Università degli studi di Verona

Flicli

Andrea Colato
Tommaso Bonetti

progetto per il corso di

Verifica automatica dei sistemi

7 agosto 2017

Indice

Capitolo 1

Introduzione

Il corso di *Verifica automatica di sistemi* dell'Università degli studi di Verona tenuto dalla Prof.re Nicola Fausto Spoto ha come scopo la presentazione della programmazione e verifica di sistemi software, con particolare attenzione agli aspetti di concorrenza e mobilità. Vengono inizialmente presentati i costrutti sintattici, la loro semantica e gli strumenti per la programmazione concorrente, i problemi conseguenti e le possibilità di verifica di correttezza tramite strumenti automatici. Viene quindi presentata la programmazione di servizi web concorrenti basati su servlet. Infine viene descritta la programmazione di software mobile, tipicamente accoppiato a dei servizi web, in cui la concorrenza ha un utilizzo ormai sempre più esteso. Il corso utilizza il linguaggio Java e il framework Android come contesti concreti in cui sviluppare le soluzioni software.

Questo progetto è stato svolto con l'obiettivo di soddisfare i requisiti richiesti dal Prof.re come soglia di superamento dell'esame, è stata successivamente una nostra volontà quella di scrivere questa relazione dando particolare enfasi alla Sicurezza Informatica.

Viene quindi definita in generale la sicurezza informatica e i tre fondamentali pilastri sulla quale si basano tutte le sue generalizzazioni. Successivamente si introduce lo sviluppo dell'applicazione e le metodologie adottate descrivendone i dettagli. Concludendo viene fatta una panoramica dello stato dell'arte della sicurezza delle moderne applicazioni Android.

Capitolo 2

Sicurezza Informatica

Un tempo la *sicurezza*, o meglio *riservatezza* delle informazioni, riguardava principalmente il settore militare: per migliaia di anni sovrani e generali hanno avuto il bisogno di comunicazioni efficienti per governare i loro paesi e comandare i loro eserciti, consapevoli delle conseguenze che avrebbe avuto la caduta dei loro messaggi in mani ostili.

Nel V secolo a.C. era ben sviluppata la «steganografia», l'arte cioè di "coprire la scrittura". Gli spartani, ad esempio, inviavano gli ordini ai capi militari tramite messaggi scritti su una striscia di cuoio che, avvolta su un bastone (lo scitale) di un diametro ben preciso, permetteva di leggere il testo in chiaro lungo il bastone. Erodoto nelle "Storie" narra della pratica inventata da Istieo per comunicare l'ordine di ribellione ad Aristagora, di rasare la testa del più fidato degli schiavi, per incidervi poi il messaggio ed inviarlo al destinatario dopo che fossero ricresciuti i capelli (libro V) o ancora la tecnica di Demarato di raschiare la cera da una tavoletta doppia, incidervi il messaggio da recapitare e ricoprirlo nuovamente di cera (libro VII):

Quando Serse decise la campagna contro l'Eliade, Demarato, che era a Susa e che era stato informato, volle darne l'annunzio ai Lacedemoni. E non potendo comunicare diversamente, per il rischio di essere colto in fallo, ricorse a siffatto espediente. Prese una doppia tavoletta, ne raschiò la cera, e poi scrisse sul legno della tavoletta il piano del Re; e, ciò fatto, tornò a versare la cera sullo scritto, perché la tavoletta liscia non recasse al portatore nessuna noia da parte dei custodi delle strade. Ma quando essa giunse a Lacedemone, i Lacedemoni non sapevano raccapezzarsi, prima ricevere, come mi riferisce, il suggerimento della figlia di Cleomene e moglie di Leonida. Ella intuì; e invitò a raschiare la cera, asserendo che avrebbero trovato lo scritto sul legno. Fu ascoltata, e cosi trovarono lo scritto; lo lessero e poi lo inviarono agli altri Elleni. Così si dice che i fatti si siano svolti.

"Storie" di Erodoto

Da Plinio il Vecchio (I secolo d.C.) ad Umberto Eco ("Il nome della rosa"), la letteratura è disseminata di esempi di scrittura a base di limone o latice di titimabo, che appaiono invisibili, ma ricompaiono una volta che il testo venga esposto a una fonte di calore (cd. inchiostro simpatico).

Tuttavia il punto debole di quest'arte, la possibilità cioè di scoprire il messaggio, favorì lo sviluppo e l'evoluzione della crittografia, non solo scienza ma vera e propria arte di "nascondere il messaggio", più precisamente il suo significato rendendolo incomprensibile secondo un procedimento preconcordato dal mittente e dal destinatario. Si narra che lo stesso Giulio Cesare fosse solito cifrare i propri messaggi sostituendo ogni lettera con quella che nell'alfabeto segue di qualche posizione(Cifrario di Cesare). Dall'antico metodo utilizzato da Cesare, l'innovazione e la tecnologia hanno portato allo sviluppo di tecniche sempre più sofisticate per garantire la sicurezza nella tecnologia stessa.

Con l'avvento della società dell'informazione, basata cioè sull'uso delle informazioni come parte integrante delle attività umane, la necessità di sicurezza delle informazioni è diventata una componente della sicurezza dei beni in generale, o security, e non si limita più alle tecniche per nascondere il contenuto dei messaggi.

2.1 Cosa proteggere

Se, sotto quest'aspetto di digitalizzazione, i crittosistemi classici (disco cifrante, enigma) sono stati ormai soppiantati da un cifrario a doppia chiave di sicurezza, pubblica e privata, la sicurezza informatica si snoda su più livelli di protezione dagli attacchi informatici: a livello fisico e materiale, ponendo gli strumenti (pc, server) in luoghi sicuri possibilmente dotati di sorveglianza e di controllo degli accessi; a livello immateriale attraverso un sistema di autorizzazioni per l'accesso degli utenti.

Abbiamo quindi sette principali categorie dove possiamo contestualizzare la sicurezza informatica rispetto all'ambiente dove questa è chiamata a operare:

Hardware
Di questa categoria fanno parte tutte le apparecchiature fisiche come server, dischi, infrastrutture, ecc.

Software
Il Sistema operativo e i suoi applicativi, software, servizi in cloud, ecc.

Dati
Informazioni presenti sul filesystem, database, backup, ecc.

Reti
Tecnologie di trasporto dell'informazione, collegamenti e apparati.

Accessi
Gli strumenti che vengono forniti ai soggetti per accedere alle risorse come password, generatori di token, ecc.

Individui chiave
Questa categoria fa riferimento agli amministratori di sistema o eventuali operatori specializzati.

2.2 Come proteggerlo

Il problema della sicurezza dei sistemi informatici, a differenza della sicurezza delle informazioni, risale ai primi anni '60, quando vengono progettati i primi meccanismi per "garantire" la sicurezza informatica degli stessi. Sempre in questi anni vengono riportati i primi casi di frode informatica e appaiono su riviste specializzate i primi contributi scientifici che mirano a definire con precisione il problema della sicurezza informatica e ad individuarne possibili soluzioni. Dobbiamo però attendere la fine degli anni '80 per avere una definizione universalmente accettata di sistema di calcolo sicuro. Tale definizione è riportata nel manuale ITSEC e può essere così sintetizzata: *"Un sistema di calcolo viene considerato sicuro quando è in grado di garantire il soddisfacimento delle proprietà di confidenzialità, integrità e disponibilità".*

Confidenzialità

Un sistema raggiunge gli obiettivi di sicurezza prefissati quando i dati non sono accessibili o comunque non interpretabili dai non aventi diritto. Ciò significa che, anche se i dati dovessero essere intercettati, la loro lettura deve risultare impossibile o, per essere più realisti, eccessivamente complessa. Quindi nessun utente (o attaccante) deve poter ottenere o dedurre dal sistema informazioni che non è autorizzato a conoscere.

Si ottiene la confidenzialità nascondendo ai non autorizzati le informazioni, tramite l'utilizzo di strumenti crittografici, e le risorse, sviluppando metodologie di controllo degli accessi.

Inoltre la proprietà di confidenzialità deve essere in grado di garantire, a prescindere da dove questa venga applicata, le caratteristiche di:

- **Riservatezza dei dati**
  Le informazioni confidenziali non devono essere rivelate o rilevabili da utenti non autorizzati.
- **Privacy**
  L'utente controlla o influenza quali informazioni possono essere collezionate e memorizzate.

In un sistema che garantisce realmente la confidenzialità, una terza parte che dovesse entrare in possesso delle informazioni scambiate tra mittente e destinatario, non dovrebbe essere in grado di ricavarne alcun contenuto informativo intelligibile. Nonostante vi siano precauzioni come i meccanismi precedentemente citati è bene ricordare che non esistono meccanismi di protezione sicuri in assoluto.

Integrità

In un'ottica generale di sicurezza non deve essere possibile alterare i dati oggetto di una qualsivoglia transazione. Questo fattore riveste particolare importanza, sopratutto se si fa riferimento ad operazioni come la contrattualistica digitale o le transazioni economiche.

Questa proprietà deve quindi impedire l'alterazione diretta o indiretta delle informazioni sia da parte di utenti e processi non autorizzati, che a seguito di eventi accidentali. Se i dati vengono alterati è necessario fornire strumenti per poterlo verificare. L'integrità è possibile classificarla in due modi, a seconda della risorsa dove questa deve essere applicata:

- **Integrità dei dati**
  Le informazioni e i programmi possono essere modificati solo se autorizzati.
- **Integrità del sistema**
  Il sistema funziona e non è compromesso

L'integrità dei dati scambiati è molto spesso associata all'autenticazione di questi visto che molti protocolli o meccanismi o algoritmi crittografici di tipo simmetrico o asimmetrico (es. MAC e firma digitale nei documenti digitali) sono in grado di assicurare entrambi i requisiti. Molti protocolli di comunicazione all'interno dello stack protocollare di rete, assicurano il controllo sull'integrità dei dati scambiati in una comunicazione attraverso un opportuno campo checksum contenuto nell'intestazione (header) del pacchetto.

Disponibilità

Rendere disponibili a ciascun utente abilitato le informazioni alle quali ha diritto di accedere, nei tempi e nei modi previsti.

Per disponibilità si intende semplicemente la possibilità di utilizzare una determinata risorsa o un'informazione nel tempo, alcuni attacchi come il classico DDoS, tendono a diminuire e/o annullare la disponibilità di alcune risorse.

Requisiti di disponibilità:

- **Prestazioni**
  Misura in cui il sistema utilizza risorse hardware.
- **Robustezza**
  Misura in cui il sistema si comporta in situazioni impreviste.

Il requisito di disponibilità viene soddisfatto tramite l'implementazione di tecnologie che sono in grado di incidere sui fattori di scalabilità e affidabilità dell'architettura. Un esempio ne è l'implementazione del meccanismo di "load balancing", che consiste nel distribuire il carico di elaborazione di uno specifico servizio, ad esempio le richieste di un sito web, tra più server evitando i sovraccarichi al sistema.

2.3 Minacce e attacchi

La sicurezza informatica ha lo scopo di studiare le minacce e le vulnerabilità. Questi due termini non sono assolutamente dei sinonimi ma hanno significati ben diversi. Una **minaccia** è una potenziale causa di un incidente (accidentale o volontario) che può danneggiare più elementi che costituiscono il patrimonio informativo. Una **Vulnerabilità** invece è una debolezza presente nel sistema operativo, nelle procedure di sicurezza, nei controlli interni o nell'implementazione. In base a queste due entità vengono verificate e implementate le politiche e i meccanismi di sicurezza.

Politiche di sicurezza

Una politica di sicurezza è un'indicazione di cosa è e cosa non è permesso. Queste indicazioni possono riguardare operazioni che si possono usare su certi dati, gli utenti che possono usufruire di queste operazioni e eventuali profili di utenti con specifici diritti.

Meccanismi di sicurezza

Un meccanismo di sicurezza è un metodo (strumento o procedura) per garantire una politica di sicurezza. Data una politica di sicurezza che distingue le azioni "sicure" da quelle "non sicure", i meccanismi di sicurezza devono **prevenire**, **scoprire** o **recuperare** un attacco.

La **prevenzione** significa che il meccanismo deve rendere impossibile l'attacco. È difficile trovare un compromesso tra usabilità e prevenzione perché spesso questi meccanismi sono pesanti ed interferiscono con il sistema al punto di renderlo scomodo da usare. La **scoperta** significa che il meccanismo è in grado di scoprire che un attacco è in corso. È utile quando non è possibile prevenire l'attacco, ma può servire anche a valutare le misure preventive. Si usa solitamente un monitoraggio delle risorse del sistema, cercando eventuale tracce di attacchi. Il **recupero** da un attacco invece è possibile farlo in due modi. Il primo è fermare l'attacco e recuperare la situazione pre-attacco. Il secondo invece è continuare a far funzionare il sistema correttamente durante l'attacco (fault-tolerant).

Capitolo 3

Flicli

L'applicazione Android Flicli è un client Flickr che permettere di effettuare le seguenti operazioni:

- Ricerca delle immagini partendo da una stringa ricevuta come input;
- Ricerca delle ultime immagini caricate;
- Ricerca delle immagini più popolari;

Il risultato viene visualizzato in una lista di titoli di immagini, con alla loro sinistra una preview dell'immagine (75x75 pixel). Ciascun elemento della lista dovrà reagire ai seguenti eventi:

- **Click semplice:** viene visualizzata l'immagine ad alta risoluzione, con sotto gli ultimi commenti inseriti per quell'immagine;
- **Click lungo:** Si aprirà un menu contestuale che permetterà di effettuare le seguenti due operazioni:
  - condividere l'immagine ad alta risoluzione con altre app disponibili alla condivisione (es: WhatsApp, Telegram, Gmail);
  - cercare le ultime immagini caricate dallo stesso autore;

Inoltre la funzione di condivisione di un'immagine è accessibile anche da menu nell'action bar, se si sta già visualizzando l'immagine ad alta risoluzione. Il layout è stato reso responsive, utilizzando l'approccio master/detail. Su un telefono si visualizzano alternativamente i frammenti per ricerca, lista delle immagini e singola immagine ad alta risoluzione. Su un tablet invece, il frammento della ricerca dovrà essere sempre visibile, mentre i frammenti della lista delle immagini e della singola immagine ad alta risoluzione dovranno essere mostrati in alternativa fra di loro.

Infine il menu dell'applicazione, oltre alla condivisione dell'immagine attualmente mostrata, deve includere una voce per mostrare data, versione e autore dell'applicazione.

Il tutto è stato sviluppato utilizzando il paradigma MVC.

3.1 Versioning

Il controllo di versione è un sistema che registra, nel tempo, i cambiamenti ad un file o ad una serie di file, così da poter richiamare una specifica versione in un secondo momento.

Un VCS(Version Control System - VCS) ti permette di ripristinare i file ad una versione precedente, ripristinare l'intero progetto a uno stato precedente, revisionare le modifiche fatte nel tempo, vedere chi ha cambiato qualcosa che può aver causato un problema, chi ha introdotto un problema e quando, e molto altro ancora. Usare un VCS significa anche che se fai un pasticcio o perdi qualche file, puoi facilmente recuperare la situazione.

3.1.1 Git

Git è un software di controllo versione distribuito utilizzabile da interfaccia a riga di comando, creato da Linus Torvalds nel 2005.

Wikipedia, Git

Abbiamo sviluppato l'applicazione Flicli utilizzando il sistema di versiong Git ospitando l'applicazione sul repository privato `https://github.com/jekoA/flicli`. Di seguito viene mostrato un estratto dei vari contributi apportati al Repository durante la fase di sviluppo:

commit b60fcd76e413e8c27c5daf4788cfda14226cffdd Merge: 454529f b704b6b Author: Andrea Colato <andreacolato94@gmail.com> Date: Mon Aug 7 21:58:03 2017 +0200
Merge remote-tracking branch 'origin/master'

commit 454529f5a715201eabda653d08cc931df388bd3b Author: Andrea Colato <andreacolato94@gmail.com>
Date: Mon Aug 7 21:57:50 2017 +0200
Gold plating, ended of coding

commit b704b6ba8154f097096bd21ae52547e9c7aaae42 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Sun Jul 30 23:25:23 2017 +0200
fix bugs

commit 84a6e27062d78b4af561846179e875ecf98364dc Author: Andrea Colato <andreacolato94@gmail.com>
Date: Sun Jul 30 12:51:32 2017 +0200
Refactor *Adapter, Used Gridview for listing photos

...

commit 6e57b1744fe4a9ed2256b8322c9f2d1ddb9688a8 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Fri Jun 2 10:57:46 2017 +0200
Add FlicliService

commit fe5d9f6173aaf4546c6064ee7a323af97264d3d3 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Mon May 29 21:55:20 2017 +0200
Add MultiDevice portation using master/detail

commit 63fd0a6db9215d5123409a36b827217468e17d82 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Sun May 28 21:45:54 2017 +0200
Add StartFragment Add sharedMenu and startMenu Separate Application in AppCompactActivity
e Application

commit e23e0323ef00118f64eb497f0a31b75f47e0db09 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Fri May 19 21:20:50 2017 +0200
Add Immutable class to manadge photos and comments

commit 6a352c38a8acf04863c6b4df37b57cb649107039 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Fri May 19 21:00:41 2017 +0200
    Add XML view file and resource Image

commit 8470239a02a19ff1b62b8c6f696d4fc48001d081 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Thu May 18 00:09:29 2017 +0200
    Add MVC pattern

commit 8f6831ed8455d818a20fa30441290c94d3fc99d0 Author: tommi.bonetti <tommi.bonetti@gmail.com>
Date: Wed May 17 23:57:33 2017 +0200
    Add JackOptions, Add menu directory and multi-language support

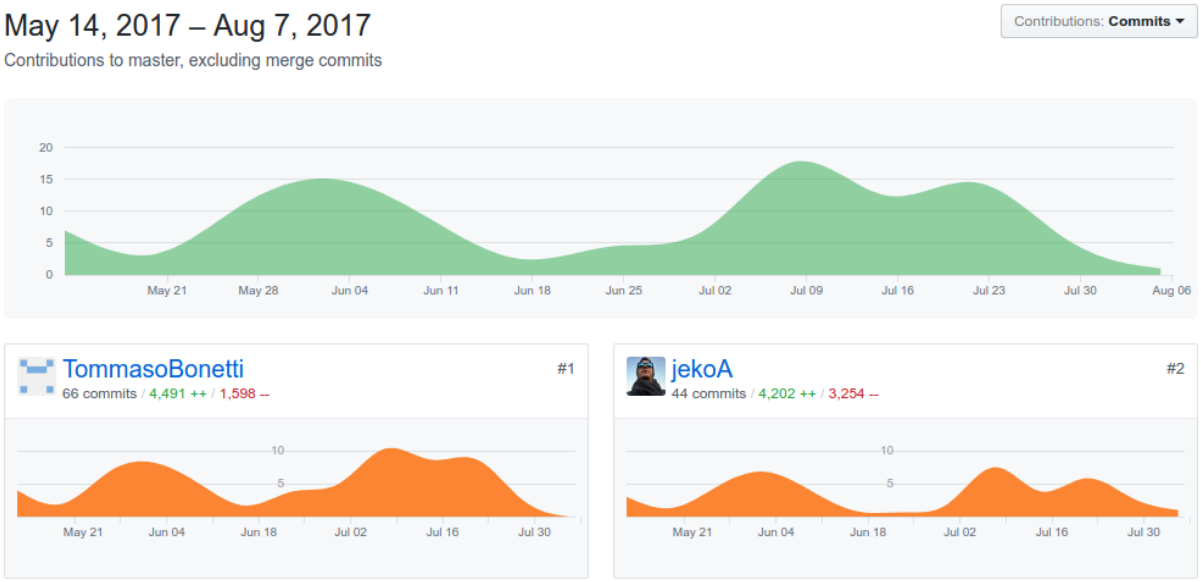
commit 1d6be57c2f0810c5599d916611522230d9e0d878 Author: Andrea Colato <andreacolato94@gmail.com>
Date: Wed May 17 21:07:07 2017 +0200
    Tryng with private repo

commit 2ac9cefe9650cb571d765fff8b1484a6c6197f84 Author: Andrea Colato <andreacolato94@gmail.com>
Date: Wed May 17 21:01:02 2017 +0200
    Setting Android Sutdio Commit

commit b75e56ca0a0fd47f7ead99cbf18d94f0678e133e Author: Andrea Colato <andreacolato94@gmail.com>
Date: Wed May 17 20:59:10 2017 +0200
    Initial commit

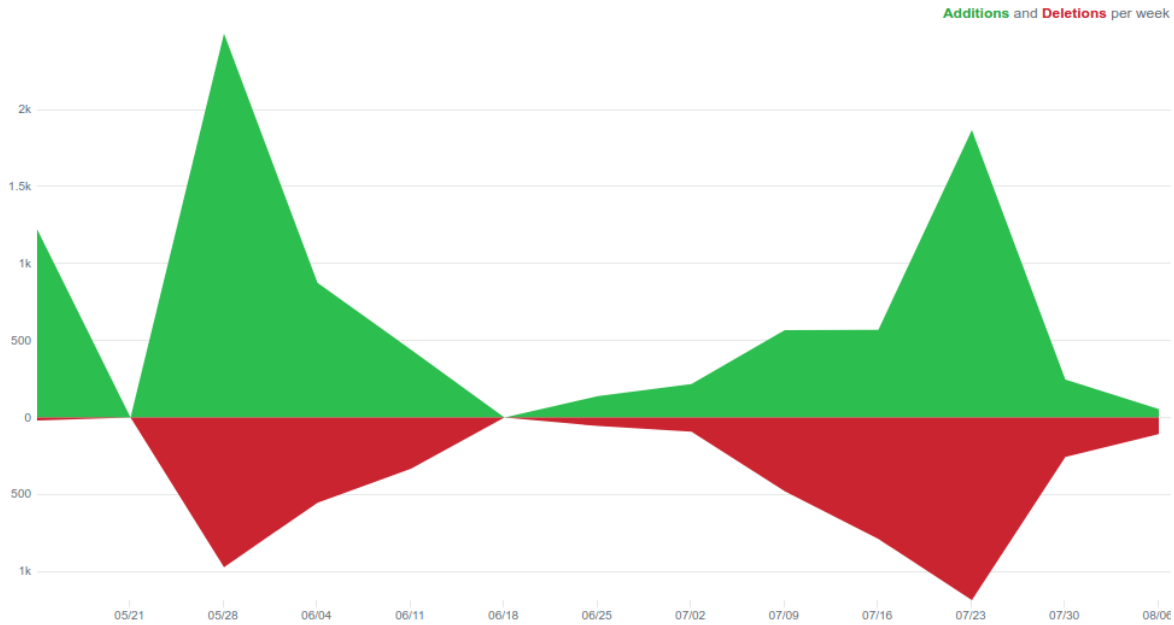
3.1.2 Sviluppo

Volendo trarre qualche considerazione dalla frequenza dei vari contributi possiamo notare subito la durata del progetto. Flicli ha avuto inizio il 17 Maggio ed è terminato con la stesura di questa relazione. Di seguito sono riportati due grafici che indicano la distribuzione dei contirbuti su base temporale e la "Code Frequency" ossia la relazione tra codice aggiunto e codice rimosso.



Distribuzione dei Commit

Con il grafico successivo di "Code Frequency" si potrà notare che durante lo sviluppo del progetto è stata rimossa una notevole quantità di codice. Questo comportamento è evidenza del fatto che che non ci siamo limitati a scrivere un'app che funzionasse ma abbiamo cercato di migliorare il codice scrivendolo sempre più pulito e comprensibile inserendo delle fasi di Refactoring del codice.



Frequenza del codice

Capitolo 4

Analisi Statica

Per verificare che un sistema rispetti le sue specifiche il metodo più usato è quello di eseguirlo e testarne il comportamento. Avere un problema di sicurezza per'o non significa che il software non rispetti le sue caratteristiche, piuttosto che siano presenti delle "funzionalit'a" non desiderate le quali possono portare a comportamenti non sicuri. Individuare e gestire queste problematiche aiuta a rendere un programma solido e ci sono vari metodi che ci aiutano a farlo.

Noi abbiamo utilizato il metodo dell'Analisi Staica con lo scopo di analizzare il codice alla ricerca di vulnerabilità o errate configurazione senza eseguirlo. Ci siamo avvalsi del tool Open Source Mobsf che ci ha aiutato ad automatizzare processo aiutandoci a identificare la fonte di problemi di sicurezza. Lo svantaggio principale dell'utilizzo questi tools è che non sempre sono precisi e perci'o si tratta di un tipo di analisi che non si pu'o ritenere completamente automatizzata. I risultati che forniscono vanno interpretati e non sempre evidenziano tutti i pericoli; non sostituiscono le persone ma rappresentano un valido aiuto per l'analisi del software

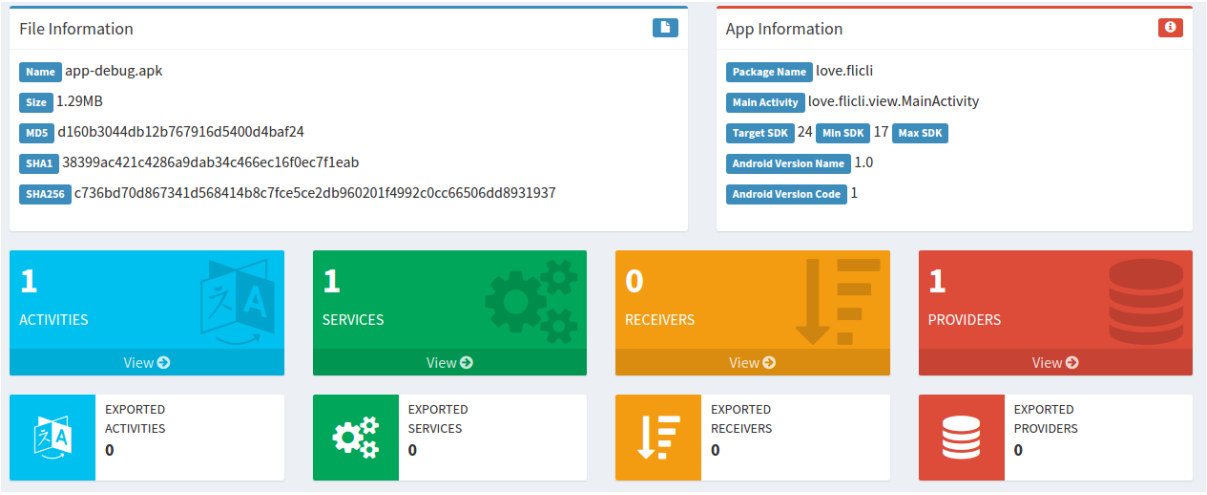
4.1 Mobsf Scan

Mobsfscan è un tool Open Source che permette di analizzare un'applicazione android attraverso le metodologie di analisi statica, analisi dinamica ed è inoltre un Web API fuzzer. Nel dettaglio la presentazione del tool è la seguente:

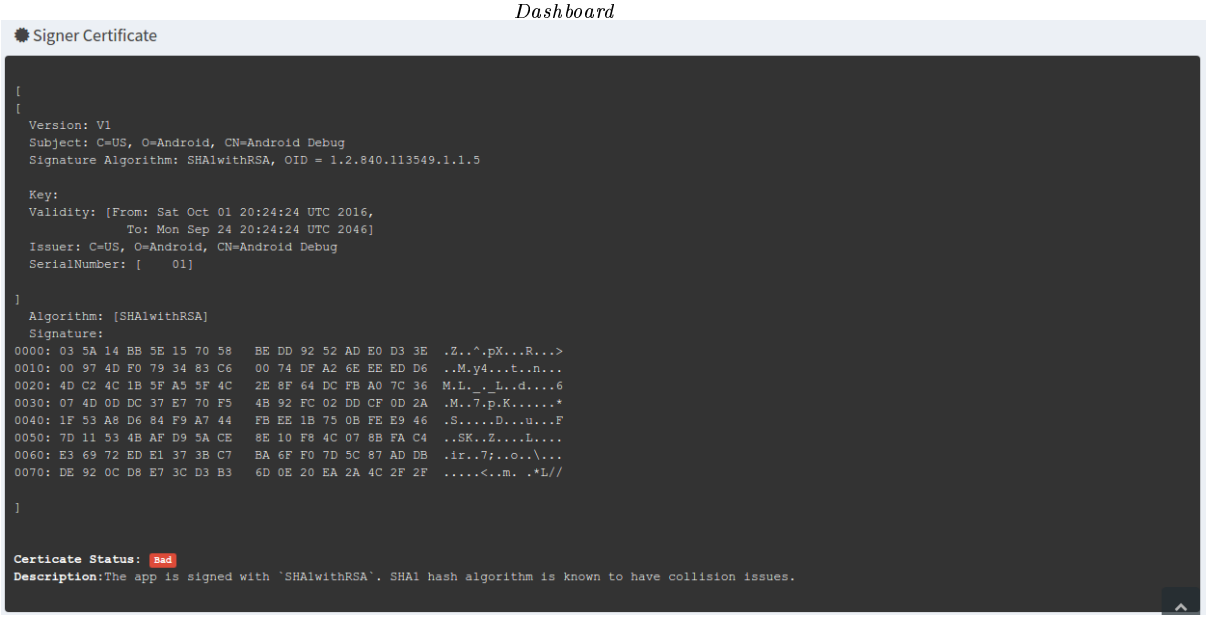
Mobile Security Framework (MobSF) is an intelligent, all-in-one open source mobile application (AndroidiOSWindows) automated pen-testing framework capable of performing static and dynamic analysis. It can be used for effective and fast security analysis of Android, iOS and Windows mobile Applications and supports both binaries (APK, IPA & APPX) and zipped source code. MobSF can also perform Web API Security testing with it's API Fuzzer that can do Information Gathering, analyze Security Headers, identify Mobile API specific vulnerabilities like XXE, SSRF, Path Traversal, IDOR, and other logical issues related to Session and API Rate Limiting.

https://github.com/MobSF/Mobile-Security-Framework-MobSF

Ci siamo limitati ad eseguire un'analisi statica dell'applicazione e di seguito sono riportate alcune schermate riassuntive del risultato dell'analisi.



Dashboard



Signer

Dato che l'applicazione sostanzialmente è un client che non implementa alcuna *Buisness Logic* visto che si interfaccia direttamente con le API ufficiali di FLick, non sono state trovate particolari vulnerabilità a fronte dell'analisi Statica effettuata. Vengono comunque segnalati i vari permessi che sono stati conferiti all'applicazione e viene segnalato che la console di debugging non è stata disabilitata.

Android Permissions

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.READ_EXTERNAL_STORAGE	dangerous	read SD card contents	Allows an application to read from SD Card.
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete SD card contents	Allows an application to write to the SD card.
android.permission.ACCESS_NETWORK_STATE	normal	view network status	Allows an application to view the status of all networks.
android.permission.INTERNET	dangerous	full Internet access	Allows an application to create network sockets.

Android Library Binary Analysis

ISSUE	SEVERITY	DESCRIPTION	FILES

DEX Malware Analysis

ANTI-VM	COMPILER	OBFUSCATOR	PACKER	DROPPER	MANIPULATOR	ANTI-ASSEMBLY	ABNORMAL PATTERN

Code Scans

Manifest Analysis

ISSUE	SEVERITY	DESCRIPTION
Debug Enabled For App [android:debuggable=true]	high	Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.
Application Data can be Backed up [android:allowBackup=true]	medium	This flag allows anyone to backup your application data via adb. It allows users who have enabled USB debugging to copy application data off of the device.

Manifest Scan

Capitolo 5

Conclusioni

Il quinto rapporto sulla sicurezza stilato dal Clusit racconta in numeri vulnerabilità e minacce, descrivendone l'evoluzione. Gli esperti sottolineano come stia diminuendo invece l'hacktivism il che significa che crescono gli attacchi sempre più mirati al business. La media di durata di un attacco è stimata attorno alle 18 ore. Negli ultimi 48 mesi, infatti, le perdite economiche sono aumentate di 4 volte (molto più del numero degli attacchi). In generale, i crimini informatici nel nostro Paese e nel mondo fanno registrare il numero di attacchi gravi più elevato degli ultimi 5 anni: 1012 solo quelli di dominio pubblico nel 2015 (contro gli 873 del 2014). A registrare l'incremento più alto è lo spionaggio industriale, che cresce di un 39%.

Il rapporto inoltre evidenzia come ci sia una preferenza, quella per Android, adottata anche dai cybercriminali che, in un'impennata di "creatività" senza pari che nel 2015 ci ha regalato quasi un milione di nuovi malware rispetto al biennio 2013/2014, hanno dato vita a oltre 2,3 milioni di nuovi ceppi di malware prodotti ai danni dell'utenza, frutto di una capacità evolutiva con cui le aziende non riescono a tenere il passo, e presa assolutamente sotto gamba dai produttori di device mobili.

Il fattore di vulnerabilità primario di Android è infatti che ad oggi oltre l'80% dei dispositivi presenti un firmware obsoleto, alla mercè dei cybercriminali (cfr. G DATA Mobile Malware Report Q3/2015), la stragrande maggioranza monta ancora Android 4.4 e inferiori.

L'insicurezza cibernetica a livello globale è cresciuta in modo significativo causata dal consistente aumento delle tipologie di aggressori assistendo ad un conseguente incremento della superficie di attacco esposta dalla nostra società digitale, sempre più iper-connessa, anche in conseguenza dell'aumento di tecnologie a basso costo, che sono intrinsecamente poco o per nulla sicure se confrontate con le capacità di nuocere degli avversari.

Viste le premesse, la sicurezza non può essere considerata un oggetto *plug and play*, una proprietà che si dimostra tramite operazioni algebriche o una certificazione che verifica la correttezza di una qualche politica aziendale. La sicurezza deve essere considerata come un processo che si acquisisce e si migliora attraverso una continua ricerca e cura dei dettagli, in ogni ambiente questa venga applicata. Proprio per questo è una caratteristica che deve sempre essere messa in discussione perché il rischio non può mai essere un fattore interamente estinguibile.

Bibliografia

[1] Erodito: `Storie`, Introduzione di Livio Rossetti, Traduzione di Piero Sgroj, Roma, 1nd edition, Giugno 2013.

[2] Introduzione storica: `http://www.disfipeq.unich.it/sites/st10/files/10._la_sicurezza_informatica.pdf`;

[3] Official Git documentation: `https://git-scm.com/`;

[4] Static Analyzer: `https://github.com/MobSF/Mobile-Security-Framework-MobSF/wiki/1.-Documentation`;

[5] Vincenzo Calabro, Slide sulla Sicurezza Informatica: `http://www.divini.net/maponi/si/SicurezzaInformatica.pdf`;

[6] Nicola Fausto Spoto, Verifica automatica dei sistemi: *Slide del corso di Verifica Automatica dei Sistemi*;

[7] Isabella Mastroeni, Analisi dei sistemi informatici: *Slide del corso di Analisi dei Sistemi Informatici*;