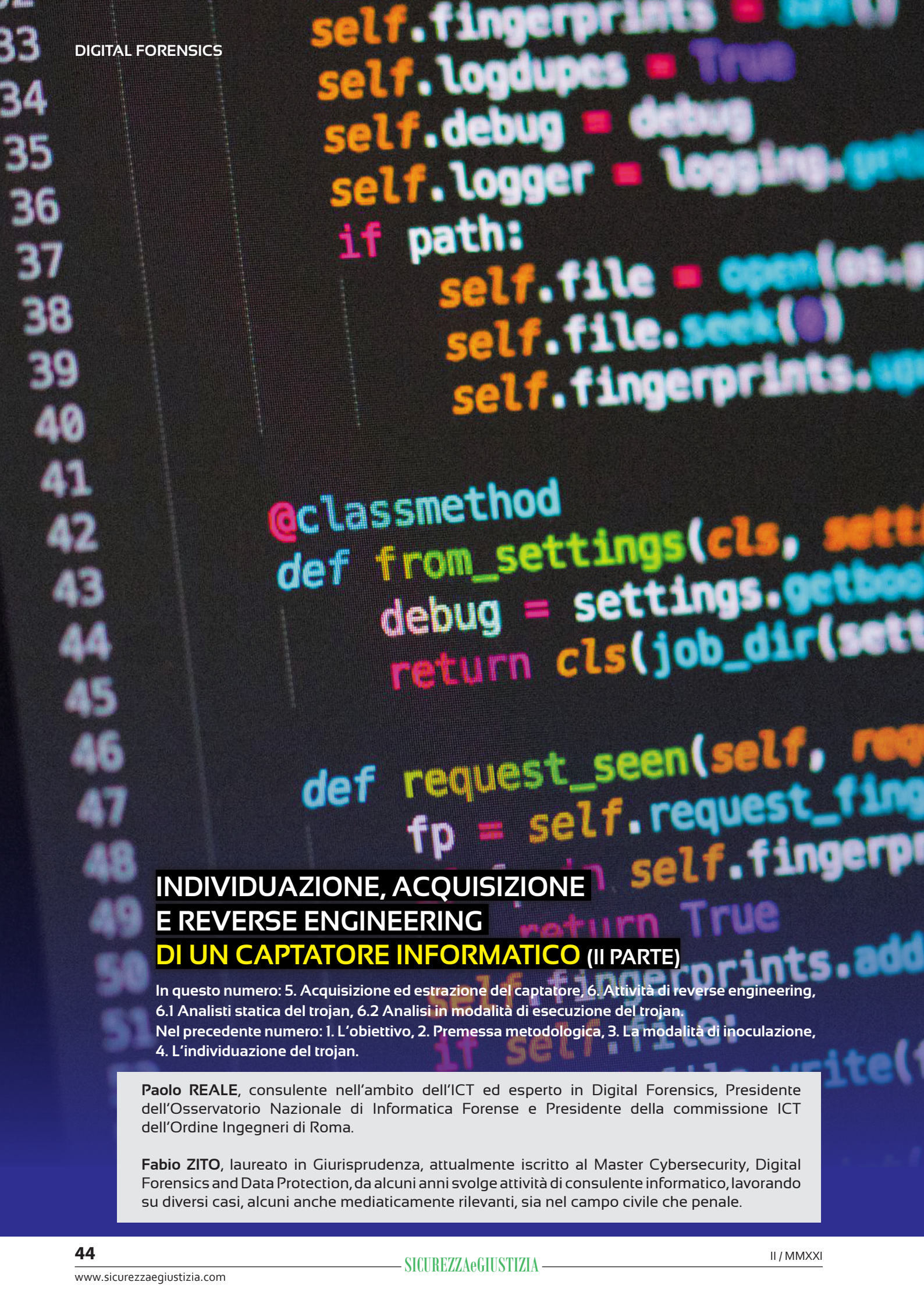

INDIVIDUAZIONE, ACQUISIZIONE E REVERSE ENGINEERING DI UN CAPTATORE INFORMATICO (II PARTE)

In questo numero: 5. Acquisizione ed estrazione del captatore, 6. Attività di reverse engineering, 6.1 Analisti statica del trojan, 6.2 Analisi in modalità di esecuzione del trojan.
Nel precedente numero: 1. L'obiettivo, 2. Premessa metodologica, 3. La modalità di inoculazione, 4. L'individuazione del trojan.

Paolo REALE, consulente nell'ambito dell'ICT ed esperto in Digital Forensics, Presidente dell'Osservatorio Nazionale di Informatica Forense e Presidente della commissione ICT dell'Ordine Ingegneri di Roma.

Fabio ZITO, laureato in Giurisprudenza, attualmente iscritto al Master Cybersecurity, Digital Forensics and Data Protection, da alcuni anni svolge attività di consulente informatico, lavorando su diversi casi, alcuni anche mediaticamente rilevanti, sia nel campo civile che penale.

5. Acquisizione ed estrazione del captatore

Grazie ad un'acquisizione di tipo *file system full* è stato possibile recuperare la directory (individuata nel *path* del file *Manifest.plist*[1] fig. 1) contenente tutti i file dell'applicazione come identificata in precedenza.

ogni applicazione iOS- da cui è possibile ottenere tutte le informazioni di dettaglio sull'applicazione stessa. Proprio l'analisi di questo file ha permesso di comprendere che il nome assegnato all'applicazione visibile a livello utente (una volta installata l'app) non era quello dell'eseguibile, bensì un altro, cioè quello individuato nel valore della chiave "*Bundle display*

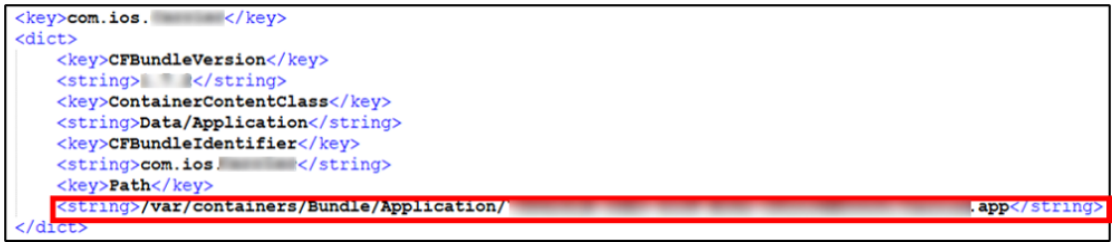

Figura 1 - Estratto del file Manifest.plist presente sul telefono analizzato relativo all'app individuata

Nella directory, infatti, oltre al file *embedded.mobileprovision*, cioè il file XML di cui si è già parlato, sono presenti tutti i file necessari al funzionamento dell'app stessa quali ad esempio immagini, librerie, file di configurazione, etc. (fig. 2).

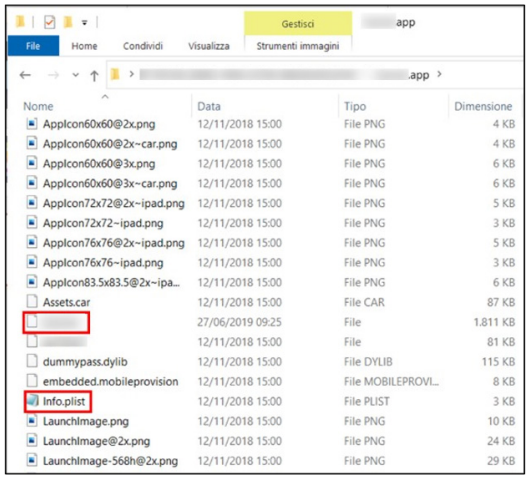

Figura 2 - Parte dell'elenco dei file presenti della directory relativa all'app individuata

Tra questi sono presenti il file eseguibile (il nome è stato oscurato), ovvero il vero e proprio programma, e il file *Info.plist*[2] -presente in

name" (fig. 3, oscurato) contenente un nome appositamente scelto per "camuffare" il nome reale dell'applicazione.

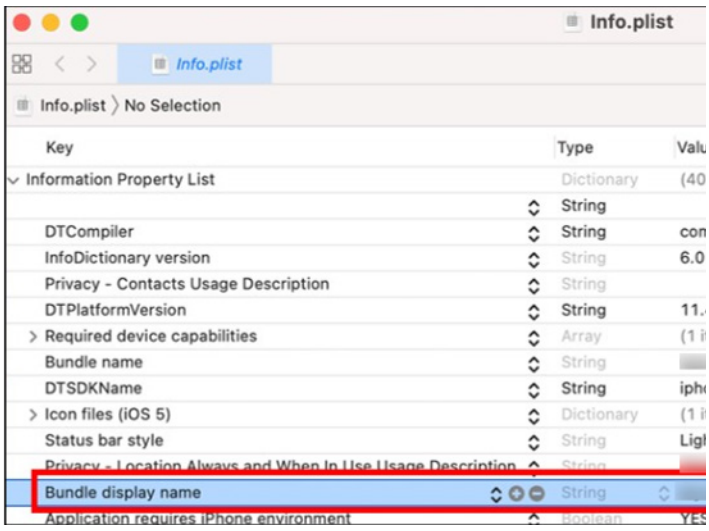

Figura 3 - Estratto del file Info.plist dell'app individuata

6. Attività di reverse engineering

6.1. Analisi statica del Trojan

Individuata, quindi, la cartella contente il file binario del trojan, si è proceduto preliminar-

1 Quando si esegue un backup dispositivo iOS (iPhone, iPad, iPod) generalmente con il software "iTunes" o nel caso di specie con il UFED viene creato un file Manifest.plist dove tra le atre cose vengono registrate le informazioni delle applicazioni installate.

2 File in formato XML dove sono raccolte tutte le proprietà e le informazioni sempre nel formato chiave-valore relative ad una applicazione per maggiori dettagli: https://developer.apple.com/documentation/bundleresources/information_property_list

mente a disassemblare[3] lo stesso per provare ad identificare attraverso una analisi di tipo statico le informazioni di connessione con il server. La prima attività svolta è stata una ricerca di tipo testuale tra i simboli identificati utilizzando alcune parole chiave quali ad esempio ip, server, connection, etc.

Tra i risultati di questa prima attività si è posta particolare attenzione al metodo *initWithURL* della classe *ASIHTTPRequest* (fig 4), in quanto come si evince dal nome stesso e leggendo la documentazione[4] questo metodo viene utilizzato per inizializzare connessioni di tipo HTTP/HTTPS. Essendo noto che le comunicazioni tra trojan e server avvenissero in HTTPS, abbiamo ritenuto probabile che fosse proprio questo il metodo utilizzato dal captatore per comunicare con il server. Anticipando quello che si vedrà meglio nel prossimo paragrafo, è proprio partendo da quest'ultimo metodo che è stato possibile individuare i dati di connessione.

luppatore su uno smartphone "muletto" (fig. 5).
Una volta installato, si provveduto ad eseguirlo in modalità *debugging* direttamente con *lldb* impostando un *breakpoint* sul metodo *initWithURL* -visto nel paragrafo precedente- consentendo di fatto di "bloccare" l'esecuzione del captatore all'inizio della invocazione del metodo stesso.

Così facendo è stato possibile leggere il valore memorizzato nel registro x2, nel quale si è individuata una *url* in HTTPS, che (come scoperto in seguito) appare essere quella da cui probabilmente il trojan leggeva il file di configurazione contente la schedulazione di attivazione/disattivazione del microfono. Come gran parte degli strumenti *debugging*, anche *lldb*, partendo da una determinata istruzione, permette di risalire alle precedenti che l'hanno invocata (*stack-walk*).

```
-[ASIHTTPRequest initWithURL:]:
sub         sp, sp, #0x60 ; Objective C Implementation defined at 0x10011b490 (instance method), DATA XREF=0x10011b490
stp         x29, x30, [sp, #0x50]
add         x29, sp, #0x50
adrp        x8, #0x10012b000 ; 0x10012bc10@PAGE
add         x8, x8, #0xc10 ; 0x10012bc10@PAGEOFF, &@selector(init)
stur        x0, [x29, var_8]
stur        x1, [x29, var_10]
stur        x2, [x29, var_18]
ldur        x0, [x29, var_8]
```

Figura 4 - Dettaglio del metodo Initwithurl del codice disassemblato del trojan

Solo per completezza di informazione, è necessario sottolineare che -oltre a quella appena illustrata- sono state effettuate una moltitudine di prove ed analisi, tra cui il *porting* di porzioni di codice *assembly* in *Objective-C* che però in molti casi non hanno portato a nessun risultato. Per questo motivo, e per non appesantire la lettura, non è stato ritenuto utile descrivere tali fasi in questo articolo.

6.2. Analisi in modalità esecuzione del trojan

Per poter procedere con l'attività di analisi *stack-trace* del captatore, descritte di seguito, onde evitare di operare alcuna alterazione sul telefono originale, è stato scelto di installare il Trojan con l'ausilio di un certificato di tipo svi-

Se questa operazione viene effettuata in modo ricorsivo è possibile ripercorrere a ritroso il "percorso" fatto da una determinata applicazione fino a rintracciare il codice di interesse. Partendo dalla *url* sopra citata, è stato possibile individuare il metodo nel quale sono state inserite in modalità hard coding[5] le informazioni di connessione. Tale metodo, nel codice disassemblato, è denominato come *a932821303*[6] così come mostrato in fig. 6.

3 Principalmente con la versione demo di Hopper Disassembler

4 https://allseeing-i.com/ASIHTTPRequest

5 In informatica, con l'espressione hard coding, o valori cablati, si intende la prassi di introdurre in un codice sorgente dei valori costanti che non possono essere cambiati senza ricompilazione del codice sorgente e quindi irrigidiscono il programma ottenuto dalla compilazione di quel codice sorgente. In altre parole, quel dato è come 'scolpito' all'interno del programma.

6 Probabilmente il fatto che il nome del metodo risulta poco intellegibile è dovuto ad un'attività di obfuscation effettuata sul codice

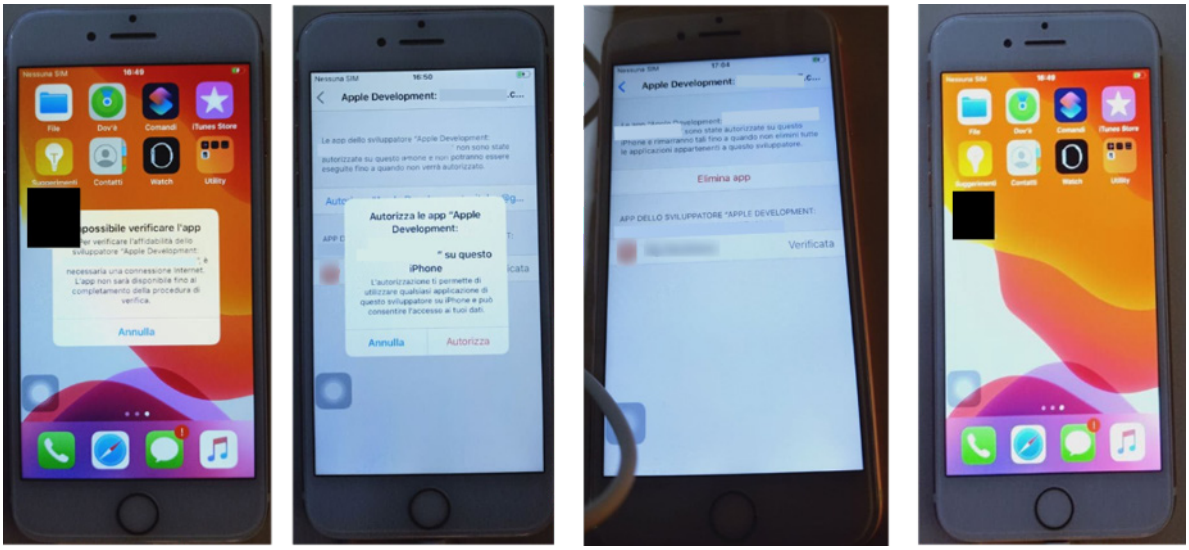

Figura 5 - Dettaglio installazione del trojan su smartphone "Muletto"

```
-[          a932821303]:
stp         x28, x27, [sp, #-0x30]! ; Objective C Implementation defined at 0x100115f60 (instance method),
stp         x20, x19, [sp, #0x10]
stp         x29, x30, [sp, #0x20]
add         x29, sp, #0x20
sub         sp, sp, #0x3c0
sub         x8, x29, #0x48
orr         x9, xzr, #0x7fffffffffffffff
str         x0, [x8, #0x20]
str         x1, [x8, #0x18]
adrp        x0, #0x100146000 ; 0x10012f0b8@PAGE
orr         w10, wzr, #0x1
```

Figura 6 - Parte iniziale del metodo a932821303 del codice disassemblato delle'eseguibile del trojan

Inserendo un *breakpoint* su quest'ultimo, ed eseguendolo un'istruzione alla volta, si è individuata all'offset +2096 nel registro x2 la proprietà (variabile) contente una stringa codificata in 'base64' così come mostrato in figura 7. *a932821303* -subito a valle della proprietà sopracitata- viene invocato il metodo *decryptAndDecodeString:withKey:* il cui valore di ritorno all'offset +2018 nel registro xo è una stringa codificata in esadecimale (fig. 8).

```
(lldb) po $x2
AwF/pceBxcfAB+nwN3ATBNkF7njumTKCZdfbgc+tuOnqnP+3WbBJwnR19xHjlfw+9JtHDe20bHykphLcdEnBAeg58e
QugTOJ0Wy//cRqrwBztZd+YKynD+RHFzJmTG4OwzTRIrJ+pka19mWe8HrZR4XaK97ptity3K6mfEi3Z9u385sUhgyq
eHrL838zzaRJeCJ7dxSwDNBDmnFk+yHSbiEqbqZ/OS3GoI8ra5klfNjQptuCnv6z1cLUG0oAS5+pEfAdFMN9NRMcKx
m3SePrTD4QOs7yH/BiZ1Ly+YUm4TfQOsCOaU74oe2QI6PJ36/v138zbqV2x5AZe7zQ53s5d1+RrbHhYpmqHthgn2sf
cwvb4Va4azGhjOTjUKUqedwmXYWyGUsTlBsdKl/w46xIpaVdOPrOolOVAWJFPKZrmGr8azBoxNXwld2ovRaeQxDmKk
```

Figura 7 - Porzione del valore del registro x2 alla posizione +2096 del metodo a932821303

Il valore decodificato di questa stringa però è un oggetto criptato, quindi per leggerne il valore in "chiaro" è necessaria sia una decodifica che la decrittazione. Per fare ciò, all'interno di Copiando e incollando il valore di tale stringa su uno "strumento" di conversione online[7] è stato possibile decodificare il valore da esadecimale in ASCII, ottenendo come risultato quanto riportato in figura 9, cioè un file in formato *plist* contenente una serie di coppie di chiavi e valori. L'analisi di questo file non solo

da parte dello sviluppatore. Tale attività, infatti non permette a Hopper di risolvere tutti i simboli e per questo motivo, dove non riesce a risolvere ne crea di dinamici dandogli nomi pseudo casuali.

7 https://www.rapidtables.com/convert/number/hex-to-ascii.html

```
(lldb) po $x3
6749442272

(lldb) po $x0
<3c3f786d 6c207665 7273696f 6e3d2231 2e302220 656e636f 64696e67 3d225554 462d3822 3f3e0a3c 21
06c652f 2f445444 20504c49 53542031 2e302f2f 454e2220 22687474 703a2f2f 7777772e 6170706c 652e
46422 3e0a3c70 6c697374 20766572 73696f6e 3d22312e 30223e0a 3c646963 743e0a09 3c6b6579 3e6175
63c 2f737472 696e673e 0a093c6b 65793e62 756e646c 65645572 6c3c2f6b 65793e0a 093c7374 72696e67
f 62757369 6e657373 2f686f6d 65627573 696e6573 733c2f73 7472696e 673e0a09 3c6b6579 3e636861 6
74727565 3c2f7374 72696e67 3e0a093c 6b65793e 6368756e 6b73697a 653c2f6b 65793e0a 093c7374 726
```

Figura 8 - Estratto del valore del registro x0 alla posizione +2108 del metodo a932821303

ha permesso l'individuazione dell'IP e la porta del server con il quale il captatore scambiava dati, ma anche l'individuazione dei percorsi dove il trojan leggeva il file di configurazione contente la schedulazione di quando il microfono dello smartphone doveva essere attivato e disattivato e il percorso dove caricava i file audio registrati.

Un'ulteriore verifica effettuata con registrazione del traffico di rete generato dall'app stessa - con l'ausilio del software Wireshark[8] - ha confermato anche gli stessi valori di IP e porta, così come rilevati sul file sopra indicato, arrivando quindi a fornire conferma incrociata della bontà dell'analisi svolta.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>audiocodec</key>
        <string>m4af</string>
        <key>bundledUrl</key>
        <string>[redacted]</string>
        <key>channelEncypted</key>
        <string>true</string>
        <key>chunksize</key>
        <string>262144</string>
        <key>cm</key>
        <string>[redacted]/</string>
        <key>cmd</key>
        <string>[redacted]</string>
        <key>compress</key>
        <string>true</string>
        <key>gps</key>
        <string>disabled</string>
        <key>inetAddr</key>
        <string>[redacted]</string>
        <key>inetPort</key>
        <string>8443</string>
        <key>room</key>
        <string>[redacted]</string>
        <key>upl</key>
        <string>[redacted]</string>
        <key>ws</key>
        <string>[redacted]</string>
        <key>wsEnabled</key>
        <true/>
</dict>
</plist>
```

Figura 9 - Risultato della Conversione del valore del registro x0 alla posizione +2108 del metodo a932821303 da Esadecimale ad Ascii

Benché il tipo di approccio adottato si riveli dispendioso in termini di tempo necessario e di attività da svolgere, i risultati ottenuti hanno consentito di rispondere alle domande poste e avere una migliore conoscenza delle modalità di inoculazione e funzionamento di un captatore informatico, e probabilmente anche di suggerire un possibile metodo - quanto meno- di individuazione di un software "anomalo" in un dispositivo di questa tipologia." ©

8 https://www.wireshark.org/