

Rapport : IOS Reverse

Projet « Stop Covid or it will be a BOOM»

18/09/2020



Client

Mathieu Renard

Auteur

Yanis ALIM

Arezki MEHADDI

Iurie TOMA

SOMMAIRE

INTRODUCTION	2
Objectif	2
Contenu	3
1. Questions de cours	4
2. Countdown	7
Description de l'application	7
Identification des fichiers à analyser	7
Identification des fichiers intéressants pour l'analyse	7
INFO.PLIST	7
MACH-O	7
Fonction de sécurité identifiée	9
Workflow	9
Recommandation	15
3. StopCovid	16
Description de l'application	16
Package IPA	16
Identification des fichiers à analyser	16
Identification des fichiers intéressants pour l'analyse	16
INFO.PLIST	17
MACH-O	17
Fonction de sécurité identifiée	19
Workflow	20
Recommandation	21

Introduction

A. Objectif

Le but de ce TP est d'analyser deux applications installées sur un téléphone iPhone5s fourni par le formateur.

Le nom des applications sont les suivantes :

- ❖ StopCovid
- ❖ Countdown

Certaines de ces applications peuvent être analysées statiquement.

B. Contenu

Ce document regroupe les résultats de l'analyse de sécurité des deux applications mises à disposition en prenant soin de détailler le workflow complet, les fonctions de sécurité de chaque application et pour finir les remédiations possibles.

1. Questions de cours

1. Citez 3 mécanismes de sécurité sur iOS.

Plusieurs mécanismes de sécurité sont présent sur l'iOS, pour en citer trois, il existe :

- Le profil sandbox unique à chaque application
- Framework de contrôle d'accès obligatoire (MACF) : méthode de gestion des droits des utilisateurs pour l'usage de systèmes d'information
- Signature de code obligatoire et l'Application de la signature de code au moment de la compilation

2. Qu'est-ce que le jailbreak ?

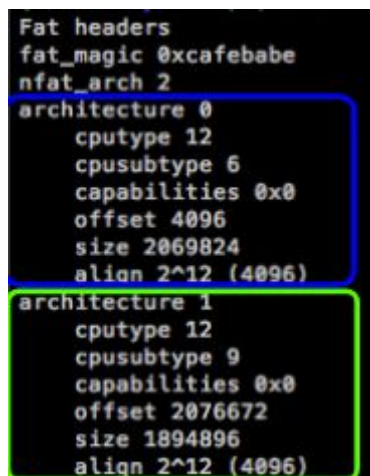
Le jailbreak consiste en l'escalade des privilèges d'un terminal iOS afin de parvenir éliminer les restrictions logicielles imposées le constructeur sur ses OS. Le jailbreak offre un accès ROOT dans le système d'exploitation, permettant ainsi d'installer des applications provenant de sources différentes de l'App Store.

3. Quelle est le format des fichiers binaires iOS ?

Le format des fichiers binaires iOS est **MACH-O**.

4. Qu'est-ce qu'un header FAT ?

Le header FAT agit comme wrapper qui regroupe différents type d'architecture CPU. (arm64, arm7, etc.). Ils contiennent plusieurs objets mach dans un seul fichier et chacun a une architecture ou une plate-forme différente.



```
Fat headers
fat_magic 0xcafebabe
nfat_arch 2
architecture 0
  cputype 12
  cpusubtype 6
  capabilities 0x0
  offset 4096
  size 2069824
  align 2^12 (4096)
architecture 1
  cputype 12
  cpusubtype 9
  capabilities 0x0
  offset 2076672
  size 1894896
  align 2^12 (4096)
```

5. Quel est le type d'architecture du SoC Apple A8 ?

Le SoC Apple A8 repose sur une architecture **Arm64**.

6. Quel est le magic d'un fichier binaires iOS mono architecture ?

Le magic d'un fichier binaire iOS mono architecture est **0xfeedface**.

```
$otool -h hello
hello:
Mach header
  magic cputype cpusubtype  caps   filetype ncmds sizeofcmds  flags
  0xfeedface    12        9    0x00        2     12      572 0x00000085
```

7. Qu'est-ce qu'un fichier plist ?

Un fichier plist, d'anglais "Property List", est un fichier texte recensant des informations généralement utile au lancement de l'application. Ce fichier, structuré en XML comporte un nœud XML racine, qui est un dictionnaire composé de l'ensemble des clés et des valeurs persistantes.

8. Quels sont les registres utilisés pour passer les arguments d'une fonction sur une architecture arm64 ?

Sur une architecture arm64, les registres de **x0 à x7** sont utilisés comme arguments lors d'appel à une fonction.

9. Qu'est-ce que le Keychain ?

Le Keychain est le système de gestion des mots de passe sous macOS. Un Keychain peut avoir différents types de données : mots de passe (pour les sites Web, les serveurs FTP, les comptes SSH, les partages réseau, les réseaux sans fil, les applications de groupware, les images de disque cryptées), notes sécurisées, certificats et clés privées.

10. Qu'est-ce que le paradigme de passage par message ?

Le paradigme de passage par message est le fait que tout appel de méthode d'une classe est un passage de message. Pour appeler un message sur un objet, on place entre crochet l'objet puis le message. On peut chaîner les appels très facilement. Ainsi dans l'exemple, *methode* renvoie un objet et sur cet objet on appelle *methode2*.

```
// passage de message  
[objet methode:argument];  
// chaînage de message  
[[objet methode:argument] methode2:argument2];
```

2. Countdown

1. Description de l'application

Dans le film Die Hard With A Vengeance, les personnages John McClane et Zeus Carver ouvrent une mallette pour découvrir que ce faisant, il a armé une bombe puissante. Il explosera en quelques minutes à moins qu'ils ne puissent désarmer. À l'intérieur de la mallette, il y a une balance. Ils ont à leur disposition deux cruches: « L'une contient exactement 5 litres et l'autre exactement 3 litres. Pour désarmer la bombe, ils doivent remplir la cruche de 5 litres avec exactement quatre litres d'eau et la placer sur la balance. Quelques grammes de trop ou trop peu feront exploser la bombe. L'eau peut être obtenue à proximité d'une fontaine. Ici, la bombe est dans votre smartphone. L'application Countdown est donc la bombe à désamorcer.

a. Identification des fichiers à analyser

Nom du fichier	Version	Taille (octet)	SHA256
CountDown.ipa	N/A	3 021 053	9ea1a1028fde5f832ec45fe895e7376d04441431d4c242f23ff022bb575d8a0b

b. Identification des fichiers intéressants pour l'analyse

Afin de pouvoir collecter le plus d'information possible sur l'application, une première étape consiste à identifier les fichiers les plus pertinents.

INFO.PLIST

Après examen du fichier **Info.plist**, nous retrouvons le nom du binaire qui nous intéresse : **CountDown**.

```
<key>CFBundleExecutable</key>  
<string>CountDown</string>
```

MACH-O

a. Header

```
iurie@MacBook-Pro ~/M/M/i/e/C/CountDown> ~/Downloads/jtool -h -arch arm64 CountDown
Magic: 64-bit Mach-O
Type: executable
CPU: ARM64
Cmds: 22
size: 3280 bytes
Flags: 0x200085
iurie@MacBook-Pro ~/M/M/i/e/C/CountDown> ~/Downloads/jtool -h -arch armv7 CountDown
Magic: 32-bit Mach-O
Type: executable
CPU: ARMv7
Cmds: 22
Size: 2784 bytes
Flags: 0x200085
```

L'en-tête Mach-O nous informe qu'il existe deux versions de l'application qui est empaquetée dans le fichier IPA. Une version 32bits (armv7) et une version 64 bits (armv8).

b. Chiffrement du fichier (CRYPTID)

L'application n'est pas protégée par DRM. On en conclut donc que cette application ne provient pas de l'Apple Store. La présence du fichier "embedded.mobileprovision" valide cette affirmation.

```
Yanix0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ ARCH=armv7 jtool2 -l CountDown | egrep -i "LC_ENCRYPTION_INFO" | sed "s/.*Encryption: \\([0-1]\\).*/\\1/"
0
Yanix0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ ls
Base.lproj  bomb_large.png  boom_large1.png  _CodeSignature  CountDown  embedded.mobileprovision  flag.enc  Info.plist  PkgInfo
Yanix0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ _
```

c. Signature de l'application

```
iurie@MacBook-Pro ~/M/M/i/e/C/CountDown [3]> ~/Downloads/jtool --sig -arch armv7 CountDown
Blob at offset: 417280 (24576 bytes) is an embedded signature
Code Directory (2262 bytes)
  Version: 20400
  Flags: none
  Identifier: org.gotohack.CountDown
  CDHash: f01f1a6c849c6c1c3ab0ea7a41d39517a48b12e0
  # of Hashes: 102 code + 5 special
  Hashes @222 size: 20 Type: SHA-1
Requirement Set (188 bytes) with 1 requirement:
  0: Designated Requirement (@20, 156 bytes): SIZE: 156
Ident(org.gotohack.CountDown) AND Apple Generic Anchor
Cert field
Unknown opcode 7375626a - has Apple changed the op codes?Please notify!!
False Info.plist
Entitlements (504 bytes) (use --ent to v
```

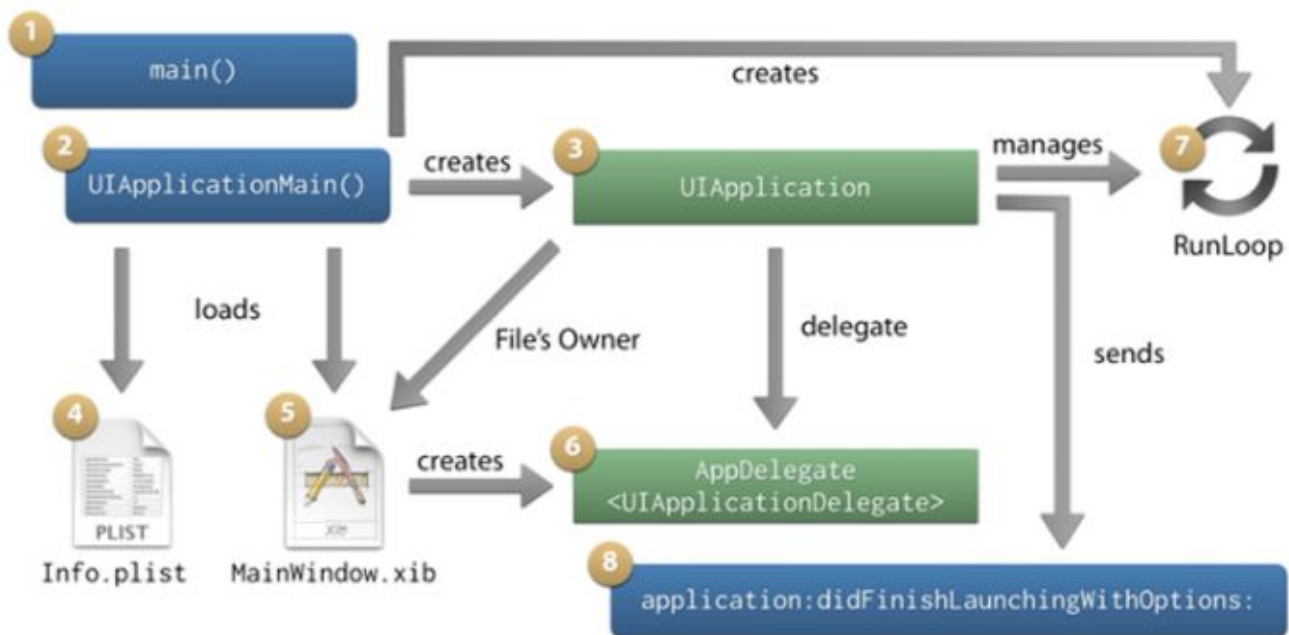

2. Fonction de sécurité identifié

En utilisant l'option d'analyse de fonction de **JTOOL**, on peut identifier des fonctions de sécurité utilisés par l'application :

```
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ ARCH=arm64 jtool2 --analyze CountDown
Analyzing file...
processLoadCommands: Not a Mach-O magic (0xbebafeca)
Resolving stubs..
Not ARM64 - will not resolve stubs..
Processing __DATA..
opened companion file ./CountDown.ARM64.229624AC-0001-3834-9587-4E036D7C417E
Dumping symbol cache to file
Symbolicated 132 symbols and 0 functions
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ egrep "*rypt*" CountDown.ARM64.229624AC-0001-3834-9587-4E036D7C417E
0x1000063d4|-[EncryptedFileURLProtocol startLoading]|
0x100006608|-[EncryptedFileURLProtocol stopLoading]|
0x1000066c8|-[EncryptedFileURLProtocol stream:handleEvent:]|
0x100006968|-[EncryptedFileURLProtocol .cxx_destruct]|
0x100006080|_CCCryptorCreate|
0x100006088|_CCCryptorFinal|
0x100006090|_CCCryptorRelease|
0x100006098|_CCCryptorUpdate|
0x100006cd0|EncryptedFileURLProtocol|
0x100061dc8|_OBJC_CLASS_$_EncryptedFileURLProtocol|
0x100061ea0|EncryptedFileURLProtocol|
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/CountDown/Payload/CountDown.app$ _
```

3. Workflow

Voici un workflow générale d'une application iOS:



Sur IDA PRO, on va voir notre **ENTRYPOINT** qui est la fonction start :

```

1  int64 fastcall start( int64 a1, int64 a2)
2  {
3      int64 v2; // x19
4      int64 v3; // x20
5      int64 v4; // x21
6      void *v5; // x0
7      int64 v6; // x0
8      int64 v7; // x22
9      int64 v8; // x19
10
11     v2 = a2;
12     v3 = a1;
13     v4 = objc_autoreleasePoolPush();
14     v5 = objc_msgSend(&OBJC_CLASS__AppDelegate, "class");
15     v6 = NSStringFromClass(v5);
16     v7 = objc_retainAutoreleasedReturnValue(v6);
17     v8 = UIApplicationMain(v3, v2, 0LL, v7);
18     objc_release(v7);
19     objc_autoreleasePoolPop(v4);
20     return v8;
21 }

```

On voit bien qu'il y a un appel (message) à la classe **AppDelegate** :

```

2  void __cdecl -[ViewController viewDidLoad:](ViewController *self, SEL a2, bool a3)
3  {
4      -[ViewController countdownTimer](self, "countdownTimer", a3);
5      dword_100061FC0 = 10;
6  }

```

Cette dernière appel **countdownTimer** :

```

1  void __cdecl -[ViewController countdownTimer](ViewController *self, SEL a2)
2  {
3      ViewController *v2; // x19
4      void *v3; // x0
5      int64 v4; // x0
6      NSTimer *v5; // x8
7
8     v2 = self;
9     v3 = objc_msgSend(
10         &OBJC_CLASS__NSTimer,
11         "scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:",
12         self,
13         "updateCounter:",
14         0LL,
15         1LL,
16         1.0);
17     v4 = objc_retainAutoreleasedReturnValue(v3);
18     v5 = v2->timer;
19     v2->timer = (NSTimer *)v4;
20     objc_release(v5);
21 }

```

CountdownTimer envoie des paramètres à la fonction **updateCounter** avec un planificateur de temps qui se rafraîchit chaque seconde :

```

15 | v3 = self;
16 | v4 = dword_100061FC0 - 1;
17 | if ( dword_100061FC0 < 1 )
18 | {
19 |     objc_msgSend((void *)self->timer, "invalidate", a3);
20 |     v13 = v3->timer;
21 |     v3->timer = 0LL;
22 |     objc_release(v13);
23 |     -[ViewController gameOver](v3, "gameover");
24 | }
25 | else
26 | {
27 |     --dword_100061FC0;
28 |     v5 = v4 / 3600;
29 |     dword_1000621A0 = v4 / 3600;
30 |     v6 = v4 % 3600;
31 |     v7 = v6 / 60;
32 |     dword_1000621A4 = v6 / 60;
33 |     v8 = (unsigned int)(v6 % 60);
34 |     dword_1000621A8 = v8;
35 |     v9 = objc_msgSend(&OBJC_CLASS__NSString, "stringWithFormat:", CFSTR("%02d:%02d:%02d"), v5, v7, v8);
36 |     v10 = objc_retainAutoreleasedReturnValue(v9);
37 |     v12 = (void *)objc_loadWeakRetained(&v3->_CounterValue, v11);
38 |     objc_msgSend(v12, "setText:", v10);
39 |     objc_release(v12);
40 |     objc_release(v10);
41 | }
42 | }

```

00075DF4 -[ViewController updateCounter:]:22 (100005DF4)

UpdateCounter, quant à elle, décompte chaque seconde tant que le compteur est différent de 0. Lorsque le compteur arrive à 0, un appel à la fonction **gameover** est fait :

```

1 | void __cdecl -[ViewController gameOver](ViewController *self, SEL a2)
2 | {
3 |     ViewController *v2; // x19
4 |     void *v3; // x0
5 |     void *v4; // x0
6 |     void *v5; // x20
7 |     void *v6; // x0
8 |     __int64 v7; // x21
9 |
10 | v2 = self;
11 | v3 = objc_msgSend(&OBJC_CLASS__UIStoryboard, "storyboardWithName:bundle:", CFSTR("Main"), 0LL);
12 | v4 = (void *)objc_retainAutoreleasedReturnValue(v3);
13 | v5 = v4;
14 | v6 = objc_msgSend(v4, "instantiateViewControllerWithIdentifier:", CFSTR("Kaboom"));
15 | v7 = objc_retainAutoreleasedReturnValue(v6);
16 | objc_msgSend(v2, "presentViewController:animated:completion:", v7, 0LL, 0LL);
17 | objc_release(v7);
18 | objc_release(v5);
19 | }

```

Cette fonction gameover annonce l'échec par l'affichage du message : Kaboom.



Au final, on peut identifier les fonctions qui déclenchent la bombe :

- ❖ **countdownTimer**,
- ❖ **CounterValue**,
- ❖ **setCounterValue**,
- ❖ **updateCounter**,
- ❖ **gameover** (la bombe explose),
- ❖ **hero** (la bombe a été désamorcé).

Désamorçage :

En parallèle du timer, une fois l'application chargée, l'utilisateur pourra taper un code de désamorçage.

```

1 void __cdecl -[ViewController disarm:](ViewController *self, SEL a2, id a3)
2 {
3     ViewController *v3; // x19
4     NSTimer *v4; // x0
5     UITextField *v5; // x0
6     void *v6; // x0
7     void *v7; // x20
8     void *v8; // x0
9     __int64 v9; // x21
10
11     v3 = self;
12     objc_msgSend((void *)self->timer, "invalidate", a3);
13     v4 = v3->timer;
14     v3->timer = 0LL;
15     objc_release(v4);
16     v5 = -[ViewController DisarmCode](v3, "DisarmCode");
17     v6 = (void *)objc_retainAutoreleasedReturnValue(v5);
18     v7 = v6;
19     v8 = objc_msgSend(v6, "text");
20     v9 = objc_retainAutoreleasedReturnValue(v8);
21     -[ViewController testDisarmCode:](v3, "testDisarmCode:", v9);
22     objc_release(v9);
23     objc_release(v7);
24 }

```

Fonction qui teste le code de désamorçage

le code entré par l'utilisateur

Ce code de désamorçage est testé au sein de la fonction **disarm**. Dans cette même fonction **disarm**, on retrouve la fonction **testDisarmCode** :

```

21 user_input = input;
22 v4 = self;
23 v5 = objc_retain(input, a2);
24 v6 = objc_msgSend(CFSTR("only for real entropy bytes!"), "dataUsingEncoding:", 4LL);
25 v7 = objc_retainAutoreleasedReturnValue(v6);
26 v18 = unk_10000A078;
27 v19 = unk_10000A088;
28 encoded_input = objc_msgSend(user_input, "dataUsingEncoding:", 4LL);
29 v9 = objc_retainAutoreleasedReturnValue(encoded_input);
30 objc_release(v5);
31 v10 = objc_msgSend(&OBJC_CLASS__NSMutableData, "dataWithLength:", 32LL);
32 v11 = objc_retainAutoreleasedReturnValue(v10);
33 v12 = (void *)objc_retainAutorelease(v9);
34 objc_msgSend(v12, "bytes");
35 objc_msgSend(v12, "length");
36 v13 = (void *)objc_retainAutorelease(v7);
37 objc_msgSend(v13, "bytes");
38 objc_msgSend(v13, "length");
39 v14 = (void *)objc_retainAutorelease(v11);
40 objc_msgSend(v14, "mutableBytes");
41 v17 = objc_msgSend(v14, "length");

```

Cette fonction prend l'entrée de l'utilisateur, l'encode, puis prends le string **"only for real entropy bytes!"** et fait, pour ces deux chaînes de caractère la même opération : **dériver la taille et les valeurs en byte**.

```

42 if ( (unsigned int)CCKeyDerivationPBKDF() )
43 {
44     NSLog(CFSTR("PBKDF2 Error..."));
45 }
46 else
47 {
48     v15 = (void *)objc_alloc(&OBJC_CLASS__NSData);
49     v16 = objc_msgSend(v15, "initWithBytes:length:", &v18, 32LL, v17);
50     if ( (unsigned int)objc_msgSend(v16, "isEqualToData:", v14) )
51         -[ViewController hero](v4, "hero");
52     objc_release(v16);
53 }
54 -[ViewController gameover](v4, "gameover", v17);
55 objc_release(v14);
56 objc_release(v12);
57 objc_release(v13);
58 }

```

A la suite de ces opération, un appel vers `CCKeyDerivationPBKDF()` est effectué, qui va chiffrer l'entrée à l'aide d'une clé et comparer le résultat avec la bonne entrée, se trouvant à ligne 50, qui est équivalente en assembleur à :

```

text:00000000100005C14      MOV     X23, X0
text:00000000100005C18      NOP
text:00000000100005C1C      LDR     X1, =sel_isEqualToData_ ; "isEqualToData:"
text:00000000100005C20      MOV     X2, X22
text:00000000100005C24      BL      _objc_msgSend
text:00000000100005C28      CBZ     W0, loc_100005C3C
text:00000000100005C2C      NOP
text:00000000100005C30      LDR     X1, =sel_hero ; "hero"
text:00000000100005C34      MOV     X0, X19 ; void *
text:00000000100005C38      BL      _objc_msgSend ; -[ViewController hero]
text:00000000100005C3C      loc_100005C3C      ; CODE XREF: -[ViewController testDisarmCode:]
text:00000000100005C3C      MOV     X0, X23
text:00000000100005C40      BL      _objc_release

```

Le résultat va donc dépendre de la valeur du registre `w0`.

Patcher avec Frida :

Prenons l'instruction se trouvant à l'adresse **0x0005C28** et voyons sa valeur dans le registre **w0** :

```

Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ cat patch.js
var targetModule = 'CountDown';
var addr = ptr(0x5C28);
var moduleBase = Module.getBaseAddress(targetModule);
var targetAddress = moduleBase.add(addr);
Interceptor.attach(targetAddress, {
    onEnter: function(args) {
        console.log('At the address ' + addr + ' the value is currently ' + this.context.x0);
    },
});

```

On lance le script avec frida :

```

/ _ _ |   Frida 12.11.15 - A world-class dynamic instrumentation toolkit
| ( _ |   Commands:
> _ _ |   help      -> Displays the help system
/_/_|_ |   object?   -> Display information about 'object'
. . . .   exit/quit -> Exit
. . . .   More info at https://www.frida.re/docs/home/

[iOS Device::CountDown]-> At the address 0x5c28 the value is currently 0x0
-

```

On observe ici que la valeur du registre est à **0x00**.

Afin de sauter directement vers la fonction **hero** plutôt la fonction **gameover**, il suffit donc de changer cette valeur de registre pour avoir : **0x01**.

```

Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ cat patch.js
var targetModule = 'CountDown';
var addr = ptr(0x5C28);
var moduleBase = Module.getBaseAddress(targetModule);
var targetAddress = moduleBase.add(addr);
Interceptor.attach(targetAddress, {
  onEnter: function(args) {
    if(this.context.x0 == 0x00){
      this.context.x0=0x01
      console.log("[*] w0 = 0x01");
    }
  },
});

```

```

Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ frida -U -l patch.js CountDown

/ _ _ |   Frida 12.11.15 - A world-class dynamic instrumentation toolkit
| ( _ |   Commands:
> _ _ |   help      -> Displays the help system
/_/_|_ |   object?   -> Display information about 'object'
. . . .   exit/quit -> Exit
. . . .   More info at https://www.frida.re/docs/home/

[iOS Device::CountDown]-> [*] w0 = 0x01

```

Script Frida :

```
/*  
  
// read the value of w0  
  
var targetModule = 'CountDown';  
var addr = ptr(0x5C28);  
var moduleBase = Module.getBaseAddress(targetModule);  
var targetAddress = moduleBase.add(addr);  
Interceptor.attach(targetAddress, {  
    onEnter: function(args) {  
        console.log('At the address ' + addr + ' the value is currently ' + this.context.x0);  
    },  
});  
  
*/  
  
// set the value of w0  
  
var targetModule = 'CountDown';  
var addr = ptr(0x5C28);  
var moduleBase = Module.getBaseAddress(targetModule);  
var targetAddress = moduleBase.add(addr);  
Interceptor.attach(targetAddress, {  
    onEnter: function(args) {  
        if(this.context.x0 == 0x00){  
            this.context.x0=0x01  
            console.log("Bypass Test1");  
        }  
    },  
});
```

Une fois cette opération faite, on relance l'application, on entre un code au hasard, et nous arrivons enfin à désamorcer cette bombe !



4. Recommandation

Plusieurs recommandation peuvent être faite afin d'éviter le bypass du mot de passe et ainsi empêcher tout désamorçage de la bombe (et par la même occasion, désintégrer tout être présent à un rayon de 50 mètres du téléphone) :

- ❖ Ne pas laisser le code de désamorçage en clair dans le code,
- ❖ Utiliser l'obfuscation au maximum afin de ralentir la progression de l'attaquant,
- ❖ Protéger l'application avec du chiffrement afin de rendre sa compréhension encore plus difficile.

2. StopCovid

1. Description de l'application

StopCovid est une application qui s'inscrit dans le plan global de déconfinement du Gouvernement dans le contexte de l'épidémie de **Covid-19**. Les médecins et les plateformes de l'Assurance Maladie assurent la mission de détection des contacts des personnes malades afin de rompre les chaînes de transmission. **StopCovid** se veut être un rempart supplémentaire contre le virus et vient compléter l'action de ces équipes et permet à chaque usager, sur la base du volontariat, de savoir s'il a eu un contact rapproché avec une personne malade ou de prévenir les autres utilisateurs s'il a été diagnostiqué comme un cas de **Covid-19**.

Selon les développeurs, l'application **StopCovid** est complètement anonyme. Elle génère seulement des pseudonymes (crypto-identifiants éphémères) qui ne sont pas associés à l'identité de l'utilisateur. Personne, pas même l'État, n'a accès à l'identité des utilisateurs.

Plusieurs questions se posent : S'agit-il de l'application officielle ? Quelles sont les données sauvegardées par l'application ? Où et comment ? Afin de répondre à ces questions, notre analyse portera sur l'ensemble du bundle de l'application StopCovid.

2. Package IPA

Identification des fichiers à analyser

Nom du fichier	Version	Taille (octet)	SHA256
StopCovid.ipa	99	30 600 122	009258096f5170f73c501f9faca0f49f8675ed8cff7c0739b210eb4c2994958f
StorageSDK	N/A	134 464	d292555d99ae5ee6f240857055b0d2ef6f1d62c4e2e126539091309169997452

Identification des fichiers intéressants pour l'analyse

Afin de pouvoir collecter le plus d'information possible sur l'application, une première étape consiste à identifier les fichiers les plus pertinents.

INFO.PLIST

Le fichier **Info.plist** indique qu'il existe qu'une seule version qui est pour l'architecture arm64. Plus d'informations peuvent être trouvés quant à la compatibilité de l'application, la version, etc.

```
<string>StopCovid</string>
<key>DTCompiler</key>
<string>com.apple.compilers.llvm.clang.1_0</string>
<key>UIRequiredDeviceCapabilities</key>
<array>
  <string>arm64</string>
</array>
<key>CFBundleDevelopmentRegion</key>
<string>en</string>
<key>CFBundleVersion</key>
<string>99</string>
<key>BuildMachineOSBuild</key>
```

MACH-O

Header

```
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ jtool2 -h StopCovid
Magic: 64-bit MachO (Little Endian)
Type: executable
CPU: ARM64 (ARMv8)
Cmds: 50
Size: 5648
Flags: 0x200085
```

L'en-tête Mach-O nous informe qu'il existe une version de l'application qui est empaquetée dans le fichier IPA : une version Arm64 (armv8).

Chiffrement du fichier (CRYPTID)

```
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ jtool2 -l StopCovid | egrep -i "LC_ENCRYPTION_INFO" | cut -d " " -f 4,5
Encryption: 0
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ _
```

```
CVNavigationChild.storyboardc
de.lproj
embedded.mobileprovision
```

L'application n'est pas protégée par FairPlay (non chiffré). On en conclut donc que cette application ne provient pas de l'Apple Store. La présence du fichier "embedded.mobileprovision" valide cette affirmation.

Signature de l'application

L'application est signé par **org.gotohack.stopcovid.ios** néanmoins l'application officiel stop covid récupéré sur AppStore comporte une signature différente, **Identifiant: fr.gouv.stopcovid.ios**. On en déduit que la signature a été modifié.

```
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ jtool2 --sig StopCovid
An embedded signature with 4 blobs:
Code Directory (8542 bytes)
  Version: 20400
  Flags: none
  CodeLimit: 0x101840
  Identifier: org.gotohack.stopcovid.ios (@0x58)
  Team ID: KCWS38TNMR (@0x73)
  Executable Segment: Base 0x00000000 Limit: 0x00000000 Flags: 0x00000000
  CDHash: 369c43f82bfe19e776edd9771cd8d6c4aeed0cdca6b6c5d8245b0e009f3b20e (computed)
  # of hashes: 258 code (4K pages) + 5 special
  Hashes @286 size: 32 Type: SHA-256
Requirement Set (192 bytes) with 1 requirement:
  0: Designated Requirement (@20, 160 bytes): Ident(org.gotohack.stopcovid.ios) AND Apple Generic Anchor Cert field [subject.CN] = 'Apple Distribution: M
athieu RENARD (KCWS38TNMR)' AND (Cert Generic[1] = WWD Relations CA)
Entitlements (458 bytes) (use --ent to view)
Blob Wrapper (4749 bytes) (0x10000 is CMS (RFC3852) signature)
  CA: Apple Certification Authority CN: Apple Root CA
  CA: Apple Worldwide Developer Relations CN: Apple Worldwide Developer Relations Certification Authority
  CA: Apple Certification Authority CN: Apple Root CA
  CA: Apple Certification Authority CN: Apple Root CA
  Timestamp: 20:37:21 2020/09/17
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ _
```

```
iurie@MacBook-Pro ~/D/s/stopCOVID> ~/Downloads/jtool --sig -arch armv7 StopCovid
Blob at offset: 975792 (44224 bytes) is an embedded signature
Code Directory (9829 bytes)
  Version: 20500
  Flags: none
  Identifier: fr.gouv.stopcovid.ios
  CDHash: 0a4d3150ce019db60d8e779a4b37844baeb824e5
  # of Hashes: 239 code + 7 special
  Hashes @5049 size: 20 Type: SHA-1
Requirement Set (104 bytes) with 1 requirement:
  0: Designated Requirement (@20, 72 bytes): SIZE: 72
Apple Generic Anchor
Cert Generic
Unknown opcode 2a864886 - has Apple changed the op codes?Please notify J!
False Info plist
Entitlements (415 bytes) (use --ent to view)
Magic: fade7172 (Unknown/Unhandled: Please tell J)
Code Directory (15649 bytes)
  Version: 20500
  Flags: none
  Identifier: fr.gouv.stopcovid.ios
  CDHash: f59704e2247191efee9c6dd7c5b1ac5df70595bf
  # of Hashes: 239 code + 7 special
  Hashes @8001 size: 32 Type: SHA-256
```

3. Fonction de sécurité identifié

En utilisant l'option d'analyse de fonction de **JTOOL**, on peut identifier des fonctions de sécurité utilisés par l'application :

```
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ jtool2 --analyze StopCovid
Analyzing file...
Resolving stubs..
Processing __DATA..
opened companion file ./StopCovid.ARM64.4EE71842-B1D4-33A5-BFC4-F8BEA94FAB2C
Dumping symbol cache to file
Symbolicated 3333 symbols and 3098 functions
Yan1x0s@1337:~/0x00/2020/iOS/exam/exam2020/StopCovid/Payload/StopCovid.app$ egrep "*rypt*" StopCovid.ARM64.4EE71842-B1D4-33A5-BFC4-F8BEA94FAB2C
0xd348|_$_s10Foundation4DataV7SwCryptE17hexadecimalStringSSyF|
0x1000ac168|_$_s10Foundation4DataV7SwCryptE17hexadecimalStringSSyF|
0x1000cc508|_$_s10Foundation4DataV7SwCryptE17hexadecimalStringSSyF|
```

3. Workflow

Notre entry point dans l'application est la fonction **_main**. Cette fonction initialise un garbage collector puis appelle la méthode `_UIApplicationMain` :

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
    __int64 v3; // x0
    void *v4; // x0
    id v5; // x21

    v3 = sub_100054954(0LL);
    v4 = (void *)NSStringFromClass(v3);
    v5 = objc_retainAutoreleasedReturnValue(v4);
    UIApplicationMain();
    objc_release(v5);
    return 0;
}
```

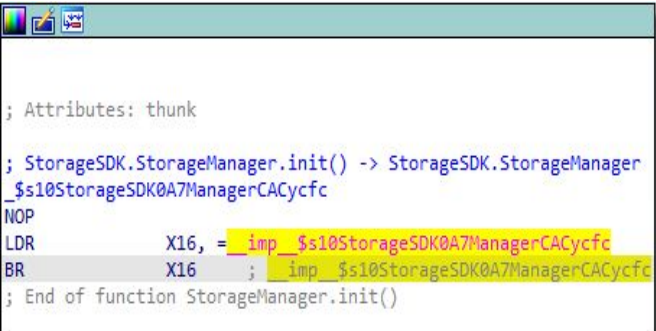
Nous nous dirigeons donc vers la méthode `_UIApplicationMain`, mais qui malheureusement ne donne rien d'intéressant :

```
int64 UIApplicationMain()
{
    return _UIApplicationMain();
}
```

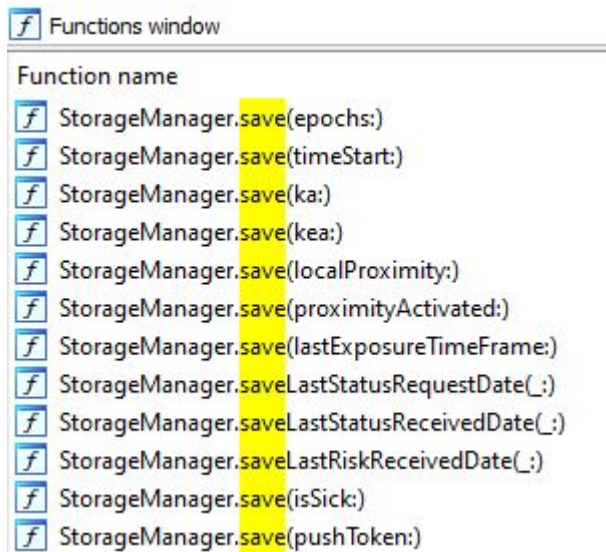
Dans le bundle de l'application, on retrouve les frameworks de **StorageSDK**, Realm utiliser pour stocker l'information dans une base de données SQL et **RobertSDK**. Intéressons nous un peu plus à **StorageSDK** :

Realm.framework	17/09/2020 22:37	Dossier de fichiers
RealmSwift.framework	17/09/2020 22:37	Dossier de fichiers
RobertSDK.framework	17/09/2020 22:37	Dossier de fichiers
ServerSDK.framework	17/09/2020 22:37	Dossier de fichiers
StorageSDK.framework	18/09/2020 13:43	Dossier de fichiers
SwCrypt.framework	17/09/2020 22:37	Dossier de fichiers

On retrouve Storage SDK dans la classe `StorageManager.init()` :

<code>static _DictionaryStorage.copy(original:)</code>	<code>_stubs</code>	000	 <pre> ; Attributes: thunk ; StorageSDK.StorageManager.init() -> StorageSDK.StorageManager ; \$s10StorageSDK0A7ManagerCACycfc NOP LDR X16, =_imp__\$s10StorageSDK0A7ManagerCACycfc BR X16 ; _imp__\$s10StorageSDK0A7ManagerCACycfc ; End of function StorageManager.init() </pre>
<code>static _DictionaryStorage.allocate(capacity:)</code>	<code>_stubs</code>	000	
<code>_DataStorage.replaceBytes(in:with:length:)</code>	<code>_stubs</code>	000	
<code>_DataStorage.init(bytes:length:)</code>	<code>_stubs</code>	000	
<code>_DataStorage._offset.getter</code>	<code>_stubs</code>	000	
<code>_DataStorage._length.getter</code>	<code>_stubs</code>	000	
<code>_DataStorage._bytes.getter</code>	<code>_stubs</code>	000	
<code>StorageManager.init()</code>	<code>_stubs</code>	000	
<code>RBManager.start(isFirstInstall:server:storage:bluetooth:filter:)</code>	<code>_stubs</code>	000	

On peut donc en déduire que *StorageManager* est la classe qui s'occupe du stockage des données. Essayons de retrouver les informations stockés par *StorageManager*. *StorageManager* utilise *Storage SDK*, ce qui nous montre, après analyse du binaire *StorageSDK*, les données sauvegardés par la classe *StorageManager* :



```
public final class StorageManager: RBStorage {

    enum KeychainKey: String, CaseIterable {
        case dbKey
        case epochTimeStart
        case ka
        case kea
        case proximityActivated
        case isAtRisk
        case lastExposureTimeFrame
        case lastStatusRequestDate
        case lastStatusReceivedDate
        case lastRiskReceivedDate
        case isSick
        case positiveToSymptoms
        case pushToken
    }
}
```

variable	type de donnée
epochs	struct Date
timeStart	Int
ka	struct Data
kea	struct Data
localProximity	struct
proximityActivated	Bool
lastExposureTimeFrame	struct Date
LastStatusRequestDate	struct Date
LastStatusReceivedDate	struct Date
LastRiskReceivedDate	struct Date
isSick	Bool
pushToken	string

Le source code dans le *StorageManager.swift* nous permet de s'assurer que les données sont stockées dans la keychain sur le portable, en effet KeyChainSwift est utilisé pour chiffrer toutes les données mentionnées précédemment, on retrouve l'importation de la API également dans le binaire de l'application.

Comment ces données sont-elles récupérées ? Il est donc temps de se concentrer sur l'identification de l'emplacement des données de l'utilisateur sur le terminal :

L'application communique avec un serveur, on utilise donc notre machine comme un proxy pour intercepter le trafic



Via Wireshark, il a été possible de récupérer le flux de données afin d'en apprendre un peu plus sur les échanges d'informations qu'effectue l'application. On lance l'application et on observe la présence de requêtes DNS pour les noms de domaines suivants :

- app.stopcovid.gouv.fr

```

▶ Frame 15978: 90 bytes on wire (720 bits), 90 bytes captured (720 bits) on interface wlan0, id 0
▶ Ethernet II, Src: Apple_04:78:e9 (f8:27:93:04:78:e9), Dst: IntelCor_78:06:e0 (44:85:00:78:06:e0)
▶ Internet Protocol Version 4, Src: 192.168.43.24, Dst: 192.168.43.95
▶ User Datagram Protocol, Src Port: 60032, Dst Port: 53
▼ Domain Name System (query)
  Transaction ID: 0x822c
  ▶ Flags: 0x0100 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    ▶ p35-keyvalueservice.icloud.com: type AAAA, class IN

```

```

CONNECT app.stopcovid.gouv.fr:443 HTTP/1.1
Host: app.stopcovid.gouv.fr
User-Agent: StopCovid/99 CFNetwork/978.0.7 Darwin/18.7.0
Connection: keep-alive
Proxy-Connection: keep-alive

HTTP/1.0 200 Connection established

```

Après une analyse du binaire **StopCovid**, en extrayant les classes contenant les strings : "URL", "SESSION" et "SERVER", on retrouve les classes en charge des connection vers le serveur :

- 0xac0|_\$_sFoundationURL.RequestV3urlcachePolicytimeoutIntervalAcA3URLV_So0NSURLRequestCacheE0VSdtcfC|
- 0x291e8|_\$_s9RobertSDK.RBManagerC5start14isFirstInstall6server7storage9bluetooth6filterrestartProximityIfPossible0E22AtRiskDidChangeHandler07didStopm17DueToLackOfEpochsT00u7ReceivemT00u4SaveM0ySb_AA8RBServer_pAA9RBStorage_pAA11RBBluetooth_pAA11RBFiltering_pSbySbSgcyycyAA010RBReceivedM0VcSgtF|

De plus, on retrouve, dans le binaire **StopCovid**, une méthode **Server.init** qui appelle la classe **ServerSDK** :

```
; Attributes: thunk
; ServerSDK.Server.init(baseUrl: () -> Foundation.URL
;$_s9ServerSDK0A0C7baseUrl9publicKey15certificateFile
NOP
LDR          X16, =__imp__$s9ServerSDK0A0C7baseUrl
BR           X16          ; __imp__$s9ServerSDK0A0C7baseUrl
; End of function Server.init(baseUrl:publicKey:certificateFile)
```

L'examen du binaire **ServerSDK** nous permet d'obtenir les méthodes suivantes :

```
Server.publicKey.getter
Server.__allocating_init(baseUrl:publicKey:certificateFile:configUrl:completion:)
type metadata accessor for Server
Server.init(baseUrl:publicKey:certificateFile:configUrl:completion:)
Server.status(epochId:ebid:time:mac:completion:)
Server.statusV3(epochId:ebid:time:mac:completion:)
Server.report(code:helloMessages:completion:)
Server.register(token:publicKey:completion:)
Server.registerV2(captcha:captchald:publicKey:completion:)
Server.registerV3(captcha:captchald:publicKey:completion:)
Server.unregister(epochId:ebid:time:mac:completion:)
Server.unregisterV3(epochId:ebid:time:mac:completion:)
Server.deleteExposureHistory(epochId:ebid:time:mac:completion:)
Server.__allocating_init()
```


On peut donc énumérer les méthodes en charge de la communication avec le serveur :

- Server.status,
- Server.statusV3,
- Serveur.report,
- Server.register,
- Server.registerV2,
- Server.registerV3,
- Server.unregister,
- Server.unregisterV3,
- Server.processRequest,
- Server.urlSession.

Le "Certificate pinning" est utilisé afin d'éviter une attaque de type MitM, le certificat ou la clé publique d'un serveur distant est stocké dans l'application. Des API facilite l'implémentation de "Certificate Pinning" comme TustKit et Alamofire.

L'application utilise l'API SecTrust pour

4. Recommendation

L'application est en soit bien conçue, les données des utilisations sont chiffrées et sont stockées de manière sécurisé.

Par ailleurs l'envoi et les communications avec les serveurs sont bien sécurisés avec la technique de Pinning. Néanmoins des mécanisme de détection de Jailbreak sera bien venue.

Une analyse approfondie de l'application est nécessaire pour vérifier son intégrité et sa sécurité.