



Università degli Studi di Salerno
Dipartimento di Informatica

Esame di Reti Geografiche
Anno Accademico 2020/2021

Analisi di applicazioni di fitness per verificare Privacy Leakage

Studenti:

Simone Auriemma
Emmanuel Tesauero

Docente:

Delfina Malandrino

Indice

Introduzione.....	2
Strumenti utilizzati	4
Dispositivi utilizzati	5
Mitmproxy	5
Har_dump.py	8
Analyze.py	9
Problematiche.....	11
Burp Suite Community Edition	12
Apple Remote Virtual Infrastructure & Wireshark	12
Fiddler	13
Charles Proxy	14
Mitmproxy	14
Workflow	16
Risultati	18
Nike Run Club.....	19
MyFitnessPal	20
Adidas Running	21
Yazio	22
Fitness Buddy	23
MapMyRun	24
JEFIT Workout Planner.....	25
Runkeeper.....	27
8fit.....	29
Lifesum.....	30
Decathlon Coach.....	31
Mywellness	32
Grafico riassuntivo	33

Introduzione

Ad oggi gli smartphone sono nostri compagni costanti nella vita quotidiana. Sempre più utenti fanno affidamento su ogni genere di applicazione presente nei vari store per le normali attività giornaliere.

Quello a cui non si pensa è che, nonostante i molteplici benefici che una specifica applicazione possa offrire, ognuna di esse potrebbe avere accesso ad un insieme significativo di informazioni sull'utente.

Nonostante questo potenziale rischio, la maggior parte degli utenti sarebbe negligente quando si tratta di controllare le autorizzazioni di accesso delle app e tendono a concedere ampi diritti (spesso tutti) di accesso senza ulteriori controlli.

Tra le svariate categorie di applicazioni che è possibile trovare in rete, quelle relative al fitness o, più in generale, alla salute e al benessere dell'utente sono quelle con il più alto tasso di rischio di **Information Leakage** (perdita di informazioni). Questo perché la stragrande maggioranza delle app che rientrano in questa categoria chiede di avere accesso ad informazioni sanitarie dell'utente, includendo anche, per esempio, alcune sue malattie o i farmaci che quest'ultimo assume.

Fortunatamente, le informazioni più sensibili dell'utente potrebbero non essere necessarie quando è sufficiente conoscere la sua routine o le sue preferenze per proporre pubblicità mirata in collaborazione con servizi terzi e/o pubblicitari.

Potenzialmente, però, la negligenza sopra citata potrebbe far trapelare informazioni sensibili ad attori potenzialmente malevoli.

È facile comprendere, quindi, quanto questa tipologia di applicazioni può risultare pericolosa in termini di privacy dell'utente.

Per questo motivo, le domande a cui questo documento vuole rispondere sono: *“Ho trovato un insieme di applicazioni nel mondo del Fitness, ma quale tra queste mi assicura di tenere al sicuro la mia privacy? Qual è ‘meno pericolosa’ da utilizzare per me?”*.

Il seguente documento è articolato come segue:

- Nella prima sezione viene fatta una introduzione all'argomento trattato all'interno del documento, definendo chiaramente l'obiettivo da raggiungere.
- Nella seconda sezione vengono elencati tutti gli strumenti utilizzati durante lo sviluppo di tale progetto.
- Nella terza sezione viene definito, passo per passo, il workflow che si è scelto di seguire.
- Nella quarta ed ultima sezione vengono tirate le somme alla fine dello sviluppo di tale progetto, elencando ogni risultato ottenuto.

Strumenti utilizzati

In questa sezione verranno elencati tutti gli strumenti utilizzati durante lo sviluppo di tale progetto e verrà evidenziato il modo in cui ognuno di essi è stato utilizzato al fine di raggiungere il nostro obiettivo.

Innanzitutto, è doveroso menzionare tutti i tool che sono stati testati nelle fasi preliminari ma che, purtroppo, non si sono rivelati utili per svariati motivi, i quali verranno ampiamente analizzati nella sezione dedicata alle problematiche incontrate.

Tra questi troviamo:

- Burp Suite Professional
- Apple Remote Virtual Infrastructure & Wireshark
- Fiddler
- Charles Proxy
- Mitmproxy

Nonostante su Windows non funzionasse come avrebbe dovuto, si è scelto di utilizzare ugualmente **mitmproxy** su MacOS accompagnato da uno script Python (**har_dump.py**) poiché la loro unione ha permesso di effettuare sniffing su una specifica porta (8080) e di esportare l'intero contenuto in vari formati, nel nostro caso il '.har'.

Inoltre, è stato scritto uno script Python (**analyze.py**) che ha permesso di generare un file '.txt' a partire da un '.har' contenente solo le informazioni che è stato deciso di ricercare come, ad esempio, il *nome e cognome* dell'utente, la *data di nascita*, il *peso* e l'*altezza*, il *nome del dispositivo*, ecc.

Infine, l'ultima parte di questo progetto è consistita nell'analizzare ogni file di testo generato e capire quali informazioni venivano inviate presso siti di terze parti e per quale scopo.

I risultati di tutte le sessioni effettuate verranno ampiamente descritti nell'ultima sezione di questo documento, ovvero **Risultati**.

Dispositivi utilizzati

Per una totale trasparenza, è giusto elencare i dispositivi che sono stati utilizzati durante lo svolgimento di tale progetto.

Da quanto è possibile percepire nelle righe precedenti, è stato necessario e sufficiente l'utilizzo di uno smartphone ed un PC/laptop.

Nel nostro caso, tali dispositivi sono i seguenti:

Tipo	Smartphone	Tipo	Laptop
Marca	Apple	Marca	Apple
Modello	iPhone X	Modello	MacBook Pro 16" 2020
OS	iOS 14.2	OS	MacOS Big Sur 11.0.0
CPU	Apple A11 Bionic	CPU	Intel Core i9 8-core 2.3GHz
GPU	Apple M11	GPU	AMD Radeon Pro 5500M 4GB
Storage	64GB	Storage	1TB

Mitmproxy

Mitmproxy è uno tool gratuito e open source la cui killer-feature è la capacità di analizzare il traffico delle app mobile crittografato con protocollo Transport Layer Security (TLS).

Questo strumento risulta migliore sia in termini di efficienza che di semplicità d'uso rispetto a Wireshark quando si tratta di esaminare il traffico di rete crittografato con TLS ed il suo prezzo di zero euro batte la non economica Burp Suite. L'unico aspetto negativo (o positivo per alcuni) è che mitmproxy è principalmente, ma non del tutto, uno strumento a riga di comando, a differenza della GUI di Burp Suite.

In realtà, quando si parla di "mitmproxy" si fa riferimento ad un insieme di tre tool che sono i seguenti:

- **Mitmproxy:** è uno strumento di console che consente l'esame interattivo e la modifica del traffico HTTP. Differisce da mitmdump

in quanto tutti i flussi sono tenuti in memoria, il che significa che è destinato a prendere e manipolare campioni di piccole dimensioni.

```

>>14:50:08 HTTPS POST api2.branch.io /v1/open 200 _plication/json 254b 289ms
14:50:08 HTTPS GET n.ls.apple.com /config/defaults?os=ios&os_version=14... 304 [no content] 179ms
14:50:08 HTTPS POST ads.mopub.com /m/gdpr_sync 200 _plication/json 282b 328ms
14:50:08 HTTPS POST ads.mopub.com /m/open 200 _plication/json 49b 416ms
14:50:08 HTTPS GET ity.api.ua.com /users/3597852524014915509?fetch_email... 200 _plication/json 876b 278ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/usergear/?limit=40&offset=8&user... 200 _plication/json 176b 324ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/heart_rate_zones/?user=185904151 200 _plication/json 326b 502ms
14:50:08 HTTPS POST ies.api.ua.com /v1/meta/sync_prefs 200 _plication/json 14b 383ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/heart_rate_zones/?user=185904151 200 _plication/json 326b 561ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/friendship/?status=pending&to_us... 200 _plication/json 196b 574ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/live_tracker/ 200 _plication/json 178b 552ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/user/185904151/ 200 _plication/json 782b 736ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/user_stats/185904151/?aggregate... 200 _plication/json 478b 746ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/remoteconnection/ 200 _plication/json 166b 350ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/user_stats/185904151/?aggregate... 200 _plication/json 478b 667ms
14:50:08 HTTPS GET ile.api.ua.com /v7.2/training/dynamic_plan/?end_date... 200 _action/hal+json 237b 1.65s
14:50:08 HTTPS GET ile.api.ua.com /v7.2/training/recurring_plan/?end_dat... 200 _action/hal+json 245b 611ms
14:50:08 HTTPS POST l.ls.apple.com /dispatcher.arpc 200 961b 162ms
14:50:08 HTTPS GET fig.api.ua.com /fota/prod 200 _plication/json 1.02k 570ms
14:50:08 HTTPS GET oubleclick.net /getConfig/pubsetting?app_name=QSAWL8W... 200 _plication/json 103b 214ms
14:50:08 HTTPS POST ity.api.ua.com /client-events 202 [no content] 457ms
14:50:08 HTTPS GET cdn.branch.io /sdc/uriskiplist_v2.json 200 text/json 2b 58ms
14:50:08 HTTPS POST amplitude.com / 200 text/html 7b 455ms
14:50:08 HTTPS GET l.ls.apple.com /dispatcher.arpc 200 961b 163ms
14:50:08 HTTPS POST l.ls.apple.com /dispatcher.arpc 200 961b 164ms
[3/48] [x:8080]

```

Figura 1 – mitmproxy in esecuzione su MapMyRun

- **Mitmweb:** è l'interfaccia utente web-based di mitmproxy che consente di esaminare e di modificare il traffico http tramite web-browser. Differisce da mitmdump per lo stesso motivo di mitmproxy.

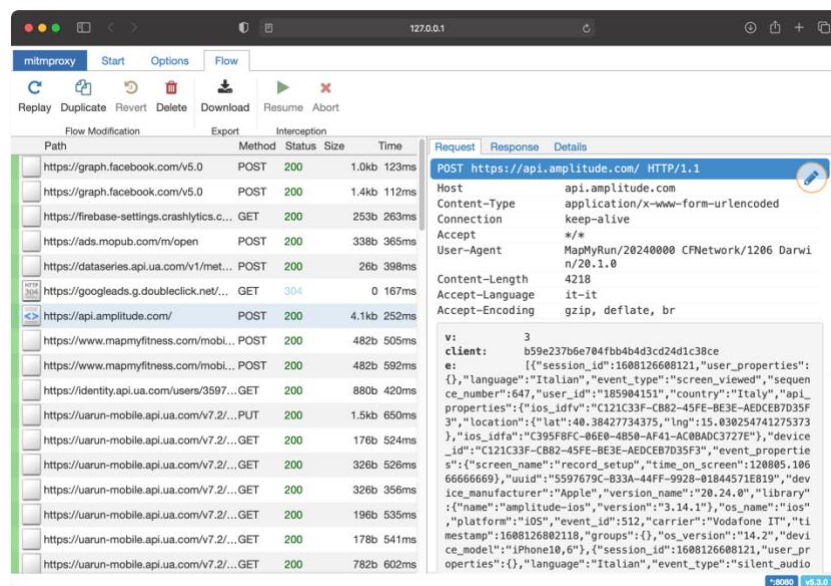


Figura 2 – mitmweb in esecuzione su MapMyRun

- **Mitmdump:** è la versione a riga di comando di mitmproxy. Fornisce funzionalità simili a tcpdump e consente di visualizzare, registrare e trasformare l'intero traffico HTTP.

```

manu@MBP-di-Emmanuel progReti % mitmdump -s ./files/har_dump.py --set hardump=./MapMyRun_iOS_2_GPS.har -- 112x28
Loading script ./files/har_dump.py
Proxy server listening at http://*:8080
192.168.178.113:57483: clientconnect
192.168.178.113:57484: clientconnect
192.168.178.113:57485: clientconnect
192.168.178.113:57486: clientconnect
192.168.178.113:57487: clientconnect
192.168.178.113:57488: clientconnect
192.168.178.113:57489: clientconnect
192.168.178.113:57490: clientconnect
192.168.178.113:57491: clientconnect
192.168.178.113:57492: clientconnect
192.168.178.113:57493: clientconnect
192.168.178.113:57494: clientconnect
192.168.178.113:57495: clientconnect
192.168.178.113:57486: GET https://graph.facebook.com/v5.0/43211574282?fields=app_events_feature_bitmask%2Cname%
2Cdefault_share_mode%2Cios_dialog_configs%2Cios_sdk_dialog_flows.os_version%2814.2.0%29%2Cios_sdk_error_categori
es%2Csupports_implicit_sdk_logging%2Csdpv4_nux_enabled%2Csdpv4_nux_content%2Capp_events_session_timeout%2Clogin
g_token%2Crestrictive_data_filter_params%2Ccam_rules%2Csuggested_events_setting%2Cauto_event_mapping_ios%2Csdk_u
pdate_messagekformat=json&include_headers=false&sdk=ios HTTP/2.0
<< 200 665b
192.168.178.113:57497: clientconnect
192.168.178.113:57496: clientconnect
192.168.178.113:57487: GET https://graph.facebook.com/v5.0/43211574282/mobile_sdk_gk?fields=gatekeepers&format=j
son&include_headers=false&platform=ios&sdk=ios&sdk_version=5.13.1 HTTP/2.0
<< 200 387b
::ffff:192.168.178.113:57486: HTTP/2 connection terminated by client: error code: 0, last stream id: 0, addition

```

Figura 3 – mitmdump in esecuzione su MapMyRun

È doveroso far notare che i tool appena elencati differiscono tra di loro solo per quanto riguarda il front-end poiché il funzionamento e l'obiettivo sono uguali per tutti e tre.

Per poter utilizzare mitmproxy su smartphone (iOS/Android) è necessario seguire tre piccoli step:

1. Su PC: lanciare il comando 'mitmproxy' (o mitmdump/mitmweb) da terminale e aspettare che parta;
2. Su smartphone: configurare il proxy, andare all'indirizzo "<http://mitm.it/>" e scaricare il certificato per il proprio dispositivo;

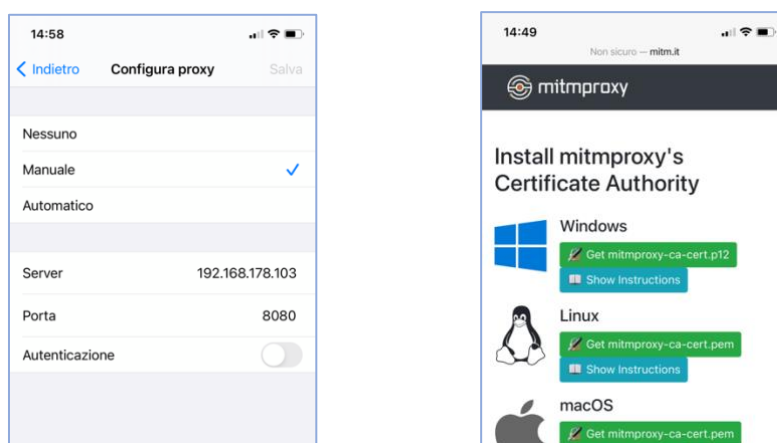


Figura 4 – configurazione del Proxy e download del certificato

3. Aprire le impostazioni del dispositivo, installare il profilo e abilitare l'attendibilità completa per il certificato appena scaricato.

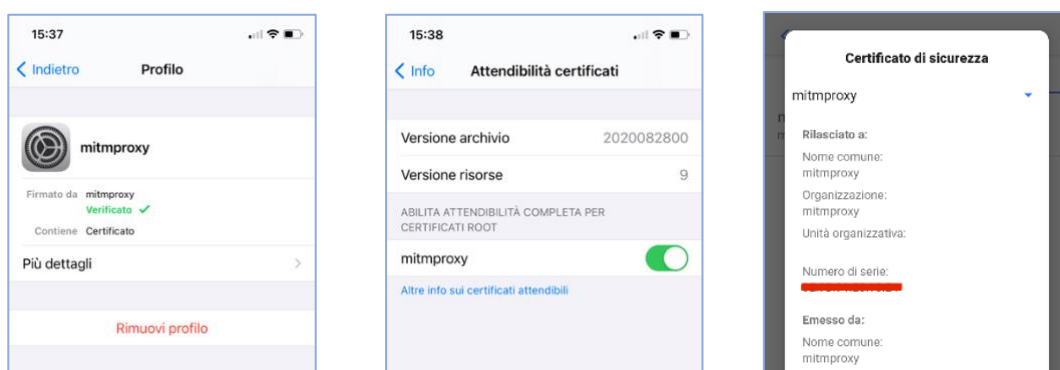


Figura 5 – installazione del certificato e abilitazione attendibilità completa iOS e Android

Har_dump.py

Har_dump.py è lo script che ha reso possibile l'esportazione dell'intera sessione di sniffing in formato '.har'.

Il formato di archivio HTTP, o HAR, è un formato di file di archivio in formato JSON per il logging dell'interazione di un web browser con un sito. Nel nostro caso, il browser web è stato Safari per iOS e Chrome per Android.

L'intero script è condiviso pubblicamente dagli sviluppatori di mitmproxy ed è possibile scaricarlo da questo [link](#).

Per quanto riguarda l'esecuzione, il comando da lanciare è il seguente:

```
1 python3 mitmdump -s ./path/to/har_dump.py --set hardump=
2 ./App_OS_Version_Localization.har
```

Durante l'esecuzione è possibile osservare direttamente sul terminale ogni richiesta che viene effettuata dall'applicazione presa in considerazione verso i siti sia di API che pubblicitari. Tutto ciò, però, non è finalizzato solo alle applicazioni che installiamo sullo smartphone. Infatti, è possibile analizzare anche tutte le richieste che vengono fatte quando si visita, ad esempio, un sito web.


Una volta soddisfatti del tempo trascorso ad analizzare, nel nostro caso, un'applicazione, basterà semplicemente interrompere lo sniffing con un **SIGABRT**, ovvero la combinazione di tasti **CTRL+C**.

A questo punto lo script genererà il file '.har' che conterrà ogni richiesta effettuata in formato json con ogni suo campo come gli headers, queryString, path, ecc....

Analyze.py

Analyze.py è lo script Python che è stato scritto per analizzare i file '.har' generati da mitmdump e generare i file di testo per semplificare la ricerca di informazioni sensibili.

Questo script può essere lanciato da terminale tramite il seguente comando:




```
1 python3 analyze.py path/to/src_file.har
```

Quello che analyze.py fa è scorrere tutte le richieste effettuate, leggere il metodo (GET, POST, DELETE, ecc...) e, sulla base di quest'ultimo, recuperare i campi interessati della richiesta e analizzarli.

Per esempio, se una richiesta è stata effettuata con metodo GET allora non sarà possibile ricercare all'interno del body il campo 'postData' ma è sufficiente soffermarsi sui campi 'url', 'headers', 'cookies' e 'queryString'.

Per ogni coppia chiave/valore presente in ognuno di questi campi sono state utilizzate, quindi, le espressioni regolari per ricercare un pattern specifico all'interno di ogni riga di testo.



```
1 class Constant:
2     NOME_COGNOME = r"\b[LL]eonardo\s*[Cc]aruso\b|\b[LL]eonardo\b|\b[cC]aruso\b"
3     EMAIL = r"(\bleonardo.caruso597@gmail.com\b)|(\bleonardocaruso597@gmail.com\b)"
4     GENDER = r"\bgender\b"
5     PESO = r"\bweight\b"
6     BIRTHDATE = r"\bbirthdate\b"
7     CELLULARE = r"\b351\d{8}\b"
8     ALTAVILLA = r"Altavilla"
9     MATTINE = r"\bMattine\b"
10    POSTAL_CODE = r"\b8404\d\b"
11    CARRIER = r"\b(Vodafone IT)|(VodafoneIT)|(Vodafone)\b"
12    PHONE_NAME = r"\b(i[pP]hone\s*(10%2C6|[1-9][0-9]?,[0-9])\d+s?|ce|[xX]|di\s*[Ee]mmanuel?)\b"
13    UDID = r"\b\d{16}\b"
14    IDFA = r"\b(idfa|IDFA)\b"
15
```

Figura 6 – pattern ricercati all'interno delle richieste

In tutto sono stati definiti 13 pattern da ricercare ed ognuno di essi è mostrato in figura 6, ovviamente oscurando le informazioni personali. Inoltre, è doveroso premettere che in alcune applicazioni sono state trovate informazioni non definite tramite espressioni regolari ma si è scelto ugualmente di conservarle poiché considerate importanti.

```
1 def analyze(request, items, output_file, index):
2
3     # Booleano per stampare oppure no le informazioni all'interno del file:
4     # - Se found = false -> non è stato trovato nulla -> non scrivere la request nel
5     # file
6     # - Se found = true -> è stata trovata qualche informazione -> inserisci
7     # l'informazione in info e scrivi la request nel file
8     found = False
9     info = []
10
11     for line in items:
12
13         if re.findall(Constant.NOME_COGNOME, line):
14             if not 'Nome_Cognome' in info:
15                 info.append("Nome_Cognome")
16                 found = True
17
18         if re.findall(Constant.EMAIL, line): ...
19
20         if re.findall(Constant.GENDER, line): ...
21
22         if re.findall(Constant.PESO, line): ...
23
24         if re.findall(Constant.BIRTHDATE, line): ...
25
26         if re.findall(Constant.CELLULARE, line): ...
27
28         ...
29
30     if found:
31         output_file.write("Ho trovato nella richiesta " + str(index) + ": ")
32         for inf in info:
33             output_file.write(inf.upper() + ", ")
34         output_file.write("\n\n")
35         output_file.write(json.dumps(request, indent=4, sort_keys=True))
36         output_file.write('\n\n#####\n\n')
```

Figura 7 – piccola parte del metodo analyze

L'intero script e le varie sottoclassi utilizzate (GUIAnalyze, Constant, Methods) sono tutti disponibili presso questa [repo](#) su Github.

L'utilizzo di tutti questi strumenti verrà mostrato successivamente nella sezione dedicata al Workflow.

Problematiche

In questa sezione verranno elencate tutte le problematiche incontrate durante lo sviluppo di tale progetto che ci hanno portato ad optare per altre soluzioni. Più nel dettaglio, si parlerà dei software di cui si è accennato nel paragrafo precedente spiegando i motivi che hanno portato ad utilizzare altri tool.

Oltre a ciò, è giusto dedicare qualche riga a quelle applicazioni che, nonostante l'installazione corretta del certificato di mitmproxy sullo smartphone, non hanno reso possibile l'analisi del traffico. Si sta parlando di:

- GPS Strava
- Puma Trac

Una soluzione a questo problema è stata trovata attraverso l'utilizzo di **Frida** ([link](#)) ma non è stato possibile integrare questo toolkit poiché è richiesto che il device abbia il Jailbreak (oppure i file .ipa delle app ma, sfortunatamente, quelli che sono stati trovati erano molto vecchi).

Purtroppo, l'iPhone che è stato utilizzato per l'analisi girava su iOS 14.2 che era attualmente l'ultima versione disponibile per cui non esisteva (e non esiste attualmente in data 17/12/2020) un Jailbreak disponibile pubblicamente e non era possibile effettuare un downgrade ad una versione precedente poiché Apple non permette questo tipo di operazioni dopo aver chiuso le firme dei certificati delle versioni più vecchie.

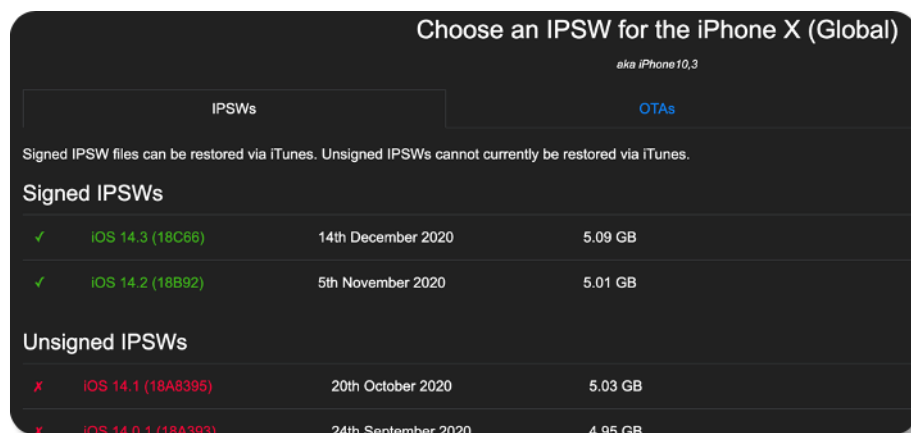
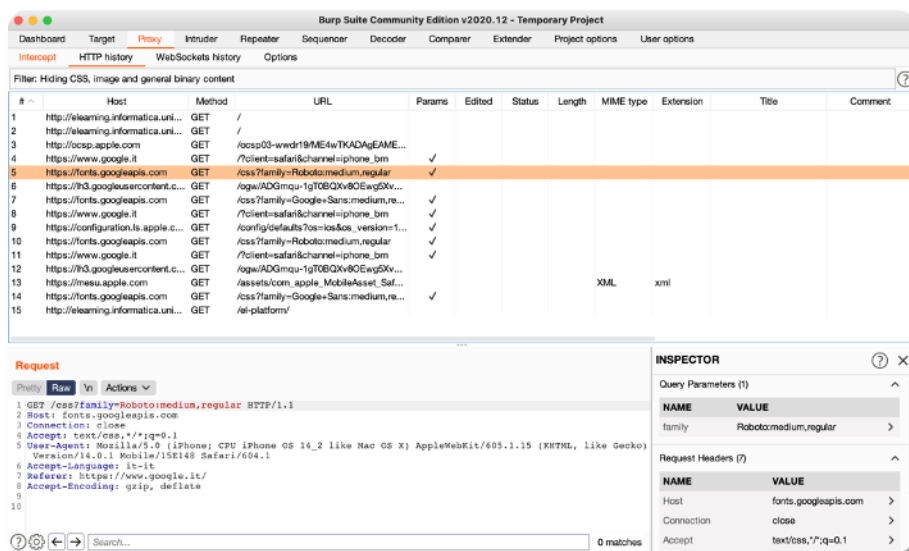


Figura 8 – ultima versione di iOS disponibile a cui poter fare un downgrade in data 17/12/2020

Inoltre, un'applicazione che risulta abbastanza scaricata che si era intenzionati ad analizzare, ovvero Sportractive Running & Fitness, nelle settimane in cui questo progetto è stato sviluppato non era presente sull'App Store di Apple e, quindi, non è stato possibile procedere.

Burp Suite Community Edition



• *Figura 9: interfaccia principale di Burp*

Come è possibile notare dalla figura 9, nonostante l'avvio della sessione e la corretta visualizzazione delle richieste in tempo reale, Burp Suite non consente di esportare in alcun modo il file di log dell'intera sessione.

Inoltre, con la versione gratuita non è possibile salvare nessuna configurazione legata al progetto, quindi ogni volta che si chiude il tool bisogna eseguire nuovamente il setup iniziale, che contiene la creazione del progetto (temporaneo) e la configurazione del proxy.

Proprio per il primo motivo, si è scelto di abbandonare questo tool e di proseguire oltre.

Apple Remote Virtual Infrastructure & Wireshark

Proseguendo nelle ricerche del tool da utilizzare, è stata trovata [questa guida](#) che spiega come utilizzare un software di Apple chiamato rvictl (Remote Virtual Interface) insieme a Wireshark per analizzare il traffico su dispositivi mobile.

Dopo aver scaricato Xcode per installare i command line tools necessari per eseguire rvictl si è proceduto, quindi, con la prima fase di test.

Il comando incaricato alla creazione dell'interfaccia virtuale su cui Wireshark avrebbe dovuto intercettare i pacchetti è

```
1 rvtctl -s <UDID_device>
```

ma, sfortunatamente, quando è stato lanciato non è accaduto nulla: nessun errore, nessuna risposta.

Dopo svariati giorni è stata trovata [questa discussione](#) in cui si parla proprio di questo problema che, a quanto sembra, è causato da un bug della prima (e ultima al momento dei test) versione di MacOS Big Sur, ovvero la 11.0.0.

Fiddler

Per utilizzare questo software è necessario creare un account personale. Fiddler è un software con un'interfaccia abbastanza pulita e moderna. Lo sniffing dei dati necessita di una configurazione del proxy e, dai test effettuati, sembra funzionare molto bene. Infatti, al contrario di Wireshark, riesce a catturare senza impostazioni particolari o estensioni aggiuntive tutti i pacchetti, compresi quelli con crittografia SSL e TLS. Il problema che ha portato Fiddler a non risultare utile per lo svolgimento di tale progetto nasce in fase di esportazione della sessione di sniffing.

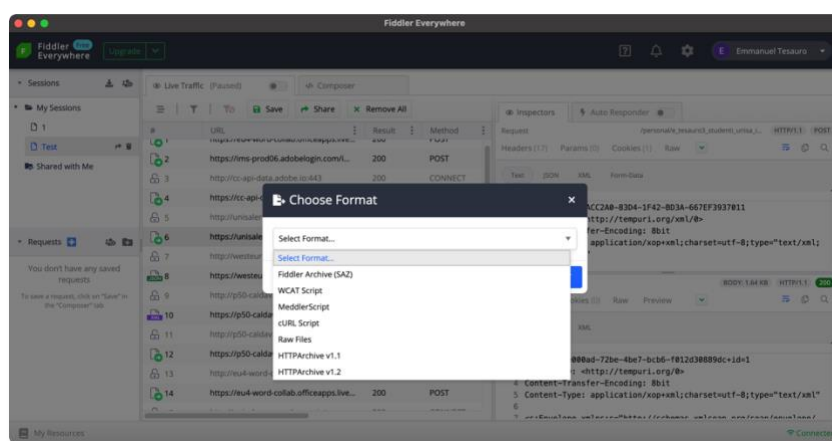


Figura 10: formati di esportazione disponibili in Fiddler

Come è possibile notare, questi sono gli unici formati nei quali è possibile effettuare l'export. Si è tentato di capire quale formato fosse più consono per

poter utilizzare lo script in Python, ma, purtroppo, tutti erano di difficile decriptazione e questo avrebbe inficiato sia sulla qualità dello script, sia sul poco tempo a disposizione per portare a termine il progetto. Si è preferito, quindi, optare per un'altra soluzione.

Charles Proxy

Charles Proxy è il software che era stato scelto da utilizzare per lo sviluppo di tale progetto. Con esso, infatti, è possibile fare sniffing su una determinata porta ed esportare il log dell'intera sessione in formato '.chlsj', ovvero una sorta di json proprietario. Nonostante esistesse uno script che convertisse i file .chlsj in json, il problema che è stato riscontrato è che la versione gratuita di Charles Proxy pone un limite di 10000 richieste da poter effettuare.

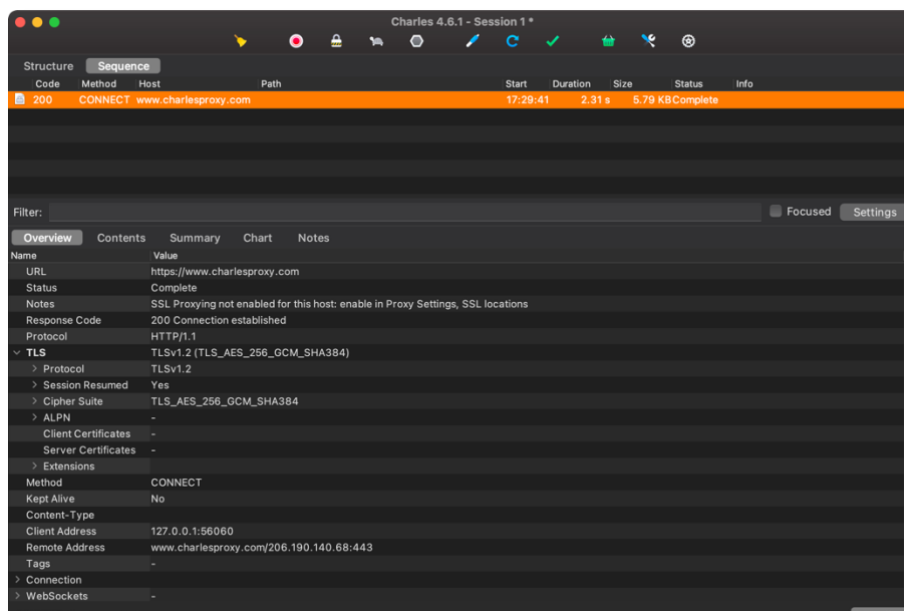


Figura 11: interfaccia di Charles Proxy

Inoltre, un'altra limitazione è data dal fatto che, sempre con la versione gratuita, Charles può essere utilizzato al massimo per 30 minuti prima di doverlo riavviare forzatamente.

Mitmproxy

Le problematiche riscontrate nell'utilizzo di questo software sono tutte focalizzate sull'esportazione della sessione in formato '.har' **su piattaforma Windows**. Infatti, a partire dal 5 Novembre 2018 è noto un bug che non

consente allo script *har_dump.py* di generare il file .har *solo ed esclusivamente su Windows*. Questo bug è molto particolare: quando si tenta di salvare il file alla fine della sessione di sniffing utilizzando l'unica combinazione di tasti disponibile **Ctrl + C**, il software interrompe correttamente la sessione ma senza salvare il file.

Purtroppo, questo problema non è stato ancora risolto e la discussione può essere trovata a questo [link](#) di Github.

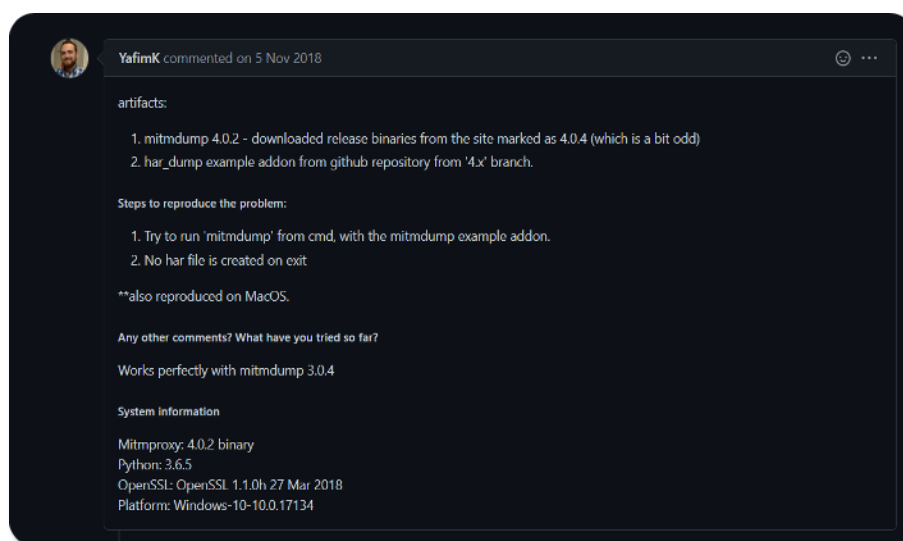


Figura 12: issue presentata su Github

Successivamente si è tentato di virtualizzare l'ambiente di sviluppo con una macchina Linux e, poi, MacOS, ma senza successo. Il problema riscontrato ha avuto a che fare con il reindirizzamento del traffico sulla macchina virtuale. Sulla documentazione è presente anche una guida disponibile a questo [link](#). Purtroppo, però, nonostante i molteplici test, si è potuto constatare che i comandi che le guide suggeriscono di usare sono cambiati e, ad oggi, non si conosce ancora un modo per eseguire la stessa operazione attraverso comandi differenti.

Quindi, alla fine, il progetto è stato effettuato solamente su un dispositivo Mac con dispositivo iOS.

Per quanto riguarda il certificato di mitmproxy da installare, i problemi si sono verificati su Android 10 in quanto non viene installato correttamente. Ripristinando il dispositivo ad Android 9, il certificato viene installato correttamente.

Workflow

In questa sezione verrà descritto il workflow che si è deciso di seguire per l'analisi di ogni applicazione che è stata presa in considerazione.

Innanzitutto, si è proceduto alla creazione di account fittizio da utilizzare in ogni applicazione che richiedesse la registrazione dell'utente in modo tale semplificare notevolmente la ricerca delle informazioni sensibili che, a questo punto, sarebbero state sempre le stesse.

L'account creato è il seguente:

Nome	Leonardo
Cognome	Caruso
Birthdate	30/01/1998
Genere	Uomo
E-mail	leonardo.caruso@gmail.com
Telefono	+39 351xxxxxxx
Pw Google	xxxxxxxxx



Figura 13: tessera sanitaria di Leonardo Caruso

Gli step da seguire durante ogni analisi sono stati i seguenti:

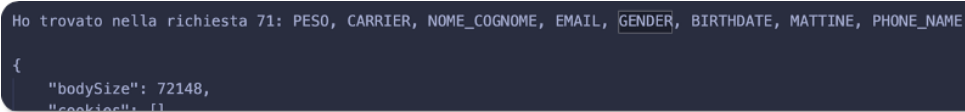
1. Assicurarsi di avere il proxy configurato correttamente su smartphone prima dell'inizio di una sessione di sniffing;
2. Far partire mitmdump su PC con il comando descritto nella sezione degli strumenti utilizzati;
3. Aprire l'applicazione per la prima volta ed effettuare, dove possibile, la registrazione; altrimenti, se l'app fosse stata già avviata almeno una volta per altre sessioni, saltare direttamente allo step 4;
4. Continuare ad usare l'app per una durata massima di 4 minuti cercando di usare le funzionalità principali dell'app stessa;

5. Al termine dei 4 minuti, interrompere la sessione ed analizzare il file generato.

A questo punto, mitmdump avrebbe generato il file di log dell'intera sessione che sarebbe andato in input ad **analyze.py** di cui si è parlato precedentemente. L'ultima operazione da fare (forse la più complessa) è stata quella di leggere tutte le richieste (una per una) nel file di testo in cui erano presenti solo quelle con informazioni sensibili.

Si è scelto di procedere in questo ordine:

6. Lettura della prima riga di ogni richiesta per capire quali informazioni sono state trovate all'interno di essa;



```
Ho trovato nella richiesta 71: PESO, CARRIER, NOME_COGNOME, EMAIL, GENDER, BIRTHDATE, MATTINE, PHONE_NAME
{
  "bodySize": 72148,
  "cookies": []
}
```

Figura 14 – informazioni trovate nella richiesta 71 di MapMyRun

7. Ricerca di ogni singola informazione sopra riportata all'interno della richiesta stessa e ricerca dell'URL di destinazione online per comprendere che tipo di servizio offrisse;



```
text: "V3d4t1cRt_039c23700c704700b4b05c47d1c30cde_03070021session_10422154007052020210422007"
},
"queryString": [],
"url": "https://api.amplitude.com/"
}
```

Figura 15 – indirizzo di destinazione della richiesta 71 di MapMyRun

8. Se l'URL di destinazione fosse stato quello di un servizio pubblicitario o di terze parti, allora sarebbe stato necessario aggiornare il foglio elettronico contenente tutti i dati ritenuti importanti aggiungendo proprio queste informazioni e questo URL in corrispondenza dell'applicazione presa in considerazione.
9. Ripetere gli step 1-9 per ogni applicazione.

Una volta definito qual è stato il workflow che si è deciso di seguire, è ora di vedere i risultati ottenuti al termine di tutte le sessioni di sniffing.

Risultati

In questa sezione, verranno ampiamente analizzati tutti i risultati ottenuti dall'analisi di tutte le applicazioni che sono state prese in considerazione.

Queste applicazioni sono le seguenti:

- Nike Run Club
- MyFitnessPal
- Adidas Running
- Yazio
- Fitness Buddy
- MapMyRun
- RunKeeper
- Lifesum
- Decathlon Coach
- MyWellness

Ovviamente, sono ovviamente escluse Sportractive Running & Fitness, PumaTrac e GPS Strava per i motivi citati nella sezione “**Problematiche**”.

È doveroso premettere che alcuni servizi pubblicitari/terzi sono stati utilizzati all'interno di molteplici applicazioni e, per tanto, ognuno di esso verrà descritto solo la prima volta che verrà incontrato in seguito.

Si procederà, quindi, discutendo di ogni applicazione, evidenziando dove necessario ogni particolare.

Al termine della descrizione di ogni applicazione, verrà inserito un grafico riassuntivo di quanto descritto nella sezione specifica.

La prima applicazione analizzata è stata NikeRunClub e di seguito saranno elencati i risultati.

Nike Run Club

Nike Run Club è la famosissima app di Nike che permette di monitorare la propria corsa e di ricevere consigli durante ogni sessione di allenamento.

Sono state eseguite 4 sessioni sniffing su quest'app e le informazioni inviate verso l'esterno che sono state scovate verranno prima analizzate e, successivamente, riassunte in un grafico.

Detto ciò, le informazioni di cui si parla sono il sesso dell'utente, l'operatore telefonico, il nome/modello del dispositivo, la versione del sistema operativo e l'IDFA (Identifier for Advertising).

Per quanto riguarda i primi tre e l'IDFA, questi vengono usati per analisi interne utilizzando l'indirizzo **analytics.nike.com** e/o **singular.net**.

Inoltre, l'operatore telefonico, il modello del dispositivo e la versione del dispositivo vengono inviati a **urbanairship.com** probabilmente per fornire annunci mirati in base alle preferenze dell'utente poiché Airship è una società americana che fornisce servizi di marketing e branding.

Si noti, però, che Nike Run Club ha inviato una versione del sistema operativo errata in quanto ha inviato la 14.0 beta 1 invece della 14.2.

Continuando, il modello del dispositivo e l'IDFA vengono inviati verso l'indirizzo **singular.net**, il quale è una piattaforma di marketing intelligence che fornisce agli operatori di marketing informazioni utili da dati precedentemente raggruppati (in questo caso, quelli degli utenti).

	analytics.nike.com	urbanairship	newrelic	api.nike.com/marketing/tracking_events	singular.net
Nike Run Club					
Gender	•				
Carrier	•	•			
Phone name	•	•	•		•
OS Version		•			
IDFA				•	•

MyFitnessPal

MyFitnessPal è una delle app maggiormente utilizzate in ambito fitness e, non a caso, si trova attualmente al 31^o posto della medesima classifica.

Permette di tenere traccia delle proprie attività, del proprio peso, dei pasti che si mangia e tanto altro ancora.

È doveroso premettere che il numero di siti pubblicitari e di terze parti contattati da questa applicazione è maggiore rispetto alla precedente anche perché gli annunci pubblicitari all'interno di MyFitnessPal sono molto di più rispetto a NikeRunClub, la quale ne ha praticamente zero.

Questo è possibile osservarlo dal momento in cui, **oltre gli stessi campi analizzati in precedenza (a differenza della versione del OS)**, quest'applicazione invia anche il campo

- E-mail.

Nel dettaglio, questo campo viene utilizzato con le api di identity.com, le quali si occupano di verificare l'identità dell'utente.

Il modello del dispositivo e l'operatore telefonico sono le informazioni maggiormente utilizzate. Ad esempio, entrambe vengono inviate a servizi pubblicitari di Amazon come **maads.amazon-adsystem.com** e **aax-eu.amazon.adsystem.com**, ma anche ad altri che, per semplicità, verranno espressi solo nel grafico seguente.

MyFitnessPal	identity.api.ua.com	maads.amazon-adsystem.com	amplitude.com	aax-eu.amazon-adsystem.com	ads.mopub.com	googleads.g.doubleclick.net	branch.io	app-measurement.com	adplatform.vrtcal.com	googlesyndication.com	inmobi.com	ads.nexage.com
E-mail	•											
Gender			•									
Carrier		•	•	•	•		•					
Phone name		•	•	•	•	•	•	•	•	•	•	•
IDFA		•		•								

Adidas Running

Adidas Running è un'app molto simile alla rivale NikeRunClub. Permette, quindi, di tracciare le corse, gli allenamenti e molte altre attività sportive.

Anche qui sono state effettuate 4 sessioni di sniffing e, dopo aver analizzato i file di testo, i risultati sono stati molto incoraggianti.

Infatti, solo 4 dei 15 campi analizzati sono stati inviati verso altre fonti.

Nel dettaglio, è possibile trovare un nuovo servizio, denominato **runtastic.pushwoosh.com**, al quale vengono inviati dati come nome e cognome e sesso dell'utente, nome del dispositivo e IDFA.

Questo servizio permette di mandare notifiche push sui dispositivi degli utenti in ogni momento.

Per quanto riguarda il modello del dispositivo, questo viene inviato, come al solito, anche ad altri servizi, i quali sono raffigurati nel grafico seguente.

Adidas Running	runtastic.pushwoosh.com	newrelic	s3.eu-west-1.amazonaws.com	mobile-events.eservice.emarsys.net	google-analytics.com	bam-cell.nr-data.net	app-measurement.com
Nome e Cognome	•						
Gender	•						
Phone name	•	•	•	•	•	•	•
IDFA	•						

Yazio

Yazio è un'applicazione che permette di gestire il proprio diario alimentare, monitorare le attività e dovrebbe garantire la perdita di peso graduale. Inoltre, permette di calcolare la quantità di calorie giornaliere.

È doveroso sottolineare che Yazio è stata l'applicazione più 'sicura' che è stata analizzata. In essa, infatti, sono state trovate informazioni rilevanti solo nelle richieste per due servizi, ovvero **app-measurement.com** e **amplitude.com**.

Inoltre, ad entrambi i servizi viene passato solo il modello del dispositivo.

Yazio	api.amplitude.com	
	app-measurement.com	
Phone name	•	•

Fitness Buddy

Fitness Buddy è un'applicazione che insegna agli utenti tutto ciò che bisogna sapere sugli esercizi, allenamenti e alimentazione.

Contiene più di 100 allenamenti, 400 esercizi 8 piani pasto, 100 ricette facili da fare e tanto altro ancora.

L'analisi di quest'applicazione ha prodotto risultati interessanti in quanto vengono inviati dati come il nome e cognome, il sesso, l'altezza, il peso e l'e-mail dell'utente ad un servizio chiamato **wzrkt.com**.

La particolarità di queste richieste è che, nonostante le ricerche su internet, non si è sicuri al 100% della provenienza di questo dominio. Infatti, visitandolo avremmo in risposta una pagina completamente bianca.

Quelle poche risposte date, seguono tutte la strada che si tratti di un analytics software ([link](#) alla discussione).

Proseguendo, le altre informazioni interessate sono, come al solito, l'operatore telefono e il modello del dispositivo.

Infine, l'ultima informazione inviata verso siti terzi è l'IDFA, in particolare verso **branch.io**.

Il grafico risultante è il seguente:

FitnessBuddy	wzrkt.com	graph.facebook.com	ads.mopub.com	app-measurement.com	clients4.google.com	api2.branch.io
Nome e Cognome	•					
E-mail	•					
Gender	•					
Altezza	•					
Peso	•					
Carrier	•	•				
Phone Name		•	•	•	•	
IDFA						•

MapMyRun

MapMyRun è un'applicazione che, come il nome lascia intendere, permette il tracciamento dei propri allenamenti sia utilizzando lo smartphone, sia dispositivi come l'Apple Watch o il Garmin.

È possibile anche ricevere supporto da altri atleti e creare sfide per sé stessi e/o per gli amici.

MapMyRun è stata una tra le applicazioni più impegnative da analizzare per via dei numerosi servizi che essa contatta.

Alcune informazioni come il nome e cognome, l'e-mail, il sesso, il peso e la data di nascita dell'utente vengono inviati a due servizi, ovvero **amplitude.com** e **uarun-mobile.api.ua.com**.

Altre, invece, vengono inviate a molteplici servizi di terze parti.

Un esempio lampante di quanto è stato appena detto è possibile vederlo dal grafico sottostante in corrispondenza, come al solito, del modello del dispositivo.

MapMyRun	amplitude.com	Uarun-mobile.api.ua.com	Identity.api.ua.com	branch.io	mobile-config.api.ua.com	pubads.g.doubleclick.net	pagead2.googleadservices.com	app-measurement.com	lh3.googleusercontent.com	www.google-analytics.com	www.google.it/pagead	www.google.it/ads	Googleads.g.doubleclick.net	Ads.mopub.com	Pagead2.googleadservices.com	mobile-config.api.ua.com
Nome e Cognome	•	•														
E-mail	•	•														
Gender	•	•														
Peso	•	•														
Data di nascita	•	•														
Altavilla	•	•	•													
Mattine		•	•													
Postal Code	•															
Carrier	•	•		•	•											
Phone Name				•	•	•	•	•	•	•	•	•	•	•	•	•
IDFA				•												

JEFIT Workout Planner

JEFIT è un'applicazione che permette di connettersi con quasi 9 milioni di utenti per ricevere suggerimenti, supporto e feedback. Con essa è possibile creare un piano di allenamento personalizzato, registrare i propri allenamenti e analizzare i propri dati per massimizzare i risultati.

È doveroso premettere questa applicazione è stata la più complessa da analizzare poiché da essa sono state individuate numerosissime richieste verso una vastità di servizi diversi.

Quest'app è sicuramente stata una tra le app più impegnative da analizzare a causa dei tantissimi servizi che contatta.

Le informazioni sensibili come e-mail e sesso dell'utente vengono inviati a servizi esterni come **<https://v/a>** e **imp.control.kochava.com**.

È stata effettuata una ricerca riguardante il primo sito in questione, ma non è stata trovata nessuna corrispondenza e questo lo rende un sito sospetto.

Un esempio dei numerosissimi servizi è possibile vederlo dai grafici sottostanti.

JEFIT	imp.control.kochava.com	https://v/a	aax-eu.amazon-adsystem.com	graph.facebook.com	firebase logging-pa.googleapis.com	aax-eu.amazon-adsystem.com	graph.facebook.com	googleads.g.doubleclick.net	tpc.googlesyndication.com	lh3.googleusercontent	pagead2.googlesyndication.com	gstatic.com	fonts.gstatic.com
E-mail		•											
Gender	•	•											
Carrier			•	•									
Phone Name			•	•	•	•	•	•	•	•	•	•	•
IDFA	•		•										

Questa è la seconda parte del grafico:

JEFIT	pr.ybp.yahoo.com	googletagsservices.com	s.yimg.com	pixel.adsafeprotected.com	ad.doubleclick.net	beap-bc.yahoo.com	s0.2mdn.net	servedby.flashtalking.com	tags.bluekai.com	cdn.flashtalking.com	dpm.demdex.net m	d9.flashtalking.com
E-mail												
Gender												
Carrier												
Phone Name	•	•	•	•	•	•	•	•	•	•	•	•
IDFA												

Infine, questa è la terza parte del grafico:

JEFIT	imp.control.kochava.com	impression-europe.liftoff.io	cdn.liftoff.io	s0.2mdn.net	cdn.doubleverify.com	diagkn.com	beap-bc.yahoo.com	s0.2mdn.net	ad.atdmt.com	tags.bluekai.com	pr.ybp.yahoo.com
E-mail											
Gender											
Carrier											
Phone Name	•	•	•	•	•	•	•	•	•	•	•
IDFA											

Runkeeper

Runkeeper è un'applicazione che conta più di 25 milioni di download dal Google Play Store. Come altre applicazioni già viste fino ad ora, permette di tracciare la propria corsa e visualizzarne successivamente l'itinerario. Inoltre, acquistando la versione a pagamento, è possibile visualizzare ed iniziare una vasta scelta di piani di allenamento.

Runkeeper si piazza vicino a MapMyRun e JEFIT Workout Planner in termini di difficoltà di analisi delle richieste, poiché, come si potrà vedere dal grafico seguente, sono state trovate moltissime richieste verso una molteplicità di servizi di terze parti.

Infatti, si inizia con l'inviare informazioni come il nome e cognome e l'e-mail dell'utente verso i servizi di **iterable.com**, i quali fanno parte di una piattaforma che consente di sfruttare i dati degli utenti ed inviare messaggi in tempo reale su tutti i canali disponibili senza interruzioni.

Sfortunatamente, si prosegue con ciò che fa storcere il naso, ovvero tutte le richieste che inviano il modello del dispositivo verso altre fonti. In queste richieste sono stati trovati esattamente 22 servizi diversi.

Per motivi stilistici, il grafico seguente sarà diviso in due parti.

Qui è possibile osservare la prima metà del grafico:

Runkeeper	api.iterable.com	graph.facebook.com	pubads.g.doubleclick.net	api.amplitude.com	cues.runkeeper.com	firebase-settings.crashlytics.com	fitnesskeeperapi.com	graph.facebook.com	ca.iadsk.apple.com	googleads.g.doubleclick.net	pubads.g.doubleclick.net
Nome e Cognome	•										
E-mail	•										
Carrier		•	•	•							
Phone Name			•		•	•	•	•	•	•	•
IDFA							•				

Mentre qui è presente la seconda parte del grafico contenente le restanti richieste verso I siti di terze parti:

Runkeeper	app-measurement.com	api-glb-euc1b.smoot.apple.com	stocks-edge.apple.com	apple-finance.query.yahoo.com	racerooster.com	cloudflare.com	googletagmanager.com	platform.twitter.com	www.google-analytics.com	googleadservices.com	Rum-http-intake.logs.datadoghq.com	stats.gdoubleclick.net
Nome e Cognome												
E-mail												
Carrier	•	•										
Phone Name			•	•	•	•	•	•	•	•	•	•
IDFA						•	•					

8fit

8fit è un'applicazione che offre una libreria di programmi di allenamento, piani alimentari e persino meditazioni sul sonno. È progettata per essere altamente personalizzabile in modo da poter trovare un piano giornaliero che funzioni con gli obiettivi dell'utente, il suo livello di forma fisica e le sue preferenze.

In 8fit sono state trovate informazioni in più a quelle ricercate. Infatti, durante l'analisi, si è notato che venivano passate informazioni ad un servizio chiamato **sentry.io** come la quantità massima di storage del dispositivo, la data dell'ultima accensione, la memoria utilizzabile rimanente e un booleano per indicare se il dispositivo disponesse dei permessi di root.

Sentry.io è una piattaforma di monitoraggio delle applicazioni che aiuta ogni sviluppatore nel diagnosticare, correggere e ottimizzare le prestazioni del loro codice.

Inoltre, verso questo servizio vengono passate altre informazioni sensibili come il nome e cognome, l'e-mail, il sesso, il peso e la data di nascita dell'utente.

Il grafico riassuntivo è il seguente:

8fit	sentry.io	api.amplitude.com	firebase logging-pa.googleapis.com	clients3.google.com	sdk.iad-01.braze.com	appboy-images.com, app.adjust.com	ca.iad.sdk.apple.com	app-measurement.com	api.amplitude.com	deaxm436wfgs4.cloudfront.net	api-glb-euc1b.smoot.apple.com	app.adjust.com	s2s.adjust.com
Nome e Cognome	•												
E-mail	•												
Gender	•												
Peso	•												
Data di nascita	•												
Carrier		•											
Phone Name			•	•	•	•	•	•	•	•	•		•
IDFA								•				•	•

Lifesum

Lifesum è un'app che ha come obiettivo quello di guidare gli utenti a controllare e riconoscere gli alimenti migliori per la loro dieta. L'app è disponibile sia per dispositivi Apple che Android ed è gratuita per gran parte delle sue funzioni.

Anche qui, i servizi esterni contattati non sono molti e, esattamente come nella maggior parte delle altre app analizzate, è possibile trovare molteplici richieste al cui interno è presente il modello del dispositivo.

Qui, viene richiesto un servizio che prima d'ora ancora non era stato incontrato, ovvero **stream-600.optimove.net**. Da quanto è possibile leggere su Internet, Optimove è una società privata che sviluppa e commercializza un software di marketing relazionale come servizio. Il prodotto Optimove ha al centro una piattaforma dati per i clienti e applica l'ottimizzazione algoritmica per migliorare autonomamente le campagne multicanale.

Inoltre, ritroviamo servizi come **app-measurement.com**, **app-adjust.com** e **branch.io**.

Il grafico riassuntivo dell'analisi di Lifesum è il seguente:

Lifesum	realtime-600.optimove.net	app-measurement.com	realtime-600.optimove.net	graph.facebook.com	api2.branch.io	App.adjust.com	mixpanel.com	stream-600.optimove.net/piwik.php	mbaas.optimove.net
E-mail	•								
Peso	•								
Gender		•	•						
Carrier				•					
Phone name		•		•	•	•	•	•	•
IDFA					•	•	•		

Decathlon Coach

Decathlon Coach è l'applicazione della famosissima catena commerciale di articoli sportivi. Grazie ad essa, è possibile effettuare sessioni di allenamento per molteplici sport come il calcio, la corsa, la camminata sportiva e tanti altri.

L'analisi di questa applicazione ha prodotto risultati abbastanza positivi in quanto le informazioni coinvolte sono solo tre, ovvero l'operatore telefonico, l'IDFA e, come al solito, il modello del dispositivo.

Per quanto riguarda i primi due, questi vengono inviati solo ai servizi di **facebook**, **app-adjust.com** e/o **s2s.adjust.com**.

L'ultimo della lista è l'informazione maggiormente utilizzata, in linea con quanto osservate precedentemente con le analisi effettuate.

Nel grafico seguente è possibile osservare che il modello del dispositivo viene inviato all'indirizzo **account.geonaute.com** che, dopo una ricerca su internet, sembrerebbe essere un servizio dedicato agli account Decathlon.

Decathlon Coach	graph.facebook.com	app.adjust.com	app-measurement.com	clients4.google.com	www.gstatic.com	apple-finance.query.yahoo.com	account.geonaute.com	profile-completion.decathlon.net	s2s.adjust.com
Carrier	•								
Phone name	•	•	•	•	•	•	•	•	
IDFA		•							•

Mywellness

Mywellness è un'applicazione di Technogym e The Gym Hub facile da usare che consente agli utenti di avere uno stile di vita attivo e sano senza le solite complicazioni, stress o spese.

Durante l'analisi di questa applicazione non è stata trovata nessuna anomalia rispetto alle altre app analizzate precedentemente.

È possibile trovare, però, per la prima volta un servizio chiamato **onesignal.com**. OneSignal è un servizio che abilita le notifiche push, estraendo dettagli come la piattaforma su cui è in esecuzione il dispositivo.

L'ultimo grafico riassuntivo è il seguente:

MyWellness	graph.facebook.com	api.onesignal.com	firebase.dynamiclinks.googleapis.com	api.onesignal.com	pay.mywellness.com	cms.mywellness.com	messenger.mywellness.com
Carrier	•	•					
Phone name	•	•	•	•	•	•	•

Grafico riassuntivo

Al termine dell'analisi di ogni applicazione, si è pensato di riassumere nel seguente grafico tutto ciò di cui si è parlato in questa sezione.

Il grafico presenta sulle colonne le varie informazioni che sono state ricercate all'interno delle richieste verso tutti i servizi di cui si è discusso.

Sulle righe, invece, sono presenti tutte le applicazioni analizzate.

RECAP	Nome_Cognome	E-mail	Sesso	Peso	Data di nascita	Cellulare	Altavilla	Mattine	Codice Postale	Operatore Telefonico	Modello del dispositivo / OS Version	UDID	IDFA
Nike Run Club			•			•					•		•
MyFitnessPal		•	•							•	•		•
Adidas Running	•		•								•		•
Yazio											•		
Fitness Buddy	•	•	•	•						•	•		•
MapMyRun	•	•	•	•	•		•	•	•	•	•		•
JEFIT		•	•							•	•		•
Runkeeper	•	•								•	•		•
8fit	•	•	•	•	•					•	•		•
Lifesum		•	•	•						•	•		•
Decathlon Coach										•	•		•
MyWellness										•	•		

Da quanto è possibile notare, l'applicazione 'meno sicura' è stata MapMyRun in quanto invia tutte le informazioni, eccetto il numero di cellulare e lo UDID del dispositivo, verso siti terzi. Al contrario, Yazio è risultata notevolmente più sicura in quanto utilizza solo il modello del dispositivo.