzams, kā tas parādās lietotnē.



5.18 att. **Izmaiņas lietotnē „*Swedbank*”**

Programmu „*Burp*” var izmantot arī, lai pētītu kādi pieprasījumi un atbildes nāk, komunicējot lietotnei ar serveri. Šāda informācija ir ļoti noderīga ielaušanās testētājam.

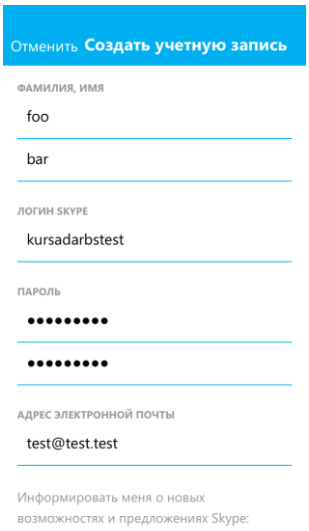
6. *iOS* DROŠĪBAS PASĀKUMI

Izstrādājot lietotnes *iOS* ierīcēm, programmētājiem vienmēr ir jādomā par datu drošību. Neveiksmīgi izstrādāta lietotne var būt par iemeslu tam, ka lietotāja dati tiek pārtverti un izmantoti pret pašu lietotāju.

6.1 Drošības ieteikumi izstrādātājiem

Viens no potenciālajiem riskiem parādās, kad lietotne nonāk fona darbības režīmā. Šīs pārejas laikā *iOS* veic lietotnes ekrānuzņēmumu un saglabā to lietotnes faila sistēmā nešifrētu. Ja lietotnes ekrāns satur sensitīvu informāciju par lietotāju vai lietotāja datus, tad veidojas iespēja, ka šie dati var kļūt publiski pieejami. Lietotne pāriet fona režīmā kad:

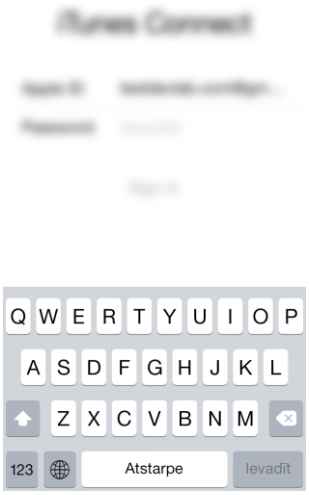
- lietotājs izvēlēs citu lietotni;
- ierīce pāriet miega režīmā;
- ierīce tiek izslēgta;
- lietotājs nospiež *„Home”* pogu.



6.1. att. **Lietotnes *„Skype”* ekrānuzņēmums**

Attēlā 6.1 var redzēt ekrānuzņēmumu lietotnei *„Skype”* brīdī, kad notiek lietotāja reģistrācija un, kā var redzēt, tagad ir iespējams uzzināt lietotāja vārdu, uzvārdu, e-pastu, *Skype* lietotājvārdu.

Apple ir izstrādājis vadlīnijas, kurās ir ieteikts, ka ir jāsamaina dati, kas atrodas lietotnes ekrānā, brīdī, kad lietotne pāriet fona režīmā vai ir jāuzstāda noklusējuma ekrāns. Tādā veidā saglabātais ekrānuzņēmums nesaturēs sensitīvus datus par lietotāju.



6.2. att. **Lietotnes *„iTunes Connect”* ekrānuzņēmums**

Attēlā 6.2 ir redzams ekrānuzņēmums lietotnei *„iTunes Connect”*, kuras izstrādātājs ir *Apple*. Var redzēt, ka attēlam ir piešķirts izplūdes efekts, kas slēpj lietotājvārdu.

iOS ņem vērtības no *UITextField* elementa laukiem un saglabā tās atklāta teksta veidā failu sistēmā kā daļu no automātiskās pabeigšanas (*AutoComplete*) funkcionalitātes. Līdz ar to, ja lietotājs vairākas reizes ievada paroli, iespējams, tā var parādīties atklāta teksta veidā. Veidi, kā iespējams apturēt šādu sistēmas darbību, kuras rezultātā lietotāja sensitīvi dati ir pieejami, ir:

- izmantot „*SecureTextEntry*” atribūtu — šis atribūts norāda, ka „*UITextField*” klases elementa attēlotais teksts netiek rādīts lietotājam, ļoti noderīgi, ja elements ir paredzēts paroles ievadei;
- izslēgt automātiskās pabeigšanas funkciju — kad vien ir iespējams, automātiskās pabeigšanas ir jāizslēdz sensitīvo datu laukiem. To var paveikt, izmantojot „*UITextAutocorrectionTypeno*” atribūtu.

iOS atbalsta iespēju lietotnei reģistrēt savu protokola apdarinātāju. Piemēram, ja lietotnes nosaukums ir *„MyTestApp”*, tā reģistrē protokola shēmu *„TestApp”*, tad citas lietotnes var palaist lietotni *„MyTestApp”*, izsaucot protokolu ar kodu: „*TestApp://Print?name=Foo&lastName=Bar*”. Šajā piemērā *„MyTestApp”* lietotnei būtu jāizvada teksts, kas saturētu lietotāja vārdu un uzvārdu. Ir svarīgi pirms šīs darbības izpildīšanas informēt lietotāju, citādi citas lietotnes var izpildīt nevēlamas darbības, lietotājam par to nezinot. Pie tam, citas lietotnes var reģistrēt tādu pašu *URL* shēmu un pārtvert pieprasījumus “pareizajai” lietotnei. Tāpēc, kad vien ir iespējams, ir jāizvairās no sensitīvas informācijas padošanas, izmantojot doto veidu, lai nodrošinātu, ka lietotāja informācija netiek nopludināta.

Apple piedāvā drošu glabātuvi (konteineri), kurā izstrādātājs var glabāt paroles serveriem, šifrēšanas atslēgas. Šo glabātuvi sauc *„keychain”* un to var izmantot individuāla lietotne, vai arī

tā var būt kopīga. *Mac OS X* sistēmā lietotājam tiek pieprasīts sākumā atbloķēt pašu glabātuvi, un tad tiek atbloķēts glabātuves saturs. *iOS* sistēmā lietotājam tas netiek prasīts. Tā vietā *iOS* uzglabā atslēgu, kura ir nepieciešama, lai atbloķētu *keychain*, padarot šo procesu neredzamu lietotnei, kurai pieder šī glabātuve, un lietotājam. *Keychain* ir pieejama tikai tai lietotnei, kura to ir izveidojusi. Ir jāpiemin, ka ir iespēja vairākām lietotnēm izmantot vienu un to pašu glabātuvi, ja tās pieder vienam izstrādātājam.

Korekts veids, kā pārbaudīt drošu (*SSL* vai *TLS*) pieslēgumu *iOS* sistēmā, ir izmantojot *Apple „Certifikate, Key, Trust Services"* satvaru. Satvars pārbauda:

- neuzticamas sertificēšanas iestādes;
- sertifikāta derīguma termiņu;
- domēnu vārdu sistēmas (*DNS – Domain Name System)* nesaderību, u.c.

Pēc noklusējuma, *Apple* ierīces aizsargā lietotāju no *Man-in-the-Middle* uzbrukumiem. Lai pārliecinātos, ka *SSL* veic savu darbu, ir svarīgi pārbaudīt, ka izstrādātājs nav izslēdzis *SSL* drošības pārbaudi, kad izmanto kādu no dotajām metodēm:

- „*NSURLRequest*";
- „*setAllowsAnyHTTPSCertificate*";
- „*NSURLConnection*" delegāts;
- „*continueWithoutCredentialForAuthenticationChallenge*";
- „*CFNetwork*";
- „*kCFStreamSSLAllowsExpiredCertificates*".

Daudzas lietotnes izmanto konfigurācijas failu ar paplašinājumu „*plist*". Faila struktūra ir līdzīga vārdnīcas struktūrai — ir atslēga, un katrai atslēgai ir piekārtota vērtība. Daudzas lietotnes šajos konfigurācijas failos saglabā tīmekļa servisa vai servera lietotājvārdu un paroli, dažādus lietotāja datus, bloķēšanas kodu. Ir svarīgi atcerēties, ka pēc noklusējuma faili netiek šifrēti, līdz ar to izstrādātājs padara savu serveri, lietotni vai servisu ievainojumu.

Ja *iOS* lietotne izmanto datubāzi — nav svarīgi, vai tā ir „*SQLite*" datubāze, vai tiek izmantota iebūvētā „*CoreData*" klase, tā netiek šifrēta pēc noklusējuma, un visi dati ir pieejami atklātā tekstā. Viss, kas nepieciešams, ir programma, kura spēj strādāt ar datubāzes failiem.

Ja lietotne apstrādā lietotāja sensitīvos datus, tad, pēc autora domām, izstrādātājam vajadzētu nodrošināt pārbaudi, vai *iOS* ierīcei nav veikts *jailbreak* process. Kad tiek veikts *jailbreak* process, *iOS* operētājsistēmā parādās jauni faili, līdz ar to ar vienkāršu pārbaudi, vai fails eksistē, var arī pārbaudīt, vai ierīcei ir veikts *jailbreak*. Daži no failiem, kas parādās pēc *jailbreak*, ir [35]:

- „*/bin/bash*";

- „*/var/log/syslog*”;
- „*/private/var/lib/cydia*” u.c.

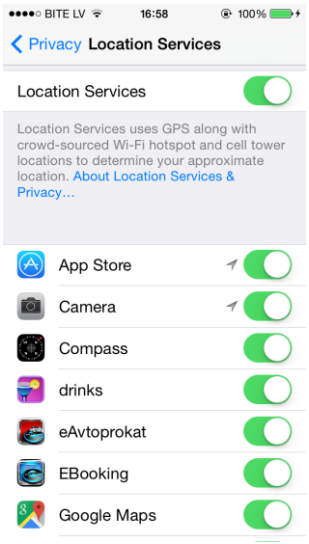
6.2 Lietotāja drošības pasākumi

Apple piedāvā iebūvētas kontroles un opcijas, kas ļauj lietotājiem noteikt, kā un kad lietotne izmanto lietotāja informāciju, kā arī tiek noteikts, kāda informācija tiek izmantota.

Lokācijas servisi izmanto *GPS*, *Bluetooth*, bezvada tīklu un mobilo sakaru torņus, lai noteiktu lietotāja aptuveno atrašanās vietu. Lokācijas servisi var tikt izslēgti, izmantojot *„Settings”* lietotni, kurā arī ir iespējams iestatīt, kādas lietotnes drīkst izmantot ierīces atrašanās vietas koordinātes. Lietotne var pieprasīt lokācijas datus tikai tad, kad tā tiek izmantota. Lietotājs var aizliegt piekļuvi lietotnēm, un var mainīt savu izvēli lietotnē *„Settings”*. Izmantojot šo pašu lietotni, var uzstādīt dažādas darbības iespējas:

- aizliegt;
- atļaut tikai tad, kad lietotne ir aktīva;
- vienmēr atļaut.

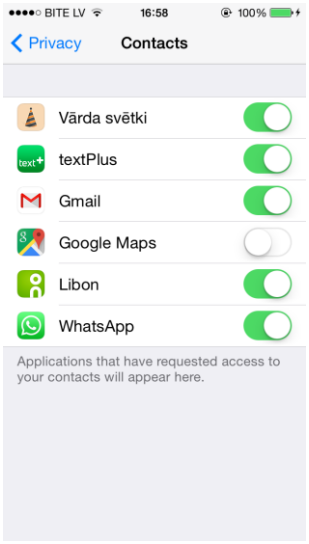
Attēlā 6.3 ir parādīta atrašanās vietas pakalpojumu (*Location services*) sadaļa, kura uzrāda lietotnes, kuras ir pieprasījušas atļauju izmantot lietotāja atrašanās vietu. Šajā pašā ekrānā var atļaut vai aizliegt lietotnēm izmantot lietotāja atrašanās vietu.



6.3. att. **Atrašanās vietas pakalpojumu sadaļa**

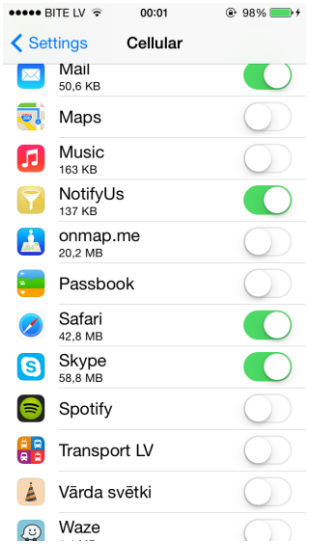
iOS palīdz pasargāt lietotāja datus no lietotnēm. Lietotnē *„Settings”* ir iespējams redzēt, kādām lietotnēm ir pieejama kāda veida informācija, kā arī ir iespējams mainīt šīs atļaujas. Lietotnes var pieprasīt izmantot informāciju no dažādām lietotnēm un servisiem, piemēram, *„Contacts”*, *„Calendars”*, *„Photos”*, *„Reminders”*, sociālo tīklu kontiem, piemēram,

„Facebook”, „Twitter”, ierīces kamerai, mikrofonam utt. Attēlā 6.4 var redzēt lietotnes, kuras ir pieprasījušas izmantot datus par lietotāja kontaktiem.



6.4. att. **Lietotāju kontaktu piekļuves kontrole**

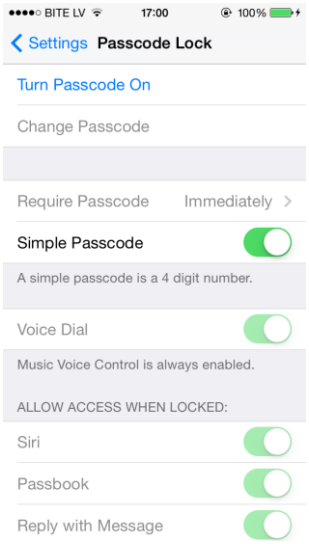
iOS dod iespēju lietotājam uzstādīt atļaujas lietotnēm, kurām ir nepieciešams interneta pieslēgums, bet izmanto datu pārraidi. Tādā veidā var kontrolēt lietotnes, kurām ir nepieciešama liela un bieža datu apmaiņa, piemēram, *„Skype”*. Attēlā 6.5 var redzēt, kurām lietotnēm ir dota atļauja izmantot mobilo sakaru tīklu, lai pārraidītu datus.



6.5. att. **Datu pārraides kontrole**

Visām *iOS* ierīcēm iespējams uzstādīt *passcode*. Kad ierīcei tiek uzstādīts *passcode*, lietotājam automātiski tiek ieslēgta datu aizsardzības (*Data Protection*) tehnoloģija. *iOS* atbalsta 4 ciparu un nenoteikta garuma ciparu un burtu kombinācijas kodu. *Passcode* ne tikai ļauj atbloķēt ierīci, bet arī aizsargā dažādas šifrēšanas atslēgas. Ja *passcode* minēšanai tiek izmantots pārlases uzbrukums, tad pēc noteikta skaita nepareizi ievadīto kombināciju, katra nākamā ievade tiek

palēnināta no operētājsistēmas puses. Ir iespējams arī uzstādīt, ka pēc 10 neveiksmīgām koda ievadītajām kombinācijām, visi dati uz ierīces tiek dzēsti.



6.6. att. ***Passcode*** **uzstādīšana**

Attēlā 6.6 ir redzama lietotnes *„Settings"* sadaļa, kurā ir iespējams uzstādīt ierīcei bloķēšanas kodu. Opcija *„Simple Passcode"* uzstāda, vai kods ir vienkāršs — 4 cipari — vai sarežģīts, tas ir, nenoteikta garuma ciparu un burtu kombinācija.

Telefoniem *iPhone 5s, iPhone 6* un *iPhone 6+* ir uzstādīts pirksta nospieduma sensors *Touch ID*. *Touch ID* ir praktiskāks par *passcode*, jo to var ātrāk burtu un ciparu kombināciju kodam, un ir drošāks, ja lietotājs izmanto 4 ciparu koda bloķēšanu.

Lai izmantotu *Touch ID*, lietotājam ir nepieciešams iestatīt savu ierīci, lai tā izmantotu kodu bloķēšanu. Attēlā 6.7 ir redzama sadaļa lietotnē *„Settings"*, kurā ir iespējams uzstādīt, lai ierīce tiek atbloķēta, izmantojot pirksta nospiedumu.

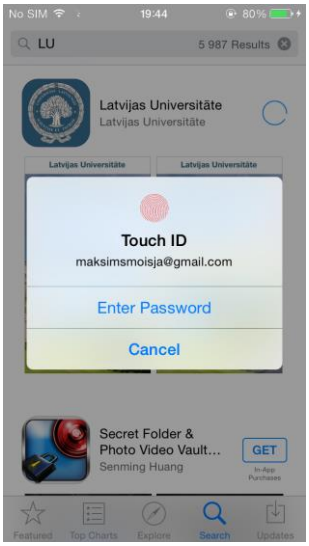
6.7. att. ***Touch ID*** **uzstādīšana**

Kad *Touch ID* sensors noskenē un atpazīst pirksta nospiedumu, ierīce tiek atbloķēta, nepieprasot atbloķēšanas kodu. Atbloķēšanas kods var tikt lietots arī *Touch ID* vietā, un tas vienmēr tiek pieprasīts šādos gadījumos:

- ierīce tiek ieslēgta vai restartēta;
- ierīce netika atbloķēta pēdējo 48 stundu laikā;
- ierīce ir saņēmusi bloķēšanas komandu;
- pēc 5 neveiksmīgām pirksta nospieduma nolasīšanas reizēm;
- kad tiek reģistrēti jauni pirksta nospiedumi ar *Touch ID* palīdzību.

Kad *Touch ID* opcija ir ieslēgta, ierīce uzreiz tiek bloķēta, kad gulēt/pamosties (*Sleep/Wake*) poga tiek nospiesta. Kad tikai koda bloķēšanas iespēja ir aktivizēta, daudzi lietotāji uzstāda atbloķēšanas aizkaves laiku, lai nevajadzētu katru reizi ievadīt kodu, lai atbloķētu ierīci. Ar *Touch ID* opciju, ierīce katru reizi pieprasa pirksta nospiedumu — vai *passcode* — kad ierīce tiek pamodināta.

Touch ID var pielietot arī citos gadījumos. To ir iespējams uzstādīt, lai akceptētu pirkumus, izmantojot lietotnes *„iTunes Store", „App Store", „iBooks Store",* līdz ar to lietotājam nav jāievada *Apple ID* parole. Attēlā 6.8 var redzēt dialoglodziņu (*DialogBox*), kurā lietotājam tiek piedāvāts apstiprināt lietotnes lejupielādi, izmantojot pirksta nospiedumu. Protams, to var veikt arī, ievadot paroli.

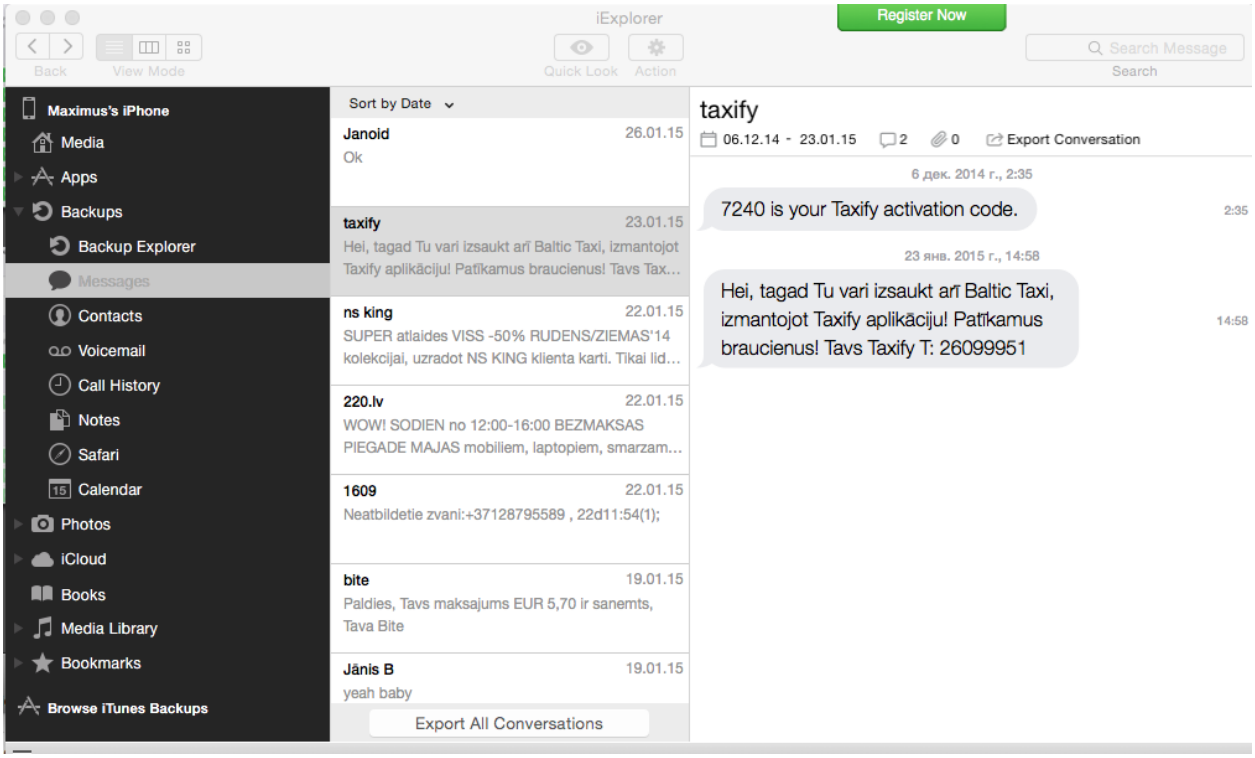


6.8. att. **Pakalpojuma iegādes apstiprinājuma dialoglodziņš**

Touch ID var izmantot kopā ar *Apple Pay* pakalpojumu (*iPhone 6* un *iPhone 6+*), kas ļauj veikt maksājumus, izmantojot savu telefonu. Diemžēl autors nespēj to nodemonstrēt, jo uz darba rakstīšanas brīdi Latvijā nav pirkumu terminālu, kas atbalstītu *Apple Pay* pakalpojumu, ar kuru varu izmantot *Touch ID*, lai veiktu pirkumu.

Programma *„iTunes"* palīdz veikt datu sinhronizāciju ar *iOS* ierīci. Pie tam var uzstādīt, ka

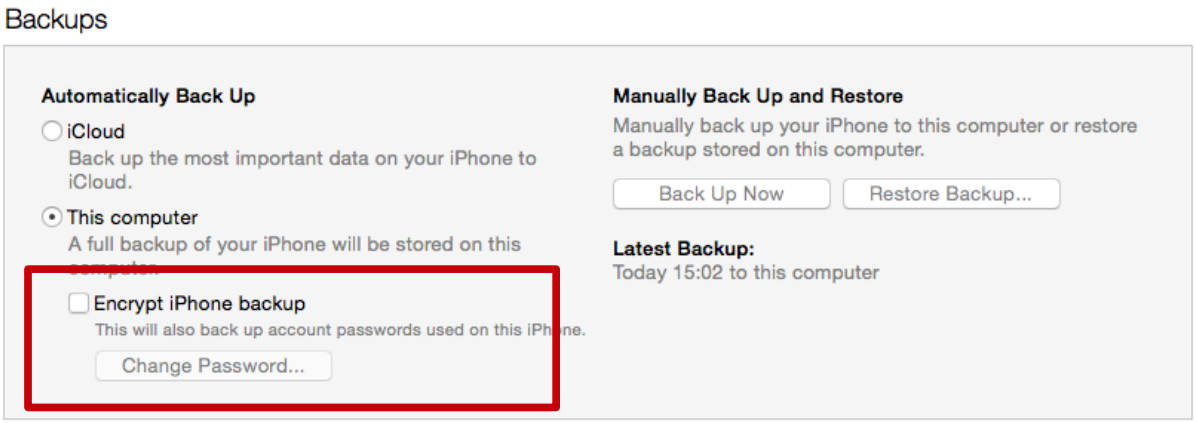
katru reizi, kad *iOS* ierīce tiek pieslēgta datoram, *„iTunes”* veic rezerves kopiju. Pēc noklusējuma rezerves kopija netiek šifrēta. Līdz ar to var iegūt visus lietotāja datus, tāpēc ieteicams rezerves kopiju šifrēt.



6.9 att. **Rezerves kopijas apskats**

Attēlā 6.9 var redzēt, ka ir iespējams iegūt īsziņu sarakstes (*SMS – Short Message Service*) vēsturi no *iOS* ierīces rezerves kopijas, izmantojot programmu *„iExplorer”*. Tāpat ir iespējams apskatīt zvanu vēsturi, kontaktus, pierakstus.

Lai rezerves kopija tiktu šifrēta, programmā *„iTunes”* ir jāizvēlas opcija *„Encrypt iPhone backup”* un ir jāuzstāda parole, kā tas ir atzīmēts attēlā 6.10.



6.10. att. **Rezerves kopijas šifrēšana**

NOBEIGUMS

Maģistra darba ietvaros darba autors ir iepazinies ar *iOS* operētājsistēmas piedāvātiem drošības līdzekļiem. Autors ir izpētījis ievainojamības, kuras ir saistītas ar pašu operētājsistēmu, lietotnēm un trešo pušu bibliotēku, ko izstrādātāji izmanto. Praktiskās daļas ietvaros darba autors veica lietotņu testēšanu, lai pārbaudītu, vai tās nodrošina lietotāja datu drošību. Maģistra darbā ir aprakstīti veidi, kā lietotņu izstrādātājs un ierīces lietotājs var pasargāt savus datus.

Autors maģistra darba laikā ir veicis secinājumus:

- *Apple* nodrošina ierīču ar *iOS* operētājsistēmu drošību aparātprogrammatūras, aparatūras un programmatūras līmenī;
- *iOS* lietotņu integritāte tiek pārbaudīts vairākas reizes;
- *iOS* sistēma piedāvā daudzas iespējas, kā izstrādātājs var realizēt savu un lietotāju datu drošību;
- *iOS* sistēma izmanto jaunākos un modernākos tīkla protokolus un standartus, lai nodrošinātu lietotāju datu drošību datu pārraides laikā;
- telefoni *iPhone* un *iOS* sistēma piedāvā vairākas iespējas lietotājam, kā kontrolēt savu datu drošību;
- ja lietotnes izstrādes laikā tiek izmantota trešās puses bibliotēka, ir jāveic tās testēšana;
- lietotne „*WhatsUp*" lietotāja saraksti glabā nešifrētu, līdz ar to ir diezgan vienkārši to iegūt;
- lietotnei „Latvijas Universitāte" ir atrastas vairākas kļūdas, kuras ir saistītas ar datu drošību;
- lietotnei „Draugiem" tika veikta metodes apmaiņa, kuras rezultāta vai iegūt lietotāja autorizācijas datus un var izraisīt *DoS* uzbrukumu;
- *keychain* glabātuves testēšanas laikā tika atklāts, ka lielāka daļa lietotņu lietotājvārdus un paroles glabā nešifrētus;
- ar programmas *„burp"* palīdzību var pārtvert datus, ko lietotne pārraida serverim.

Izstrādājot maģistra darbu, galvenā darba autora atziņa ir tāda, ka nozīmīgākais pasākums drošības garantēšanai lietotnēm, ir pārbaude, vai *iOS* ierīces ir veikts *jailbreak* process.

IZMANTOTĀ LITERATŪRA

[1] „Common Vulnerability Scoring System" apraksts – [tiešsaiste] – [atsauce, skatīts 28.04.2015]. Pieejams https://nvd.nist.gov/cvss.cfm.

[2] Termina *DoS* skaidrojums – [tiešsaiste] – [atsauce, skatīts 30.04.2015]. Pieejams http://termini.lza.lv/term.php?term=dos&list=&lang=EN&h=yes.

[3] Termina *Man-in-the-Middle* skaidrojums – [tiešsaiste] – [atsauce, skatīts 30.04.2015]. Pieejams https://www.owasp.org/index.php/Man-in-the-middle_attack.

[4] Termina *RADIUS* skaidrojums – [tiešsaiste] – [atsauce, skatīts 27.01.2015]. Pieejams http://www.gnu.org/software/radius/.

[5] Termina *WPA 2 Enterprise* skaidrojums – [tiešsaiste] – [atsauce, skatīts 29.04.2015]. Pieejams http://www.webopedia.com/TERM/W/WPA2.html.

[6] Termina *Zero-day* skaidrojums – [tiešsaiste] – [atsauce, skatīts 30.04.2015]. Pieejams http://www.pctools.com/security-news/zero-day-vulnerability/.

[7] *iOS* drošības arhitektūras apraksts – [tiešsaiste] – [atsauce, skatīts 28.01.2015]. Pieejams https://www.apple.com/business/docs/iOS_Security_Guide.pdf.

[8] *Cultofmac* raksts par *jailbreak* vēsturi [tiešsaiste] – [atsauce, skatīts 26.01.2015]. Pieejams http://www.cultofmac.com/192850/the-history-of-jailbreaking-feature/.

[9] *Medium* raksts par *jailbreak* vēsturi – [tiešsaiste] – [atsauce, skatīts 26.01.2015]. Pieejams https://medium.com/@dannykey/the-history-of-ios-jailbreaking-d1a42f48e462.

[10] *Apple* raksts par *jailbreak* procesa iespējamām sekām – [tiešsaiste] – [atsauce, skatīts 26.01.2015]. Pieejams https://support.apple.com/en-us/HT201954.

[11] Raksts par *No iOS Zone* ievainojamību – [tiešsaiste] – [atsauce, skatīts 30.04.2015]. Pieejams http://thehackernews.com/2015/04/security-researchers-have-uncovered.html.

[12] Video demonstrācija *No iOS Zone* ievainojamībai – [tiešsaiste] – [atsauce, skatīts 30.04.2015]. Pieejams https://youtu.be/PmgI0LaFYLA.

[13] Raksts par *Masque attack* ievainojamību – [tiešsaiste] – [atsauce, skatīts 27.04.2015]. Pieejams http://www.macrumors.com/2014/11/10/masque-attack-ios-vulnerability/.

[14] Video demonstrācija *Masque attack* ievainojamībai – [tiešsaiste] – [atsauce, skatīts 27.04.2015]. Pieejams http://youtu.be/3VEQ-bJUhPw.

[15] Priekšfona novērošanas ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 27.04.2015]. Pieejams http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-1276.

[16] Raksts par ievainojamību *AFNetworking v2.5.1* bibliotēkā – [tiešsaiste] – [atsauce, skatīts 20.04.2015]. Pieejams http://arstechnica.com/security/2015/04/1500-ios-apps-have-https-crippling-bug-is-one-of-them-on-your-device/.

[17] Raksts tīmekļa dienasgrāmatā par *AFNetworking* ievainojamību – [tiešsaiste] – [atsauce,

skatīts 20.04.2015]. Pieejams http://blog.mindedsecurity.com/2015/03/ssl-mitm-attack-in-afnetworking-251-do.html.

[18] Kompānijas *SourceDNA* raksts par *AFNetworking* ievainojamību – [tiešsaiste] – [atsauce, skatīts 20.04.2015]. Pieejams http://sourcedna.com/blog/20150420/afnetworking-vulnerability.html.

[19] Pārbaude, vai lietotnēm ir *AFNetworking* ievainojamība – [tiešsaiste] – [atsauce, skatīts 20.04.2015]. Pieejams http://searchlight.sourcedna.com/lookup.

[20] Raksts par ievainojamību bibliotēkā *AFnetworking v2.5.2* bibliotēkā – [tiešsaiste] – [atsauce, skatīts 22.04.2015]. Pieejams http://arstechnica.com/security/2015/04/critical-https-bug-may-open-25000-ios-apps-to-eavesdropping-attacks/.

[21] Lietotnes *WiFi Drive Pro v1.2 iOS* ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 15.04.2015]. Pieejams https://www.exploit-db.com/exploits/36795/

[22] Lietotnes *Photo Manager Pro 4.4.0 iOS* ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 15.04.2015]. Pieejams https://www.exploit-db.com/exploits/36798/.

[23] Lietotnes *Folder Plus 2.5.1 iOS* ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 14.04.2015]. Pieejams https://www.exploit-db.com/exploits/35083/.

[24] Lietotnes *Grindr 2.1. iOS* mājaslapa – [tiešsaiste] – [atsauce, skatīts 14.04.2015]. Pieejams http://grindr.com/learn-more.

[25] Lietotnes *Grindr 2.1. iOS* ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 14.04.2015]. Pieejams https://www.exploit-db.com/exploits/36903/.

[26] Lietotnes *PayPal Inc iOS 4.6* ievainojamības apraksts – [tiešsaiste] – [atsauce, skatīts 13.04.2015]. Pieejams https://www.exploit-db.com/exploits/34957/.

[27] Drošības pārbaudes shēma – [tiešsaiste] – [atsauce, skatīts 03.05.2015]. Pieejams https://www.owasp.org/index.php/IOS_Application_Security_Testing_Cheat_Sheet.

[28] Drošības testēšanas vadlīnijas – [tiešsaiste] – [atsauce, skatīts 03.05.2015]. Pieejams https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Security_Testing_Guide.

[29] *Jailbreak* procesa instrukcijas – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams http://v2.guidemyjailbreak.com/iphone/jailbreak-iphone/iphone-5/how-to-jailbreak-iphone-5-ios-7-1-2-using-pangu/.

[30] Nepieciešamie faili *USB* slēguma izmantošanai – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams https://code.google.com/p/iphone-dataprotection/source/browse/usbmuxd-python-client/?r=3e6e6f047d7314e41dcc143ad52c67d3ee8c0859#usbmuxd-python-client%253Fstate%253Dclosed.

[31] Programma *class-dump-z* – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams

https://code.google.com/p/networkpx/wiki/class_dump_z.

[32] Programma *clutch* – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams https://github.com/KJCracks/Clutch/releases/tag/1.4.6.

[33] Programma *burp suite* – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams http://portswigger.net/burp/downloadfree.html.

[34] Programma *keychain-dumper* – [tiešsaiste] – [atsauce, skatīts 04.05.2015]. Pieejams https://github.com/ptoomey3/Keychain-Dumper.

[35] Failu saraksts, kas parādās pēc *jailbreak* – [tiešsaiste] – [atsauce, skatīts 01.05.2015]. Pieejams https://www.trustwave.com/Resources/SpiderLabs-Blog/Jailbreak-Detection-Methods/.

DOKUMENTĀRĀS LAPAS FORMA

Maģistra darbs: **Mobilo lietotņu datu drošība *iOS* vidē**

Ar savu parakstu apliecinu, ka pētījums veikts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: _____
(Autora paraksts)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____
(Vadītāja paraksts)

Darbs iesniegts maģistrantūras sekretariātā _____.
(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.
(Metodiķes paraksts)

Recenzents: _____
(Akad. amats, zin. grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____
(Darba aizstāvēšanas datums)

Komisijas sekretārs: _____
(Sekretāra paraksts)