



Rapport de Reverse IOS

MSI P9

Réalisé par:

Layadi Mohamed Dhia Eddine
Redouane Taoussi
Anastasia Cotoroba

Encadré Par :

Mathieu Renard

18/09/2020

Introduction

La prolifération des équipements sous iOS et leur utilisation dans la vie privée en font une cible de plus en plus intéressante pour les auteurs de code malveillant. Il est ainsi intéressant d'étudier le fonctionnement des applications afin de s'assurer qu'elles protègent au mieux les données ou n'effectuent pas d'opérations malveillantes.

Ce rapport a été réalisé dans le cadre de l'examen du module IOS de l'école AFTI ,afin de se familiariser au reverse engineering d'application IOS et de permettre la mise en évidence de certaines limites dans les application proposé , et de pouvoir suggérer des pistes d'amélioration .

Objective

L'objectif de l'examen et le reverse de deux application IOS qui sont Countdown et StopCovid .

Pour la première application **CountDown** on va essayer de trouver un moyen sûr de désamorcer la bombe tout en trouvant :

- les mécanismes déclenchant la bombe (lister les fonctions/classes méthodes).
- les fonctions à patcher pour désamorcer la bombe le plus rapidement possible.
- Proposer un script Frida.

Pour la deuxième **StopCovid** on va devoir trouver un moyen pour analyser l'application et vérifier le point suivant :

- Identification et provenance de l'application qui vous a été fournie.
- Identification des données stockées sur le terminal.
- Identification de l'emplacement des données de l'utilisateur sur le terminal
- L'application communique avec un serveur
- Est-ce que les échanges sont bien protégé en confidentialité ? ..

Et Pour clôturer le rapport on va répondre a des question de cours sur le thème du reverse et du développement IOS .

Analyse de Countdown

L'analyse de l'application va se faire en BlackBox avec la méthodologie suivant :

Analyse statique :

L'analyse Statique va se baser sur le fait d'inspecter et de comprendre le fonctionnement du code sans l'exécuter et cela en réalisant :

- Inspection du binaire de l'application
- Analyse du binaire dans un désassembleur (IDA)
- Dump de la section de code crypté (AppstoreDRM)
- Utilisation des outils Mach-O : otool, class-dump

Analyse dynamique :

L'analyse dynamique va se base sur le fait d'exécuter l'application sur l'appareil pour pour observer son comportement et cela va permettre de :

- Surveiller les entrées/sorties
 - Système de fichiers, préférences de l'utilisateur, keychain
 - IPCs "Carte mère, schémas URI"
 - Services-réseau : proxy , trafic dans l'application
- Hook functions: MobileSubstrate, CydiaSubstrate
- Déboguer l'application en utilisant GDB ou JDB
- Bypass jailbreak/root détection

1. Analyse statique

On va débiter notre analyse avec l'utilisation des outils jtool/otool pour analyser le Mach-O

```
otool -h Countdown
Magic:    32-bit Mach-O
Type:     executable
CPU:      ARMv7
Cmds:     22
Size:     2784 bytes
Flags:    0x200085
```

```
jtool --sig Countdown
```

```
Blob at offset: 417072 (24576 bytes) is an embedded signature
Code Directory (2262 bytes)
  Version: 20400
  Flags: none
  Identifier: org.gotohack.CountDown
  CDHash: ded7179e13e3d10a8e44cfb3fb8958a85237d797
  # 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 J!
False Info plist
Entitlements (504 bytes) (use --ent to view)
Code Directory (3546 bytes)
  Version: 20400
  Flags: none
  Identifier: org.gotohack.CountDown
  CDHash: 63ec4c8efc6c87697d8ae9a74168e352768f6026
  # of Hashes: 102 code + 5 special
  Hashes @282 size: 32 Type: SHA-256

Blob Wrapper (4830 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
Time: 200917214558Z
```

On remarque de nombreuses informations telles que:

- Le hash du binaire
- La version
- Le flag
- L'architecture ...

Pour vérifier si le binaire est crypté on va réaliser la commande otool suivante vu que le résultat est 0 alors on réalise que le binaire n'est pas crypté:

```
otool -l Countdown | grep -A4 LC_ENCRYPTION_INFO
```

```
cmd LC_ENCRYPTION_INFO
cmdsize 20
cryptoff 16384
cryptsize 376832
cryptid 0
--
cmd LC_ENCRYPTION_INFO_64
cmdsize 24
cryptoff 16384
cryptsize 376832
cryptid 0
```

Nous utiliserons ensuite l'outil de rétro ingénierie IDA afin d'analyser statiquement le binaire Countdown:

```

; Attributes: bp-based frame

EXPORT start
start

var_20= -0x20
var_10= -0x10
var_s0= 0

STP      X22, X21, [SP, #-0x10+var_
STP      X20, X19, [SP, #0x20+var_
STP      X29, X30, [SP, #0x20+var_
ADD      X29, SP, #0x20
MOV      X19, X1
MOV      X20, X0
BL       _objc_autoreleasePoolPus
MOV      X21, X0
NOP
LDR      X0, =_OBJC_CLASS_$_AppDe
NOP
LDR      X1, =sel_class ; "class"
BL       _objc_msgSend
BL       _NSStringFromClass
MOV      X29, X29
BL       _objc_retainAutoreleased
MOV      X22, X0
MOV      X0, X20
MOV      X1, X19
MOV      X2, #0
MOV      X3, X22
BL       _UIApplicationMain
MOV      X19, X0
MOV      X0, X22
BL       _objc_release
MOV      X0, X21
BL       _objc_autoreleasePoolPop
MOV      X0, X19
LDP      X29, X30, [SP, #0x20+var_
LDP      X20, X19, [SP, #0x20+var_
LDP      X22, X21, [SP+0x20+var_2
RET
; End of function start

```

Dès le chargement du binaire on voit la fonction **start** et on remarque la présence de la classe **UIApplicationMain**, cette classe contient seulement un return, on va continuer notre inspection en cherchant tous les string du code .

```

s| __objc_met... 0000000C C viewDidLoad
s| __objc_met... 0000000F C countdownTimer
s| __objc_met... 0000000B C DisarmCode
s| __objc_met... 00000015 C resignFirstResponder
s| __objc_met... 0000000B C invalidate
s| __objc_met... 00000005 C text
s| __objc_met... 00000010 C testDisarmCode:
s| __objc_met... 00000013 C dataUsingEncoding:
s| __objc_met... 00000010 C dataWithLength:
s| __objc_met... 00000006 C bytes
s| __objc_met... 00000007 C length
s| __objc_met... 0000000D C mutableBytes
s| __objc_met... 00000016 C initWithBytes:length:
s| __objc_met... 0000000F C isEqualToData:
s| __objc_met... 00000005 C hero
s| __objc_met... 00000009 C gameover
s| __objc_met... 0000001B C storyboardWithName:bundle:
s| objc met... 00000029 C instantiateViewControllerWithIdentifier:

```

On remarque directement la présence du string intrigant qui est **DisarmCode**, **testDisarmCode** on se rend sur celui-ci et on tombe sur une fonction pour la première c'est un simple return

```

UITextField * __cdecl -[ViewController DisarmCode](ViewController *self, SEL a2)
{
    __int64 v2; // x0

    v2 = objc_loadWeakRetained(&self->_DisarmCode, a2);
    return (UITextField *)objc_autoreleaseReturnValue(v2);
}

```

Alors que pour le deuxième on tombe sur la vraie fonction de désamorçage

```

v3 = a3;
v4 = self;
v5 = objc_retain(a3, a2);
v6 = objc_msgSend(CFSTR("only for real entropy bytes!"), "dataUsingEncoding:", 4LL);
v7 = objc_retainAutoreleasedReturnValue(v6);
v23 = unk_10000A078;
v24 = unk_10000A088;
v8 = objc_msgSend(v3, "dataUsingEncoding:", 4LL);
v9 = objc_retainAutoreleasedReturnValue(v8);
objc_release(v5);
v10 = objc_msgSend(&OBJC_CLASS__NSMutableData, "dataWithLength:", 32LL);
v11 = objc_retainAutoreleasedReturnValue(v10);
v12 = (void *)objc_retainAutorelease(v9);
v13 = objc_msgSend(v12, "bytes");
v14 = objc_msgSend(v12, "length");
v15 = (void *)objc_retainAutorelease(v7);
v16 = objc_msgSend(v15, "bytes");
v17 = objc_msgSend(v15, "length");
v18 = (void *)objc_retainAutorelease(v11);
v19 = objc_msgSend(v18, "mutableBytes");
v20 = objc_msgSend(v18, "length");
if ( (unsigned int)CCKeYDerivationPBKDF(2LL, v13, v14, v16, v17, 3LL, 10000LL, v19, v20) )
{
    NSLog(CFSTR("PBKDF2 Error..."));
}
else
{
    v21 = (void *)objc_alloc(&OBJC_CLASS__NSData);
    v22 = objc_msgSend(v21, "initWithBytes:length:", &v23, 32LL);
    if ( (unsigned int)objc_msgSend(v22, "isEqualToData:", v18) )
        -[ViewController hero](v4, "hero");
    objc_release(v22);
}
-[ViewController gameover](v4, "gameover");
objc_release(v18);
objc_release(v12);
objc_release(v15);
}

```

Après lecture du code on se rend compte que la fonction va utiliser la chaîne de caractère **“only for real entropy bytes!”** pour générer un mot de passe.

Par contre on va faire une comparaison avec la fonction **isEqualToData** et le résultat de celui-ci va aboutir à l'appelle de la fonction hero ou la fonction gameover

```
void __cdecl -[ViewController hero](ViewController *self, SEL a2)
{
    ViewController *v2; // x19
    void *v3; // x0
    void *v4; // x0
    void *v5; // x20
    void *v6; // x0
    __int64 v7; // x21
















    v2 = self;
    v3 = objc_msgSend(&OBJC_CLASS__UIStoryboard, "storyboardWithName:bundle:", CFSTR("Main"), 0LL);
    v4 = (void *)objc_retainAutoreleasedReturnValue(v3);
    v5 = v4;
    v6 = objc_msgSend(v4, "instantiateViewControllerWithIdentifier:", CFSTR("Hero"));
    v7 = objc_retainAutoreleasedReturnValue(v6);
    objc_msgSend(v2, "presentViewController:animated:completion:", v7, 0LL, 0LL);
    objc_release(v7);
    objc_release(v5);
}
```

```
void __cdecl -[ViewController gameover](ViewController *self, SEL a2)
{
    ViewController *v2; // x19
    void *v3; // x0
    void *v4; // x0
    void *v5; // x20
    void *v6; // x0
    __int64 v7; // x21

    v2 = self;
    v3 = objc_msgSend(&OBJC_CLASS__UIStoryboard, "storyboardWithName:bundle:", CFSTR("Main"), 0LL);
    v4 = (void *)objc_retainAutoreleasedReturnValue(v3);
    v5 = v4;
    v6 = objc_msgSend(v4, "instantiateViewControllerWithIdentifier:", CFSTR("Kaboom"));
    v7 = objc_retainAutoreleasedReturnValue(v6);
    objc_msgSend(v2, "presentViewController:animated:completion:", v7, 0LL, 0LL);
    objc_release(v7);
    objc_release(v5);
}
```

De manière générale, ces deux fonctions ont le même fonctionnement. Elles sont appelées pour lancer une animation dès qu'un utilisateur a réussi ou a échoué à trouver le mot de passe , pour hero on va appeler **l'animation hero**, et pour l'autre **l'animation kaboom**.

Pour pouvoir continuer notre inspection on va utiliser la fonction view d'IDA

	-[ViewController disarm:]	__text
	-[ViewController testDisarmCode:]	__text
	-[ViewController hero]	__text
	-[ViewController gameover]	__text
	-[ViewController updateCounter:]	__text
	-[ViewController countdownTimer]	__text
	-[ViewController didReceiveMemoryWarning]	__text
	-[ViewController myCounterLabel]	__text
	-[ViewController setMyCounterLabel:]	__text
	-[ViewController CounterValue]	__text
	-[ViewController setCounterValue:]	__text
	-[ViewController DisarmCode]	__text
	-[ViewController setDisarmCode:]	__text
	-[ViewController winImg]	__text
	-[ViewController setWinImg:]	__text

On remarque deux contrôleurs intéressants:

- UpdateCounter
- CountdownTime

```
void __cdecl -[ViewController updateCounter:](ViewController *self, SEL a2, id a3)
{
    ViewController *v3; // x19
    int v4; // w8
    int64 v5; // x9
    int v6; // w8
    int64 v7; // x10
    int64 v8; // x8
    void *v9; // x0
    int64 v10; // x20
    int64 v11; // x1
    void *v12; // x19
    NSTimer *v13; // x0

    v3 = self;
    v4 = dword_100061FC0 - 1;
    if ( dword_100061FC0 < 1 )
    {
        objc_msgSend((void *)self->timer, "invalidate", a3);
        v13 = v3->timer;
        v3->timer = 0LL;
        objc_release(v13);
        -[ViewController gameover](v3, "gameover");
    }
    else
    {
        --dword_100061FC0;
        v5 = v4 / 3600;
        dword_1000621A0 = v4 / 3600;
        v6 = v4 % 3600;
        v7 = v6 / 60;
        dword_1000621A4 = v6 / 60;
        v8 = (unsigned int)(v6 % 60);
        dword_1000621A8 = v8;
        v9 = objc_msgSend(&OBJC_CLASS__NSString, "stringWithFormat:", CFSTR("%02d:%02d:%02d", v5, v7, v8);
        v10 = objc_retainAutoreleasedReturnValue(v9);
        v12 = (void *)objc_loadWeakRetained(&v3->_CounterValue, v11);
        objc_msgSend(v12, "setText:", v10);
        objc_release(v12);
        objc_release(v10);
    }
}
```

Cette fonction permet de décrémenter l'horloge de la bombe et de transformer les seconde on heure et on minute.


```

void __cdecl -[ViewController countdownTimer](ViewController *self, SEL a2)
{
    ViewController *v2; // x19
    void *v3; // x0
    __int64 v4; // x0
    NSTimer *v5; // x8

    v2 = self;
    v3 = objc_msgSend(
        &OBJC_CLASS__NSTimer,
        "scheduledTimerWithTimeInterval:target:selector:userInfo:repeats:",
        self,
        "updateCounter:",
        0LL,
        1LL,
        1.0);
    v4 = objc_retainAutoreleasedReturnValue(v3);
    v5 = v2->timer;
    v2->timer = (NSTimer *)v4;
    objc_release(v5);
}

```

cette fonction va set le compteur et faire un appel à UpdateCounter pour le décrémenter .

on continue notre inspection pour voir un contrôleur qui charge le fichier flag.enc ce contrôleur utilise la bibliothèque NSURL cette bibliothèque va permettre de crypter URL on utilisant le path

```

void __cdecl -[HeroViewController viewDidLoad](HeroViewController *self, SEL a2)
{
    HeroViewController *v2; // x19
    void *v3; // x0
    void *v4; // x0
    void *v5; // x21
    void *v6; // x0
    __int64 v7; // x20
    void *v8; // x0
    __int64 v9; // x0
    __int64 v10; // x21
    void *v11; // x0
    __int64 v12; // x0
    __int64 v13; // x22
    void *v14; // x0
    __int64 v15; // x23
    UIImageView *v16; // x0
    void *v17; // x19
    HeroViewController *v18; // [xsp+0h] [xbp-40h]
    __objc2_class *v19; // [xsp+8h] [xbp-38h]

    v2 = self;
    v18 = self;
    v19 = &OBJC_CLASS__HeroViewController;
    objc_msgSendSuper2(&v18, "viewDidLoad", self, &OBJC_CLASS__HeroViewController);
    v3 = objc_msgSend(&OBJC_CLASS__NSBundle, "mainBundle");
    v4 = (void *)objc_retainAutoreleasedReturnValue(v3);
    v5 = v4;
    v6 = objc_msgSend(v4, "pathForResource ofType:", CFSTR("flag"), CFSTR("enc"));
    v7 = objc_retainAutoreleasedReturnValue(v6);
    objc_release(v5);
    v8 = objc_msgSend(&OBJC_CLASS__NSURL, "encryptedFileURLWithPath:", v7);
    v9 = objc_retainAutoreleasedReturnValue(v8);
    v10 = v9;
    v11 = objc_msgSend(&OBJC_CLASS__NSData, "dataWithContentsOfURL:", v9);
    v12 = objc_retainAutoreleasedReturnValue(v11);
    v13 = v12;
    v14 = objc_msgSend(&OBJC_CLASS__UIImage, "imageWithData:", v12);
    v15 = objc_retainAutoreleasedReturnValue(v14);
    v16 = -[HeroViewController winImg](v2, "winImg");
    v17 = (void *)objc_retainAutoreleasedReturnValue(v16);
    objc_msgSend(v17, "setImage:", v15);
}

```

Après cela on va réaliser un dump de l'application

```
...

@interface ViewController : UIViewController
{
    NSTimer *timer;
    UILabel *_myCounterLabel;
    UILabel *_CounterValue;
    UITextField *_DisarmCode;
    UIImageView *_winImg;
}

@property(n nonatomic) __weak UIImageView *winImg; // @synthesize winImg=_winImg;
@property(n nonatomic) __weak UITextField *DisarmCode; // @synthesize DisarmCode=_DisarmCode;
@property(n nonatomic) __weak UILabel *CounterValue; // @synthesize CounterValue=_CounterValue;
@property(retain, nonatomic) UILabel *myCounterLabel; // @synthesize
myCounterLabel=_myCounterLabel;
- (void).cxx_destruct;
- (void)didReceiveMemoryWarning;
- (void)countdownTimer;
- (void)updateCounter:(id)arg1;
- (void)gameover;
- (void)hero;
- (void)testDisarmCode:(id)arg1;
- (void)disarm:(id)arg1;
- (_Bool)textFieldShouldReturn:(id)arg1;
- (void)viewDidAppear:(_Bool)arg1;
- (void)viewDidLoad;
- (void)startLoading;

@interface HeroViewController : UIViewController
{
    UIImageView *_winImg;
}

@property(n nonatomic) __weak UIImageView *winImg; // @synthesize winImg=_winImg;
- (void).cxx_destruct;
- (void)viewDidLoad;

@end

@interface NSURL (EncryptedFileURLProtocol)
+ (id)encryptedFileURLWithPath:(id)arg1;
@end

...
```

2. Analyse Dynamique

ios-deploy

Pour poursuivre l'analyse de ce binaire, cette fois dynamique, nous allons dans un premier temps déployer l'application à l'aide de l'outil *ios-deploy*.

```
~ » ios-deploy --bundle Countdown.app -W -d
```

Objection

Nous pouvons optionnellement regarder le contenu de l'environnement de l'application grâce à la commande:

```
~ » objection -g Countdown explore
```

```
/Volumes/class-dump-3.5 » objection -g Countdown explore
```

```
Using USB device `iPhone`
```

```
Agent injected and responds ok!
```

```

  _ _ _ _ _
 | | | | | | | | | |
 | . | . | | - | _ | _ | | . | |
 | _ | _ | | _ | _ | | | _ | _ |
      | _ | (object)inject(ion) v1.9.6

```

```
Runtime Mobile Exploration
```

```
by: @leonjza from @sensepost
```

```
[tab] for command suggestions
```

```
org.gotohack.CountDown on (iPhone: 12.4) [usb] # █
```

puis : env

```
org.gotohack.CountDown on (iPhone: 12.4) [usb] # env
```

Name	Path
BundlePath	/var/containers/Bundle/Application/2B8BDAAC-EEE9-4251-B417-A8A231E02858/CountDown.app
CachesDirectory	/var/mobile/Containers/Data/Application/3E1F7257-E2A2-456C-BFA5-E9BA40B8AEA9/Library/Caches
DocumentDirectory	/var/mobile/Containers/Data/Application/3E1F7257-E2A2-456C-BFA5-E9BA40B8AEA9/Documents
LibraryDirectory	/var/mobile/Containers/Data/Application/3E1F7257-E2A2-456C-BFA5-E9BA40B8AEA9/Library

```
org.gotohack.CountDown on (iPhone: 12.4) [usb] # █
```

Frida

Pour désamorcer la bombe nous avons décidé de patcher l'application à l'aide de [Frida](#). Frida est un outil qui permet l'injection des snippets JavaScript dans des applications qui tournent sur Windows, macOS, GNU/Linux, iOS, Android, et QNX.

Tout d'abord nous allons récupérer le PID de l'application Countdown, qui nous sera utile par la suite avec la commande:

```
~ » frida-ps -Uia
```

PID	Name	Identifïer
4772	App Store	com.apple.AppStore
4646	Appareil photo	com.apple.camera
4785	CountDown	org.gotohack.CountDown
4559	Mail	com.apple.mobilemail
4700	Musique	com.apple.Music
4777	Safari	com.apple.mobilesafari

Ensuite nous allons chercher l'adresse du registre que nous voulons patcher. Dans un premier temps nous allons essayer d'augmenter la valeur du chronomètre.

```

LAB_100005dd8                                     XREF[1]:
100005dd8 ff 03 01 d1      sub      sp,sp,#0x40
100005ddc f4 4f 02 a9      stp      x20,x19,[sp, #0x20]
100005de0 fd 7b 03 a9      stp      x29,x30,[sp, #0x30]
100005de4 fd c3 00 91      add      x29,sp,#0x30
100005de8 f3 03 00 aa      mov      x19,x0
100005dec e9 02 00 90      adrp     x9,0x100061000
100005df0 28 c1 4f b9      ldr      w8,[x9, #0xfc0]=>DAT_100061fc0
100005df4 08 05 00 71      subs     w8,w8,#0x1

```

Avec IDA nous pouvons chercher le registre que nous voulons modifier. Nous pouvons trouver à l'adresse 0x5dfc le registre x8 qui contient la valeur du chronomètre et qui est décrémenté juste après (subs).

Nous allons donc créer un script frida qui va aller modifier la valeur de x8.

```
var targetModule = 'CountDown';
var addr = ptr(0x5dfc);
var moduleBase = Module.getBaseAddress(targetModule);
var targetAddress = moduleBase.add(addr);
Interceptor.attach(targetAddress, {
  onEnter: function(args) {
    this.context.x8 = 100;
    console.log('At the address ' + addr + ' the value is currently ' + this.context.x8);
  },
});
```

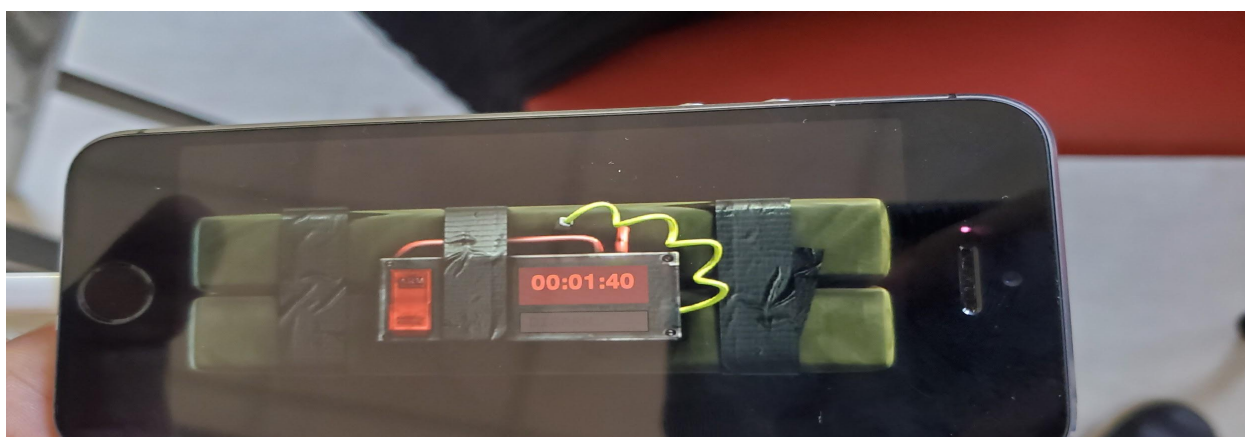
Le script sera exécuté avec la commande suivante:

```
~ » frida -U -l frida1.js 4792
```

```
~/Desktop/exam2020/CountDown/Payload » frida -U -l frida1.js 4792

----
/ _ |   Frida 12.11.16 - 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/

[iPhone::PID::4792]-> At the address 0x5dfc the value is currently 0x64
```



vu qu'on a pas pu avoir le mot de passe bien que on est réussi a augmenter le timeur on change de stratégie on alon directement chercher d'adresse de l'instruction responsable du lancement de la fonction hero et on la modifie pour bypass le IF .

```

100005c1c 61 fa 2d 58    ldr      x1=>s_isEqualToData: 100008c0e,PTR_s_isEqualTo...
100005c20 e2 03 16 aa    mov     x2,x22
100005c24 d0 0a 00 94    bl      stubs::_objc_msgSend
100005c28 a0 00 00 34    cbz     w0,_AB_100005c3c
100005c2c 1f 20 03 d5    nop
100005c30 01 fa 2d 58    ldr      x1=>s_hero_100008c1d,PTR_s_hero_100061b70
100005c34 e0 03 13 aa    mov     x0,x19
100005c38 cb 0a 00 94    bl      __stubs::_objc_msgSend

```

on va écrire un script frida qui va modifier le return du IF , on se rend compte qu'après l'instruction CBZ le résultat est écrit dans le registre x0 . quand la valeur de x0 est 0x00 on passe dans le else donc on modifie la valeur, on la set a 0x01

```

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 Disarm");
    }
    console.log('At the address ' + addr + ' the value is currently ' + this.context.x0);
  },
});

```

Et Voilà :-)



Recommandation

- prévoir un mécanisme de détection de jailbreak et annuler le fonctionnement de l'application si celui-ci est activé .
- Le mot de passe peut être trouvé vu qu'il est issu de la modification d'une phrase claire.
- chiffrer l'application pour empêcher un utilisateur malveillant d'avoir accès au code et ajouter une couche d'obfuscation .

Analyse de Stop_Covid

L'application nous a été fournie sous deux formes:

- En format "IPA", qui nous a servi pour effectuer une analyse dynamique de l'application
- Installé dans un iPhone 5s, qui nous a permis d'analyser dynamiquement l'application

Nous avons donc commencé par une analyse statique des différents binaires composant le paquet StopCovid.ipa. Le premier binaire analysé "StopCovid" nous a permis d'identifier la légitimité de l'application, en utilisant l'outil JTOOL:

```
~ jtool --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:      KCWS38TNWR (@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: Mathieu RENARD (KCWS38TNWR)' 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
```


En analysant le résultat de l'outil JTOOL, nous avons constaté que le certificat a été distribué par "Matthieu Renard", et que le nom du domaine est "org.gotohack.stopcovid.ios", l'application n'est donc pas légitime.

Nous avons ensuite analysé le contenu des fichiers info.plist et nous avons découvert que l'application demande à l'utilisateur d'octroyer l'accès :

- à la caméra afin de scanner des QRcodes:

```
<key>NSCameraUsageDescription</key>  
  <string>StopCovid needs to access your camera in order to flash a QR
```

- au bluetooth, pour les fonctionnalités de proximité:

```
<key>NSBluetoothPeripheralUsageDescription</key>  
  <string>StopCovid needs Bluetooth to work.</string>
```

- à l'exécution en arrière plan:

```
<key>UIBackgroundModes</key>  
  <array>  
    <string>bluetooth-central</string>  
    <string>bluetooth-peripheral</string>  
    <string>fetch</string>  
    <string>remote-notification</string>
```

Nous avons ensuite décidé d'analyser le binaire StorageSDK qui est chargé de stocker et gérer les données de l'application ainsi que celles de l'utilisateur. Nous avons aussi utilisé l'outil de rétro ingénierie IDA afin d'analyser ce binaire, et nous avons donc réussi à lister quelques types de données stockées par l'application:

- Epochs
- timeStart
- localProximity
- lastExposureTimeFrame
- lastStatusRequestDate
- lastStatusReceiveDate
- isSick

- pushToken
- lastRiskReceivedData
- atRisk

```

[f] StorageManager.save(epochs:)
[f] StorageManager.save(timeStart:)
[f] StorageManager.save(ka:)
[f] StorageManager.save(kea:)
[f] StorageManager.save(localProximity:)
[f] StorageManager.save(proximityActivated:)
[f] StorageManager.save(lastExposureTimeFrame:)
[f] StorageManager.saveLastStatusRequestDate(_)
[f] StorageManager.saveLastStatusReceivedDate(_)
[f] StorageManager.saveLastRiskReceivedDate(_)
[f] StorageManager.save(isSick:)
[f] StorageManager.save(pushToken:)

```

En analysant davantage le binaire, nous avons constaté que le StorageManager utilise en même temps le mécanisme de Keychain pour stocker les données de l'utilisateur et l'API Realm afin de stocker des données application et utilisateur.

```

; Attributes: bp-based frame

; StorageSDK.StorageManager.save(isSick: Swift.Bool) -> ()
EXPORT __$s10StorageSDK0A7ManagerC4save6isSickySb_tf
__$s10StorageSDK0A7ManagerC4save6isSickySb_tf

var_28= -0x28
var_10= -0x10
var_s0= 0





















SUB     SP, SP, #0x40
STP     X20, X19, [SP, #0x30+var_10]
STP     X29, X30, [SP, #0x30+var_s0]
ADD     X29, SP, #0x30
LDR     X20, [X20, #0x10]
LDR     X8, [X20]
NOP

LDR     X9, =__$s13KeychainSwiftAACN ; type metadata for KeychainSwift
__text:000000000000AB00      CMP     X8, X9
__text:000000000000AB04      B.EQ   loc_AB2C

X8, [X8, #0x118]
W0, W0, #1
W3, #3
X1, #0x6B6369537369
X2, #0xE600000000000000
X8
loc_AB48

__text:000000000000AB2C ; -----
__text:000000000000AB2C
__text:000000000000AB2C loc_AB2C ; CODE XREF:
__text:000000000000AB2C      AND     W0, W0, #1
__text:000000000000AB30      MOV     W3, #3
__text:000000000000AB34      MOV     X1, #0x6B6369537369
__text:000000000000AB40      MOV     X2, #0xE600000000000000
__text:000000000000AB44      BL      __$s13KeychainSwiftA/

```

	-[RealmLocalProximity id]	__text
	-[RealmLocalProximity setId:]	__text
	-[RealmLocalProximity ecc]	__text
	-[RealmLocalProximity setEcc:]	__text
	-[RealmLocalProximity ebid]	__text
	-[RealmLocalProximity setEbid:]	__text
	-[RealmLocalProximity mac]	__text
	sub_10624	__text
	-[RealmLocalProximity setMac:]	__text
	sub_10694	__text
	-[RealmLocalProximity timeFromHelloMess...	__text
	-[RealmLocalProximity setTimeFromHelloM...	__text
	-[RealmLocalProximity timeCollectedOnDe...	__text
	-[RealmLocalProximity setTimeCollectedOn...	__text
	-[RealmLocalProximity rssiRaw]	__text
	-[RealmLocalProximity setRssiRaw:]	__text
	-[RealmLocalProximity rssiCalibrated]	__text
	-[RealmLocalProximity setRssiCalibrated:]	__text
	-[RealmLocalProximity tx]	__text
	-[RealmLocalProximity setTx:]	__text

Nous avons ensuite décidé, afin de déterminer les différentes API externes avec lesquelles l'application communique, d'intercepter et analyser le réseau. Pour cela nous avons utilisé l'outil ARPSPOOF, afin de sniffer le trafic réseau de l'application:

```
[root@parrot]~/home/doudi
#arp spoof -t 192.168.43.202 192.168.43.180
d8:f2:ca:3f:80:13 0:0:0:0:0:0 0806 42: arp reply 192.168.43.180 is-at d8:f2:ca:3f:80:13
d8:f2:ca:3f:80:13 0:0:0:0:0:0 0806 42: arp reply 192.168.43.180 is-at d8:f2:ca:3f:80:13
d8:f2:ca:3f:80:13 0:0:0:0:0:0 0806 42: arp reply 192.168.43.180 is-at d8:f2:ca:3f:80:13
d8:f2:ca:3f:80:13 0:0:0:0:0:0 0806 42: arp reply 192.168.43.180 is-at d8:f2:ca:3f:80:13
d8:f2:ca:3f:80:13 0:0:0:0:0:0 0806 42: arp reply 192.168.43.180 is-at d8:f2:ca:3f:80:13
```

Nous avons ensuite utilisé l'outil Wireshark pour analyser ce trafic, mais dû à une contrainte de temps, nous n'avons pas pu identifier les différents API avec lesquelles l'application aurait pu communiquer. Nous avons malgré cela décidé de garder la trace Wireshark suivante:

207	78.505865386	192.168.43.180	74.125.133.188	TCP	66	[TCP Dup ACK 105#1] 56434 → 5228 [ACK] Seq=1 Ack=1 Win=63 Len=0
208	78.540185776	74.125.133.188	192.168.43.180	TCP	66	[TCP Dup ACK 106#1] [TCP ACKed unseen segment] 5228 → 56434 [..
214	81.884516675	192.168.43.180	216.58.209.238	TLSv1.2	105	Application Data
215	81.885403187	192.168.43.180	216.58.209.238	TLSv1.2	90	Application Data
216	81.922704229	216.58.209.238	192.168.43.180	TCP	66	443 → 56040 [ACK] Seq=40 Ack=104 Win=1050 Len=0 TSval=4146920...
217	81.923242778	216.58.209.238	192.168.43.180	TCP	66	443 → 56040 [FIN, ACK] Seq=40 Ack=104 Win=1050 Len=0 TSval=41...
218	81.923291450	192.168.43.180	216.58.209.238	TCP	66	56040 → 443 [ACK] Seq=104 Ack=41 Win=63 Len=0 TSval=390702531...
223	84.174655038	108.177.15.189	192.168.43.180	TLSv1.2	119	Application Data
224	84.174686606	192.168.43.180	108.177.15.189	TCP	66	57494 → 443 [ACK] Seq=1 Ack=213 Win=63 Len=0 TSval=3195022610...
225	84.582769893	162.159.135.234	192.168.43.180	TLSv1.2	196	Application Data
226	84.582824216	192.168.43.180	162.159.135.234	TCP	54	35166 → 443 [ACK] Seq=103 Ack=544 Win=63 Len=0
228	85.133511470	192.168.43.180	216.58.213.67	TLSv1.2	166	Application Data
229	85.170690189	216.58.213.67	192.168.43.180	TCP	66	443 → 49488 [ACK] Seq=40 Ack=140 Win=266 Len=0 TSval=38523459...
230	85.170690361	216.58.213.67	192.168.43.180	TLSv1.2	134	Application Data
231	85.170690469	216.58.213.67	192.168.43.180	TLSv1.2	140	Application Data
232	85.170731756	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=140 Ack=108 Win=63 Len=0 TSval=31448488...
233	85.170777693	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=140 Ack=182 Win=63 Len=0 TSval=31448488...
234	85.176683460	216.58.213.67	192.168.43.180	TLSv1.2	97	Application Data
235	85.176683611	216.58.213.67	192.168.43.180	TLSv1.2	105	Application Data
236	85.176716026	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=140 Ack=213 Win=63 Len=0 TSval=31448488...
237	85.176772743	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=140 Ack=252 Win=63 Len=0 TSval=31448488...
238	85.177060273	192.168.43.180	216.58.213.67	TLSv1.2	105	Application Data
240	85.217192116	216.58.213.67	192.168.43.180	TCP	66	443 → 49488 [ACK] Seq=252 Ack=179 Win=266 Len=0 TSval=3852346...
245	88.521672207	192.168.43.180	216.58.213.67	TLSv1.2	166	Application Data
246	88.555277917	216.58.213.67	192.168.43.180	TCP	66	443 → 49488 [ACK] Seq=252 Ack=279 Win=266 Len=0 TSval=3852349...
247	88.561343125	216.58.213.67	192.168.43.180	TLSv1.2	134	Application Data
248	88.561343300	216.58.213.67	192.168.43.180	TLSv1.2	140	Application Data
249	88.561343403	216.58.213.67	192.168.43.180	TLSv1.2	97	Application Data
250	88.561343509	216.58.213.67	192.168.43.180	TLSv1.2	105	Application Data
251	88.561369857	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=279 Ack=320 Win=63 Len=0 TSval=31448521...
252	88.561412436	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=279 Ack=394 Win=63 Len=0 TSval=31448521...
253	88.561429044	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=279 Ack=425 Win=63 Len=0 TSval=31448521...
254	88.561444680	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=279 Ack=464 Win=63 Len=0 TSval=31448521...
255	88.562064087	192.168.43.180	216.58.213.67	TLSv1.2	105	Application Data
256	88.597304699	216.58.213.67	192.168.43.180	TCP	66	443 → 49488 [ACK] Seq=464 Ack=318 Win=266 Len=0 TSval=3852349...
259	89.907488024	162.159.135.234	192.168.43.180	TLSv1.2	168	Application Data
260	89.907519533	192.168.43.180	162.159.135.234	TCP	54	35166 → 443 [ACK] Seq=103 Ack=658 Win=63 Len=0
264	91.363004939	192.168.43.180	216.58.213.67	TLSv1.2	157	Application Data
265	91.395243749	216.58.213.67	192.168.43.180	TCP	66	443 → 49488 [ACK] Seq=464 Ack=409 Win=266 Len=0 TSval=3852352...
266	91.401163743	216.58.213.67	192.168.43.180	TLSv1.2	157	Application Data
267	91.401163922	216.58.213.67	192.168.43.180	TLSv1.2	140	Application Data
268	91.401164022	216.58.213.67	192.168.43.180	TLSv1.2	105	Application Data
269	91.401180922	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=409 Ack=555 Win=63 Len=0 TSval=31448550...
270	91.401233909	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=409 Ack=629 Win=63 Len=0 TSval=31448550...
271	91.401245856	192.168.43.180	216.58.213.67	TCP	66	49488 → 443 [ACK] Seq=409 Ack=668 Win=63 Len=0 TSval=31448550...
272	91.401860406	192.168.43.180	216.58.213.67	TLSv1.2	105	Application Data

- ▼ Transmission Control Protocol, Src Port: 443, Dst Port: 35166, Seq: 544, Ack: 103,
 - Source Port: 443
 - Destination Port: 35166
 - [Stream index: 4]
 - [TCP Segment Len: 114]
 - Sequence number: 544 (relative sequence number)
 - Sequence number (raw): 3374406314
 - [Next sequence number: 658 (relative sequence number)]
 - Acknowledgment number: 103 (relative ack number)
 - Acknowledgment number (raw): 753489958
 - 0101 = Header Length: 20 bytes (5)
 - Flags: 0x018 (PSH, ACK)
 - Window size value: 72
 - [Calculated window size: 72]
 - [Window size scaling factor: -1 (unknown)]
 - Checksum: 0x2475 [unverified]
 - [Checksum Status: Unverified]
 - Urgent pointer: 0
 - [SEQ/ACK analysis]
 - [Timestamps]
 - TCP payload (114 bytes)
- ▼ Transport Layer Security
 - TLSv1.2 Record Layer: Application Data Protocol: http-over-tls

Nous nous sommes ensuite intéressés à la communication entre l'application et le serveur. Notamment, nous avons constaté que l'application effectue bien un "certificate pinning" afin d'éviter des attaques de type Man in The Middle.

```
f variable initialization expression of ParametersManager.CertificateFile
f Server.__allocating_init(baseUrl:publicKey:certificateFile:configUrl:configCertificateFile:deviceTimeNotAlignedToServerTimeDetected:)
f Server.init(baseUrl:publicKey:certificateFile:configUrl:configCertificateFile:deviceTimeNotAlignedToServerTimeDetected:)
f static CertificatePinning.validateChallenge(_:certificateFile:completion:)
f CertificatePinning.deinit
f CertificatePinning.__deallocating_deinit
f type metadata accessor for CertificatePinning
f _SecCertificateCopyData
f _SecTrustGetCertificateAtIndex
```

Comme on peut le voir dans la capture d'écran précédente, l'application effectue une validation d'un challenge en utilisant le certificat afin de protéger les communications.

Question de cours

1. Citer 3 mécanismes de sécurités présents sur le système iOS ?

En mécanisme de sécurités il existe sur IOS le :

Keychain (Petite base de données pour contenir des secrets)

Fairplay encryption (Gestion des droits)

Cryptd

2. Qu'est-ce que le jailbreak ?

Un jailbreak nous permet d'avoir un accès au noyau en super utilisateur.

On peut ainsi ajouter des solutions non-officielles comme un débogueur type gdb, ou bien cydia.

Grâce au jailbreak nous pouvons déchiffrer des applications et faire de la rétro-ingénierie sur des logiciels (dump etc...).

3. Quelle est le format des fichiers binaires iOS ?

Mach-O est le format utiliser sur des binaires IOS

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

On peut considérer FAT header comme le package de plusieurs architectures.

exemple : armv7 (32 bits) avec arm64

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

l'A8 représente un changement pour apple, en effet sur cette architecture il décide passer en arm64 bits .

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

Exemple de commande pour obtenir le MAGIC header

```
[mear@KITTY Countdown.app]$# otool -h fichier
```

Processing templerun2:

Magic: 32-bit Mach-O

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

Les fichier plist contient des propriété pour les binaires (demande d'autorisation application, chiffrements ...)

Ils sont souvent écrit en XML / JSON.

Celui-ci contient de propriété pour un binaire (exemple des différents droit requis pour une applications.

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

Certains arguments change en arm64, les registres r, s'écrivent x, swi se transforme en arm64 en swc

9. Qu'est-ce que le Keychain ?

Keychain est une petite base de donnée sécurisée créé par apple permettant de sauvegarder des informations sensible, telle que des mot de passe, allant jusqu'à la carte bleu. Avec des certificats délivrée par apple ainsi que des clé de chiffrement et une liste de service confiance, c'est élément renforce la sécurité de cette base de donnée.

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

En objective-C, les méthodes ne sont pas appelées directement, mais on utilise la méthode objc_msgSend qui va gérer l'appel à la méthode en question.