行動應用App安全開發說明會
iOS設計實務

指導單位：經濟部工業局

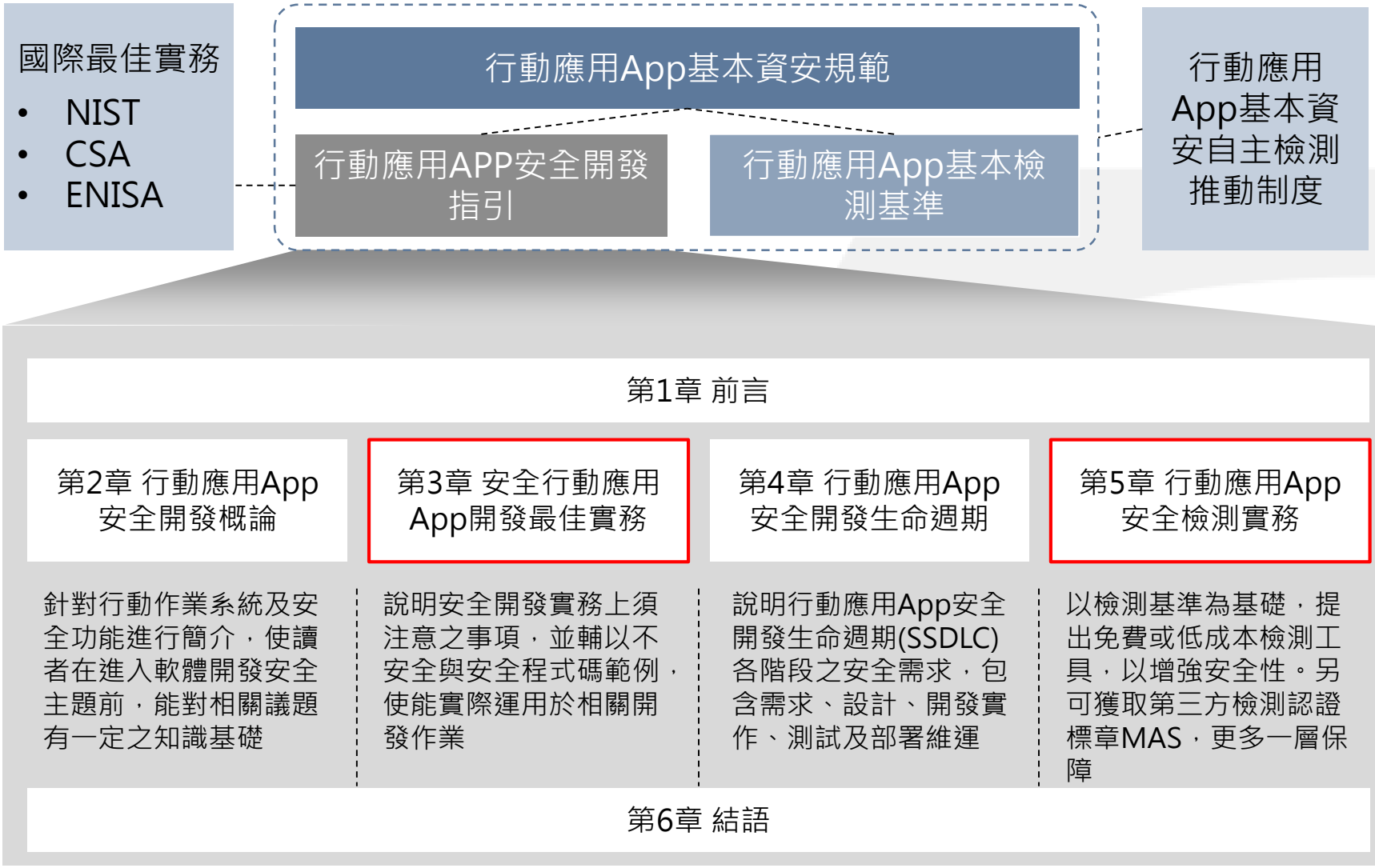
執行單位：財團法人資訊工業策進會、中華民國資訊軟體協會

協辦單位：台北市電腦公會、中華民國資訊安全學會

課程內容簡介

- iOS安全行動應用設計最佳實務
  - 本單元將介紹在iOS上開發行動應用App的安全開發注意事項，除使學員了解相關注意事項細節、解決方法，並輔以程式碼範例，使學員未來能實際運用於行動應用App開發作業中。
- 行動應用APP安全檢測流程與工具
  - 參考經濟部工業局所發展的「行動應用App基本資安規範」、「行動應用基本資安檢測基準」與「行動應用App基本資安自主檢測推動制度」等相關規範及OWASP/Cigital Touchpoint方法論。
  - 介紹行動應用App源碼檢測工具，協助開發人員得以在建構行動應用App時，從源頭貫徹資訊安全觀念作法，並能自行以低成本進行安全檢測。

行動應用APP安全開發指引架構簡介



課程大綱

第1單元	iOS安全行動應用設計最佳實務	2.5小時
第2單元	行動應用App安全檢測流程與工具	0.5小時

專案原始檔： https://github.com/rextsai

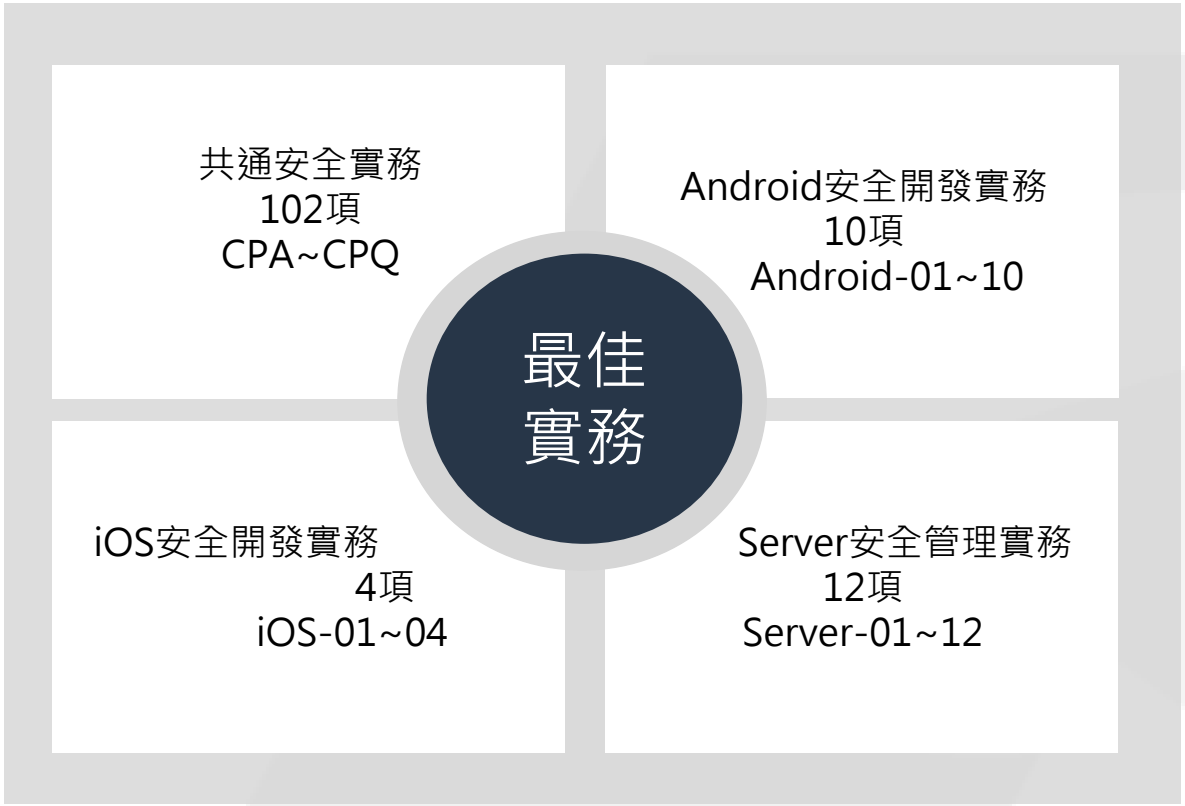
https://github.com/rextsai/MobileAppSecDev
https://github.com/rextsai/MobileAppSecDev-iOS-Project

行動應用App安全開發最佳實務

依據我國經濟部工業局「行動應用App基本資安規範」及「行動應用App基本資安檢測基準」及中國大陸、歐盟、日本、美國及CSA行動應用App安全開發實務要點。

4
個實務類別

128
項實務準則



實務準則一覽表(以iOS之App開發為主)

102
項共通實務

4
項iOS實務

12
項Server實務

共通安全開發實務準則(類別)	
A	行動應用程式發布
B	行動應用程式更新
C	行動應用程式安全性問題回報
D	敏感性資料蒐集
E	敏感性資料利用
F	敏感性資料儲存
G	敏感性資料傳輸
H	敏感性資料分享
I	敏感性資料刪除
J	付費資源使用
K	付費資源控管
L	使用者身分認證與授權
M	連線管理機制
N	防範惡意程式碼與避免資訊安全漏洞
O	行動應用程式完整性
P	函式庫引用安全
Q	使用者輸入驗證

iOS安全開發實務(類別)	
1	F.敏感性資料儲存
2	F.敏感性資料儲存
3	N.防範惡意程式碼與避免資訊安全漏洞
4	G.敏感性資料傳輸
伺服器安全實務	
1	作業系統強化與記錄留存
2	作業系統強化與記錄留存
3	網頁服務安全
4	網頁服務安全
5	網頁服務安全
6	網頁服務安全
7	網頁服務安全
8	網頁服務安全
9	網路安全防護
10	網路安全防護
11	網路安全防護
12	網路安全防護

由於時間因素，以下將挑選幾項最佳實務進行說明。

課程大綱

第1單元	iOS安全行動應用設計最佳實務	2.5小時
	共通實務(Common Practices)	
	iOS實務	
	Server實務	
第2單元	行動應用App安全檢測流程與工具	0.5小時

敏感性資料蒐集(D) -CPD-01

CPD-01 需求階段需要識別計畫開發行動應用App蒐集、處理及利用敏應資料類別(例如密碼，個人資料、地理位置、財務資訊及錯誤日誌等)及行動應用App平台與發行國的隱私法令要求，定義安全性需求。

簡介

- 通常免費或無償提供行動應用App，多數負有蒐集、利用及分享行動智慧裝置擁有者個人資料有關識別與行為資料任務 。
- 個人資料的蒐集、處理及利用應遵循發行國之個資保護相關法令規定。

> 明確告知蒐集當事人的個人資料由那一個公務或非公務機關以何種目的蒐集個資當事人的何種類別資料，且會利用之期間、地區、對象及方式，及當事人對已被蒐集的個人資料有何權利，讓使用者在下載之前或是蒐集使用者本人個人資料前有判斷的依據，讓使用者可以選擇同意或是拒絕，如果選擇拒絕是否會影響公務或非公務機關對當事人服務的品質。

- 行動應用App專案經理與系統分析人員應充分識別需求單位處理個人資料在內的敏感資料類別，並定義出安全保護需求，由系統分析 / 設計人員進一步設計安控及隱私保護措施。

開發生命週期	需求階段		
不安全程式碼範例	N/A	安全程式碼範例	N/A
行動應用App基本資安規範	4.1.2.1	行動應用App基本資安檢測基準	N/A

敏感性資料利用(E)-CPE-03

CPE-03 不得實作未經使用者同意，使行動應用App可擅自修改使用者資料的行為，包括在使用者無確認情況下刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為。

簡介

- 使用者同意行動應用App開發人員蒐集敏感性資料不代表使用者已經授權可以任意處理這些敏感性資料，破壞資料完整性與正確性，如確有需要修改使用者資料，包括刪除或修改使用者連絡人資料、通話記錄、簡訊資料和多媒體簡訊資料的行為，請參考CPD-05及CPD-06實務取得使用者同意的意向再啟動。

CPD-05:蒐集敏感性資料應實作使用者同意或拒絕選項，提示使用者選擇時機可於(1)安裝時(2)當敏感資料被儲存或傳送前(3)預設設定為關閉同意，需使用者自行開啟。

CPD-06:應依CPD-03實作行動應用App存取敏感性資料權限，不要授與過度蒐集不必要敏感性資料權限或於未告知使用者取得同意前於行動應用App背景運作蒐集敏感性資料活動。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.2.2	行動應用App基本資安檢測基準	N/A

- *檢查授權*

```
// Check the authorization status of our application for Address Book
-（void）checkAddressBookAccess
{
    switch （ABAddressBookGetAuthorizationStatus（））
    {
        // Update our UI if the user has granted access to their Contacts
        case  kABAuthorizationStatusAuthorized:
            [self accessGrantedForAddressBook];
            break;
            // Prompt the user for access to Contacts if there is no definitive answer
        case  kABAuthorizationStatusNotDetermined :
            [self requestAddressBookAccess];
            break;
            // Display a message if the user has denied or restricted access to Contacts
        case  kABAuthorizationStatusDenied:
        case  kABAuthorizationStatusRestricted:
        {
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Privacy Warning"
                                                message:@"Permission was not granted for Contacts."
                                                delegate:nil
                                                cancelButtonTitle:@"OK"
                                                otherButtonTitles:nil];
            [alert show];
        }
            break;
        default:
            break;
    }
```

敏感性資料儲存(F)-CPF-01

CPF-01 行動應用App如需要儲存敏感性資料需依CPD-02規劃於可信任之應用程式商店或行動應用程式內聲明。

簡介

- 行動應用App如需要儲存”敏感性資料”參考CPA-01將行動應用App需要儲存敏感性資料於Google Play及App Store及行動應用App的隱私權政策中聲明。

CPA-01
於行動應用程式商店提供及應用程式內實作提供隱私權政策說明連結，說明欲存取之敏感性資料、行動裝置資源及宣告權限用途。

敏感性資料(Sensitive Data)
指依使用者行為或行動App之運作，建立或儲存於行動裝置及其附屬 儲存媒介之資訊，而該資訊之洩漏有對使用者造成損害之虞，包括但不限於個人資料、通行碼、即時通訊訊息、筆記、備忘錄、通訊錄、地理位置、行事曆、通話紀錄及簡訊。

開發生命週期	設計階段		
不安全程式碼範例	N/A	安全程式碼範例	N/A
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	4.1.2.3.1

敏感性資料儲存(F)-CPF-08

CPF-08 需注意記憶體暫存的敏感性資料，如固定金鑰與密碼的安全性，當不需要時或於一定合理期間應強制清除或使用於一定期間就失效的可變量金鑰替代。

簡介

- 在行動應用App使用時，使用者或應用程式會將特定資料儲存在記憶體中，以方便使用者離開或逾時之後，下次能夠直接使用。
- 在Android上因為應用程式使用後停留在記憶體中，會被駭客利用除錯器竊取。
- 另一方面，針對敏感的密鑰、密碼，建議不要用字串儲存，改用Byte Array的方式儲存，以免容易被發現。而使用過之後，也建議應該儘快清除，或確定會被系統記憶體回收機制回收。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	N/A

安全程式碼範例：

iOS：Objective-C

例如：extern unsigned char * CC_SHA256（const void *data, CC_LONG len, unsigned char *md）；中的參數 data 來源，除了標準的資料輸入以外，還可加入長度、固定的 Salt 與私密的計算。例如：由 byte array 而來的 hash，或者計算後的 XOR。

```
 unsigned char obfuscatedSecretKey[] = { 0x33, 0xAD, 0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF,  0xDE, 0xAD,
0xBE, 0xEF, 0xDE, 0xAD, 0xBE, 0xEF };
int hash = [Server ComputeHash:obfuscatedSecretKey];
//One-at-a-Time Hash
+（int） ComputeHash: （unsigned char*） data;
{
  const int p = 16777619;
  int hash = （int）2166136261;
  int len = （int）strlen（（char*）data）;
  for （int i = 0; i < len; i++）
  {
    hash = （hash ^ data[i]） * p;
    hash += hash << 13;
    hash ^= hash >> 7;
    hash += hash << 3;
    hash ^= hash >> 17;
    hash += hash << 5;
    return hash;
  }
  // XOR the class name against the obfuscated key, to form the real key.
  unsigned char actualSecretKey[sizeof（obfuscatedSecretKey）];
  for （int i=0; i<sizeof（obfuscatedSecretKey）; i++） {
      actualSecretKey[i] = obfuscatedSecretKey[i] ^ obfuscatedSecretKey[i];
  }
```

敏感性資料儲存(F)-CPF-17

CPF-17 行動應用App有提供標準的設定參數檔函數，但由於駭客取得安裝檔之後，非常容易就可以修改這些標準的參數檔，於是為了加強安全層級，應用程式的設定參數，建議放在安全的地方，例如編譯至程式碼中，或者進行加密。

簡介

- 在iOS的行動應用App中，其應用程式的設定常儲存在某些情況下會受到影響的plist文件。同樣地，Android的開發商往往將App的設定儲存在共享的喜好XML文件或SQLite資料庫設定，預設不會加密，並可以讀取或甚至可用root權限進行修改。
- 在可行的情況下儘可能編譯設定到程式碼。有一點好處是於通過配置在iOS plist文件的應用程式，因為變更必須捆綁，且無論如何均需部署為新的應用程式。相反地攻擊者需要包括更多的時間和技能，以修改應用程式程式碼中的配置。不要存放在檔案目錄或其他文件中包含任何密鑰的設定，除非先進行加密。理想的情況下，使用由使用者提供的密碼的主密鑰，去加密所有的設定文件，或由遠端提供的一個密鑰。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	N/A

不安全程式碼範例：

iOS：Objective-C

Application Setting Read Example

- *將設定置於 plist 設定檔，容易被破解後修改*

```
//取得 plist 檔案路徑
  NSString *filePath = [[NSBundle mainBundle] pathForResource:@"secrets" ofType:@"plist"];
  NSFileManager *fileManager = [NSFileManager defaultManager];
  [self AppendLog:[NSString stringWithFormat:@"filePath=%@", filePath]];
  //判斷 plist 檔案存在才讀取
  if ([fileManager fileExistsAtPath: filePath]) {
      NSMutableDictionary *data = [[NSMutableDictionary alloc] initWithContentsOfFile:filePath];
      NSString *hash_password = [data objectForKey:@"hash_password"]; //從 data 中取值
      NSString *setting_lifes = [data objectForKey:@"setting_lifes"]; //從 data 中取值
  } else{
      [self AppendLog:@"file not exists"];
  }
```

安全程式碼範例：

iOS：Objective-C

- *protect application parameter : example code*
- *將設定或參數置於程式碼，或進行隱藏*

```
   //Stored as String in program, still not so safe to cracker
   NSString *APIKey = @"abcdef123456";
   // Stored as binary array to prevent strings attack
   // Stored using SHA(class name) XOR secret
   unsigned char obfuscatedSecretKey[] = { 0x3e, 0xcc, 0x9d, 0xca, 0x2f, 0x8a, 0xf2, 0xc1, 0x57,
0x01,  0xc2, 0x2e, 0x44, 0xea, 0x0d, 0xf5, 0x60, 0x09, 0x99, 0xe7, 0xb1, 0x13, 0x2a, 0x6f, 0x2c,
0xa2, 0xfe, 0x12, 0xbb, 0x58, 0x90, 0x85 };
```

敏感性資料傳輸(G)-CPG-01

CPG-01 行動應用App傳輸敏感性資料應規劃並實作傳輸全程全時使用TLS 1.1以上加密，維護敏感性資料機密性及完整性。

簡介

- 新的Web應用協定HTTP/2和SPDY並不支援SSL，只支援TLS。
- Google所有公共服務的加密都是使用TLS，故如果要整合Google的一些功能，只能使用TLS。
- PCI DSS規範SSL在2016年6月30日之後不得繼續使用。

Apple將在2017年1月強制所有iOS App開始使用ATS(App Transport Security, 支援TLS v1.2)，參考iOS-04。

Android支援標準TLS實作，在官方開發人員網站有完整的概念及實作介紹，並包含實例及常見問題。

開發生命週期	設計階段		
不安全程式碼範例	N/A	安全程式碼範例	N/A
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	4.1.2.3.1

使用者身分認證與授權(L)-CPL-01

CPL-01 行動應用App應設計並實作適當身分認證機制，並依使用者身分授權，以防止敏感資料被非授權人員存取。

簡介

- 開發實作人員需以「使用者意願」及「最小權限」為主要安全原則。
- 在App運行先後順序上以「1.識別使用者身分」、「2.徵詢使用者意願(企業存取政策)」及「3.取得權限」，所以使用者身分認證是行動應用App取用敏感性資料權限的必要機制。
- 行動應用App設計及開發人員應視法規及實務需求，設計且實作適當身分認證機制。

常見行動應用App身分認證方式

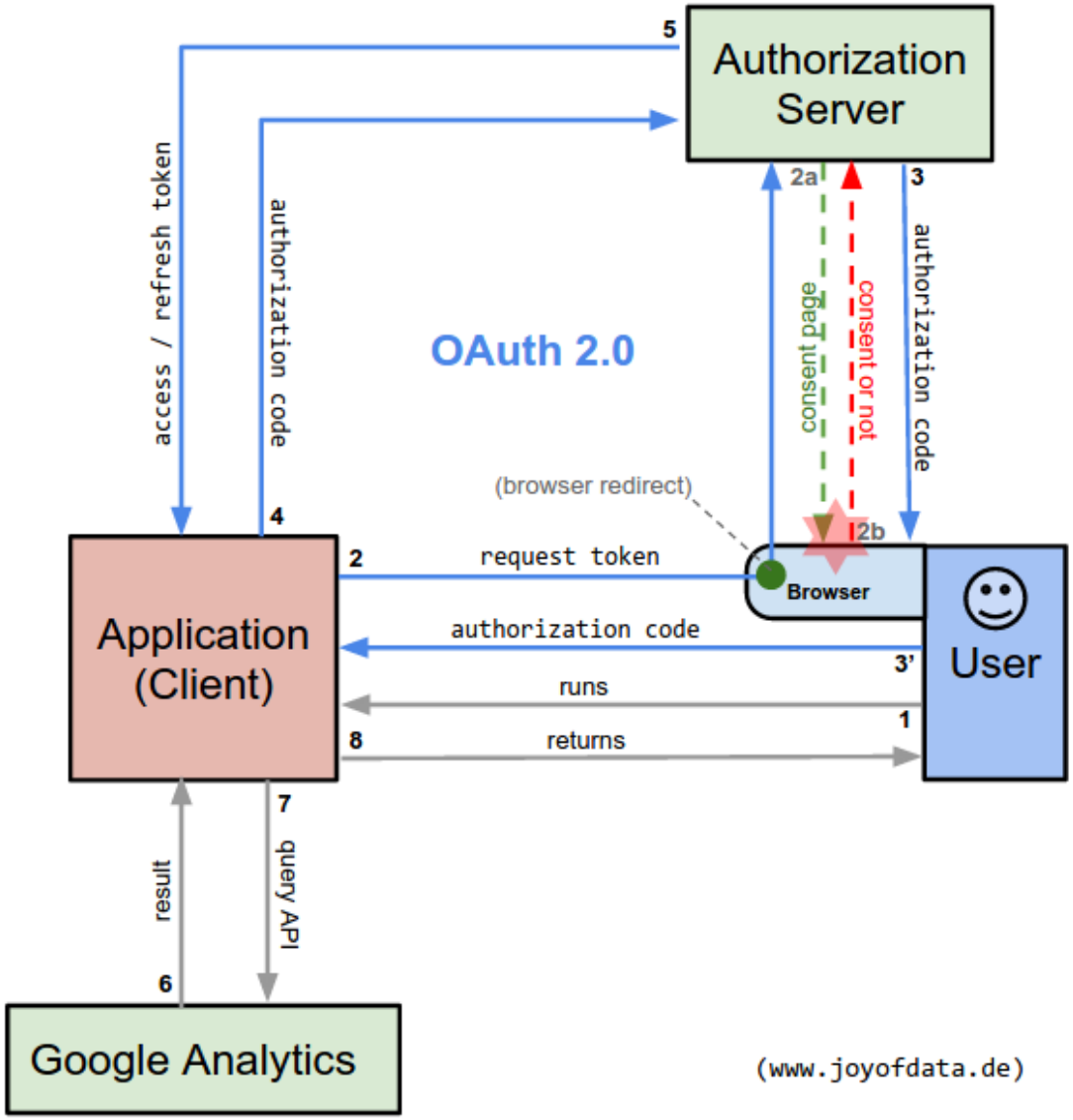
- 行動智慧裝置解鎖PIN碼
- 行動智慧裝置內建生物辨識(指紋、虹膜、面部、聲紋)
- 雲端服務認證：Apple ID、Google ID、Microsoft Live ID
- 開放式認證：OAuth 2.0
- App自建私有身分認證
- 企業整合認證：Microsoft AD、Novell LDAP

常見實作的技術

- 帳號 / 密碼
- 觸控螢幕滑動手勢
- PKI
- 多因素認證
- 生物辨識

現行最常運用的認證機制為OAuth 2.0，使用者可以使用Google、Facebook及Windows Live。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.4.1	行動應用App基本資安檢測基準	4.1.4.1.1 4.1.4.1.2


Authorization Server
5
access / refresh token
authorization code
4
2a
consent page
consent or not
2b
3
authorization code
OAuth 2.0
(browser redirect)
Application (Client)
2
request token
Browser
User
authorization code
3'
runs
1
8
returns
7
result
query API
6
Google Analytics
(www.joyofdata.de)

Validating the token

Tokens received on the fragment **MUST** be explicitly validated. Failure to verify tokens acquired this way makes your application more vulnerable to the confused deputy problem.

You can validate a token by making a web service request to an endpoint on the Google Authorization Server and performing a string match on the results of that web service request.

TokenInfo validation

Verifying a token using the Google Authorization Server endpoint is relatively simple. Your application includes the access token in the `access_token` parameter for the following endpoint:

Endpoint	Description
`https://www.googleapis.com/oauth2/v3/tokeninfo`	Accepts an access token and returns information about that access token including which application was it issued to, the intended target of the token, the scopes the user consented to, the remaining lifetime of the token, and the user ID.

Field	Description
`audience`	The application that is the intended target of the token.
`scope`	The space-delimited set of scopes that the user consented to.
`userid`	This field is only present if the `profile` scope was present in the request. The value of this field is an immutable identifier for the logged-in user, and may be used when creating and managing user sessions in your application. This identifier is the same regardless of the client ID. This provides the ability to correlate profile information across multiple applications in the same organization.
`expires_in`	The number of seconds left in the lifetime of the token.

安全程式碼範例：

iOS：Swift (參考 CPF-13 安全範例：使用 OAuth2)

- *通常會使用第三方帳號認證，茲提供 OAuth2 範例，以取得第三方的 Token，當作認證 ID*
- *OAuth 2 應用範例*

1.註冊並宣告服務

```
let googleConfig = GoogleConfig(
  clientId: "YOUR_GOOGLE_CLIENT_ID",                          // [1] Define a Google configuration
  scopes:["https://www.googleapis.com/auth/drive"])             // [2] Specify scope

  let gdModule = AccountManager.addGoogleAccount(googleConfig)     // [3] Add it to AccountManager
  self.http.authzModule = gdModule                              // [4] Inject the AuthzModule
                                                              // into the HTTP layer object

  let multipartData = MultiPartData(data: self.snapshot(),         // [5] Define multi-part
          name: "image",
          filename: "incognito_photo",
          mimeType: "image/jpg")
  let multipartArray =  ["file": multipartData]

  self.http.POST("https://www.googleapis.com/upload/drive/v2/files",   // [6] Upload image
              parameters: multipartArray,
     completionHandler: { (response, error) in
    if (error != nil) {
      self.presentAlert("Error", message: error!.localizedDescription)
    } else {
      self.presentAlert("Success", message: "Successfully uploaded!")
    }
  })
```

2.註冊 App 應用連結

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>com.raywenderlich.Incognito</string>
        </array>
    </dict>
</array>
```

3.接收 App 應用連結

```
func application(application: UIApplication,
 openURL url: NSURL,
 sourceApplication: String?,
 annotation: AnyObject?) -> Bool {
   let notification = NSNotification(name: AGAppLaunchedWithURLNotification,
     object:nil,
     userInfo:[UIApplicationLaunchOptionsURLKey:url])
   NSNotificationCenter.defaultCenter().postNotification(notification)
   return true
}
```

使用者身分認證與授權(L)-CPL-04

CPL-04 避免在行動應用App中直接使用設備ID作為使用者身分追蹤識別作為唯一因素，建議以帳號及密碼作為主要因素，設備ID可作為多因素認證的其他因素。

簡介

- 雖然作業系統有提供唯一裝置識別碼，但依據保護個人資料與資料使用的規則，不建議繼續使用此資料當作APP的主要識別碼。
- iOS部分：目前此裝置識別碼已被保護，程式碼已經抓不到正確的裝置碼，且每次安裝時，取得的資料，就會改變一次。雖然仍可被部分使用，但實際上沒有唯一性，只可被當作參考用途。
- Android部分：雖然可以取得SIM card的唯一碼，但仍不建議單獨使用它為認證資料。因為如果設備可能沒有SIM card只有WiFi，那就無法取得唯一碼。另外SIM card的唯一碼是IMSI，也可能會被偽裝，無法完全信任。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.4.1	行動應用App基本資安檢測基準	N/A

安全程式碼範例：

iOS：Objective-C

- *Device.UUID 範例碼*

```
[NSString stringWithFormat:@"UUID=%@", [[[UIDevice currentDevice] identifierForVendor] UUIDString]];
```

Apple Push Notification Service Device Token

"Prior to iOS 7, the device token was the same across all app installations on a given device. Different apps on your phone, whether Tap Tap Revenge or USA Today, would utilize the same address, i.e., device token, to route the push notification to you. The security credentials that you pair with a message would ensure it made it to the right app. On iOS 7, Apple has gone one step further and made sure that device tokens are now different in every single app install. This helps further protect users' privacy by removing another phone-level identifier."

連線管理機制(M)-CPM-03

CPM-03 行動應用App連線使用交談識別碼，應實作具備逾時失效(Session time-out)機制。

簡介

- 由於行動智慧裝置經常丟失或被盜，並且攻擊者可能利用一個應用程式來存取敏感資料，執行交易或研究設備所有者的帳戶。尤其是銀行或交易類的應用程式。建議行動應用App，在登入後也進行時間控管，以加強安全性。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.4.2	行動應用App基本資安檢測基準	4.1.4.2.1(3)

安全程式碼範例：

iOS：Objective-C

- *iOS: local-session-timeout code 實作*

1. 啟動時間計算

```
[myApp setLoginDate:[NSDate date]];
- (void) setLoginDate: (NSDate *) indate
{
    loginDate = indate;
}
```

2.檢查是否在安全時間內

```
[NSString stringWithFormat:@"isValidDate=%d",[myApp isLoginDateValid],
- (BOOL) isLoginDateValid
{
    if (!loginDate)
        return FALSE;
    //valid for 60 seconds
    NSDate *max_date = [loginDate dateByAddingTimeInterval:60];
    return
        ( ([loginDate compare:[NSDate date]] == NSOrderedAscending) &&
        ([max_date compare:[NSDate date]] == NSOrderedDescending) );
}
```

連線管理機制(M)-CPM-05

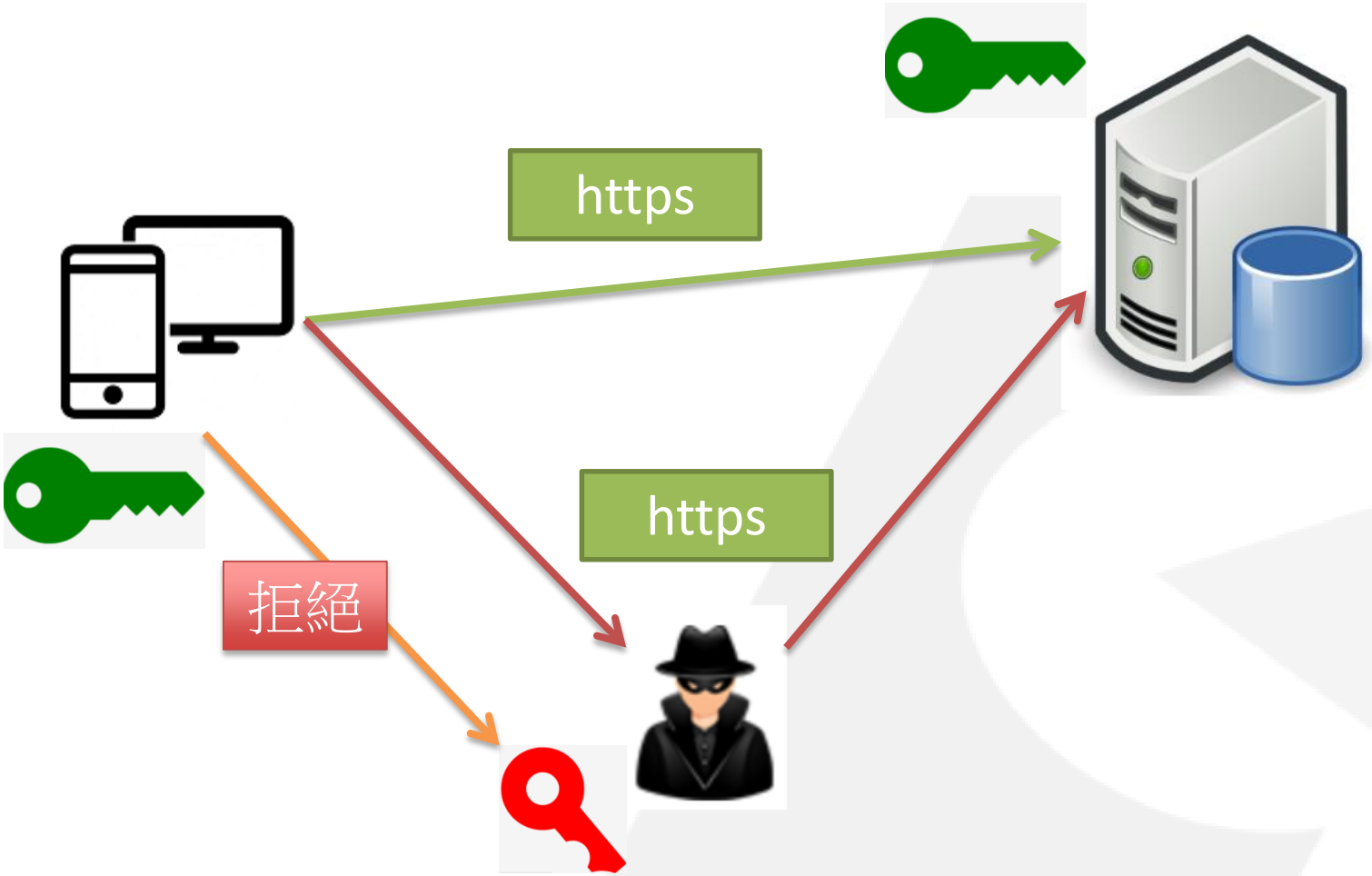
CPM-05 行動應用程式應使用憑證綁定(Certificate Pinning)方式驗證並確保連線之伺服器為行動應用程式開發人員所指定。

簡介

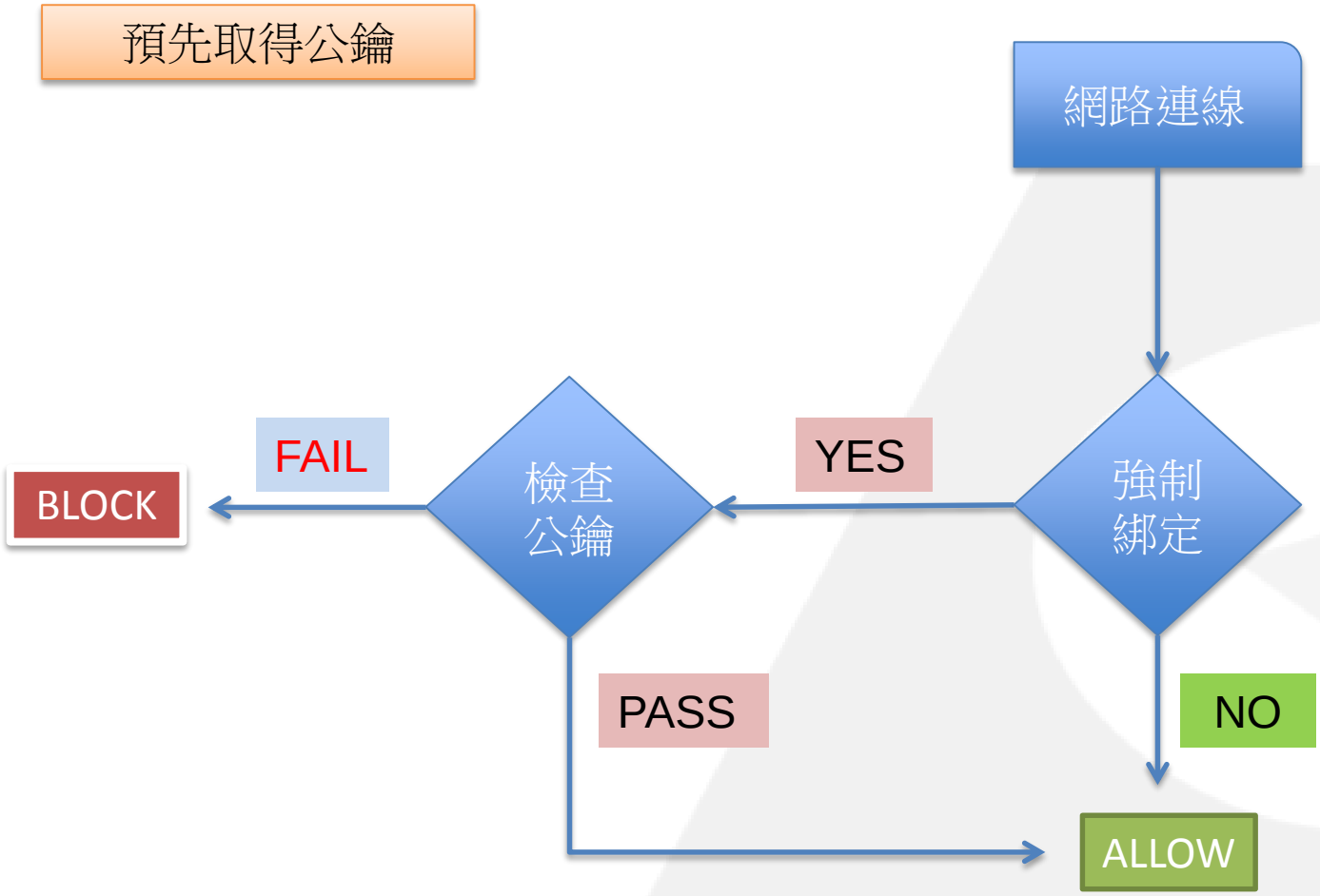
- 在全球漏洞資料庫網站揭露一個CVE(Common Vulnerabilities and Exposures)漏洞，漏洞編號為CVE-2014-6693。CVE漏洞報告指出，因為行動應用App沒有驗證x.509的CA(Certificate Authority)憑證，駭客可以藉此發動中間人攻擊，以竊取使用者敏感性資料。
- 開發人員應使用憑證綁定(Certificate Pinning)的方式，把需要比對的信任發行者發行憑證預先存放在App裡，指定特定Domain就只能使用特定憑證，等到要進行SSL Handshake的時候，再與伺服器的憑證進行比對。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.4.2	行動應用App基本資安檢測基準	4.1.4.2.2(2)

中間人攻擊（Man-in-the-middle attack, MITM）



SSL Pinning 憑證綁定



取得公鑰

1.由網站獲取公鑰，以利 **https ssl ceriticate** 認證時比對
(mac terminal console)

```
ex +'/BEGIN CERTIFICATE/,/END CERTIFICATE/p' <(echo | openssl
s_client -showcerts -connect www.owasp.org:443) -scq > owasp.crt
```

2.然後轉成**der**

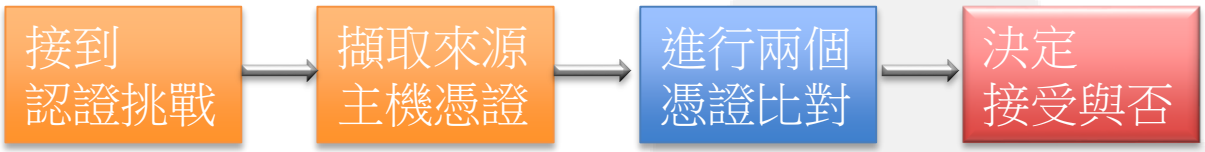
```
openssl x509 -outform der
    -in owasp.crt -out owasp2.der
```

```
-----BEGIN CERTIFICATE-----
MIIFKzCCBBOgAwIBAgIJAIkFWlbWBa0uMA0GCSqGSIb3DQEBCwUAMIG0MQswCQYD
VQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTETMBEGA1UEBxMKU2NvdHRzZGFsZTEa
MBgGA1UEChMRR29EYWRkeS5jb20sIEluYy4xLTArBgNVBAsTJGh0dHA6Ly9jZXJ0
cy5nb2RhZGR5LmNvbS9yZXBvc2l0b3J5LzEzMDEGA1UEAxMqR28gRGFkZHkgU2Vj
dXJlIENlcnRpZmljYXRlIEF1dGhvcml0eSAtIEcyMB4XDTE2MDQxODIwMzEzOFoX
DTE3MDQxODIwMzEzOFowOzEhMB8GA1UECxMYRG9tYWluIENvbnRyb2wgVmFsaWRh
dGVkMRYwFAYDVQQDEw13d3cub3dhc3Aub3JnMIIBIjANBgkqhkiG9w0BAQEFAAOC
AQ8AMIIBCgKCAQEAoDkpEOgbBup6GsLVFCcf3UXdLQQkBOA4FRocvxKJmqe/J+AC
E6ePX0WYDk+parc376Kp+P9kGykP5FVU8HjUBn2uigUYuK4TgtUWemA3sY/jLWEq
nlIY+zsSkQ5T9ZL6LxubFmDGcWFWzwrhWYAnBGdBqh8Xt+J2IffM3T0xH6w4FJ+z
ZbcL5dkyibqvlMtpcUXoTrivhvXJJnqb8n4ZBWMJn9RYcqHe7Ch55O5x0Q8FfNFW
K25WlH0AGYd3r3Gc6ABOLNh3TPbC7qXE/JW9C9bo/toxlwkE1Y+V8H8B4lf7wZHP
k5Wi2P1qjsUp/uC5tXCVpzaKYa6+eJsTkY6m3wIDAQABo4IBtjCCAbIwDAYDVR0T
AQH/BAIwADAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUHAwIwDgYDVR0PAQH/
BAQDAgWgMDcGA1UdHwQwMC4wLKAqoCiGJmh0dHA6Ly9jcmwuZ29kYWRkeS5jb20v
Z2RpZzJzMS0yMjYuY3JsMF0GA1UdIARWMFQwSAYLYIZIAYb9bQEHFwEwOTA3Bggr
BgEFBQcCARYraHR0cDovL2NlcnRpZmljYXRlcy5nb2RhZGR5LmNvbS9yZXBvc2l0
b3J5LzAIBgZngQwBAgEwdgYIKwYBBQUHAQEEajBoMCQGCCsGAQUFBzABhhhodHRw
Oi8vb2NzcC5nb2RhZGR5LmNvbS8wQAYIKwYBBQUHMAKGNGh0dHA6Ly9jZXJ0aWZp
Y2F0ZXMuZ29kYWRkeS5jb20vcmVwb3NpdG9yeS9nZGlnMi5jcnQwHwYDVR0jBBgw
FoAUQMK9J47MNIMwojPX+2yz8LQsgM4wIwYDVR0RBBwwGoINd3d3Lm93YXNwLm9y
Z4IJb3dhc3Aub3JnMB0GA1UdDgQWBBT5UR6eSrU4zHzyXQk3dCMc6SUsAjANBgkq
hkiG9w0BAQsFAAOCAQEAilkB1fdrWAfObuwmRiWI+eXJmdVM4VKaJ/mPPkhIjG/D
yzRqBZIyFjMej/q3UZuJLwGVqRI/2hSgd1OccUaNlQ2Gf8xgIGQJrE7x2Gt6RnnL
Tim8HS3EOFBJ/33RxAV2ymYhKsTZ/tBYN9O7tTZKIerLe2z5a+Q44+K2baC9N6+r
bDRT9XHLFr0DPnG/B7yq2YRB3TeUx4U8Ak3YnEDoKCJyRX2ePKCjUbcYcI4zs71K
kXRgDoMjY20uf7/FoLlAsbtLFjoH6EdpQRziQWQP9N66BXGWj7IWIyK1TYEjcK6y
m3/tH8SClCUfohCy3JqOBxBWk0P8EZ2vEFjiasS3oA==
-----END CERTIFICATE-----
```

進行憑證綁定

1.需實作：元件NSURLConnectionDelegate或元件NSURLSession
2.客製事件，決定是否認可或拒絕此連線

```
- (void)connection:(NSURLConnection *)connection
didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge {
  認可: 回傳
return [[challenge sender] useCredential: [NSURLCredential credentialForTrust:
serverTrust]   forAuthenticationChallenge: challenge];
  拒絕: 回傳
return [[challenge sender] cancelAuthenticationChallenge: challenge];
```



```
- (void)connection:(NSURLConnection *)connection didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge
*)challenge {
   //Enable Pinning
   if ([[[challenge protectionSpace] authenticationMethod] isEqualToString:
NSURLAuthenticationMethodServerTrust]){
       do
       {
           SecTrustRef serverTrust = [[challenge protectionSpace] serverTrust];
           if(nil == serverTrust)
           {
               [self AppendLog:@"serverTrust is nil"];
               break; /* failed */
           }
           OSStatus status = SecTrustEvaluate(serverTrust, NULL);
           if(!(errSecSuccess == status))
           {
               [self AppendLog:@"SecTrustEvaluate status is errSecSuccess"];
               break; /* failed */
           }
           SecCertificateRef serverCertificate = SecTrustGetCertificateAtIndex(serve
          if(nil == serverCertificate)
           {
               [self AppendLog:@"serverCertificate is nil"];
               break; /* failed */
           }
           CFDataRef serverCertificateData = SecCertificateCopyData(serverCertificate);
           if(nil == serverCertificateData)
           {
               [self AppendLog:@"serverCertificateData is nil"];
               break; /* failed */
           }
          const UInt8* const data = CFDataGetBytePtr(serverCertificateData);
          const CFIndex size = CFDataGetLength(serverCertificateData);
          NSData* cert1 = [NSData dataWithBytes:data length:(NSUInteger)size];
          if(nil == cert1)
          {
              [self AppendLog:@"server cert is nil"];
              break; /* failed */
          }
```

接到認證挑戰

擷取來源主機憑證

```
        NSString *file = [[NSBundle mainBundle] pathForResource:@"domain" ofType:@"der"];
            NSData* cert2 = [NSData dataWithContentsOfFile:file];
            if(nil == cert1 || nil == cert2)
            {
                [self AppendLog:@"local pinning cert is nil"];
                break; /* failed */
            }
            const BOOL equal = [cert1 isEqualToData:cert2];
            if(!equal){
                [self AppendLog:@"certs not equal"];
                break; /* failed */
            }
            [self AppendLog:@"certs are GOOD"];
            // The only good exit point
            return [[challenge sender] useCredential: [NSURLCredential credentialForTrust: serverTrust]
                                  forAuthenticationChallenge: challenge];

        } while(0);
    }
      // Bad dog
    return [[challenge sender] cancelAuthenticationChallenge: challenge];
}
```

進行兩個憑證比對

1.主機之NSData
2.手機之der

決定接受與否

防範惡意程式碼與避免資訊安全漏洞(N)-CPN-07

CPN-07 行動應用App應避免使用內嵌瀏覽器元件技術和防止連線劫持。

簡介

- iFrame之應用可以將Web/WAP網站的資料嵌入App並進行資料互動。這種開發應用方式可能被包裝(Wrapper)網站執行點擊劫持攻擊。
- 點擊劫持是利用熱門的網站服務，作為竊取資訊或將使用者重導向到攻擊者控制網站的常見威脅。這類的iFrame攻擊主要目的是誘騙使用者點擊惡意程式，其目標是蒐集機密資訊或通過諸如跨網站指令碼(Cross-Site Scripting，常簡稱為XSS)連結，取得受影響電腦的控制權。這種攻擊通常需要嵌入程式碼，這類超出一般使用者的知識範圍內所執行腳本的形式，可以在使用者不知情下點擊顯示執行或其他功能的按鈕來觸發。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.1	行動應用App基本資安檢測基準	N/A

行動應用程式完整性(O)-CPO-03

CPO-03 行動應用App程式碼應運用人工或工具使之增加複雜度，並輔以限制除錯器使用、反追蹤、二進位剝離等措施，使惡意人士使用逆向工程方法分析程式碼難度增加。

簡介

- 攻擊者可藉由使用軟體逆向工程工具，逆向解開應用程式程式碼，並窺視程式的邏輯、破解程式保護，甚或盜取程式碼。另一方面，當手機端App被破解時，也會使得Server端的服務被窺伺或破解。
- 為了防止此問題，應適當加入保護措施，以保護程式碼內容與Server主機安全。
- 保護措施的建議方向有：

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.2	行動應用App基本資安檢測基準	N/A

安全程式碼範例：

iOS：Objective-C

提供偵測裝置是否處於 jail-broken 模式原始碼

```
+(BOOL)isJailbroken{
    #if !(TARGET_IPHONE_SIMULATOR)
    if ([[NSFileManager defaultManager] fileExistsAtPath:@"/Applications/Cydia.app"]) {
        return YES;
    }else if([[NSFileManager defaultManager]
fileExistsAtPath:@"/Library/MobileSubstrate/MobileSubstrate.dylib"]) {
        return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/bin/bash"]) {
        return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/usr/sbin/sshd"]) {
        return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/etc/apt"]) {
        return YES;
    }
    if (  [[UIApplication sharedApplication] canOpenURL:[NSURL
URLWithString:@"cydia://package/com.example.package"]]) {
        //Device is jailbroken
```

- *iOS:提供拒絕 DEBUG 模式原始碼（for release version）*

```
//拒絕 DEBUG 模式的執行
#import <dlfcn.h>
#import <sys/types.h>
typedef int (*ptrace_ptr_t)(int _request, pid_t _pid, caddr_t _addr, int _data);
#if !defined(PT_DENY_ATTACH)
#define PT_DENY_ATTACH 31
#endif
void disable_gdb() {
    void* handle = dlopen(0, RTLD_GLOBAL | RTLD_NOW);
    ptrace_ptr_t ptrace_ptr = dlsym(handle, "ptrace");
    ptrace_ptr(PT_DENY_ATTACH, 0, 0, 0);
    dlclose(handle);
}
int main(int argc, char * argv[]) {
    #if !(DEBUG) // Don't interfere with Xcode debugging sessions.
        disable_gdb();
    #endif
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
    }
}
```

使用者輸入驗證(Q)-CPQ-01

CPQ-01 行動應用App應實作驗證使用者輸入字串資料型別及長度之正確性，避免惡意輸入導致應用程式毀損、緩衝區溢位、各種注入攻擊發生。

簡介

- 即使是從應用程式產生的資料也有可能被截取和處理，這可能導致包括應用程式意外中止(產生含密鑰日誌)攻擊、緩衝區溢位(Buffer Overflow)及或隱碼攻擊(SQL Injection)等。在iOS中，這可以很容易地透過實作UITextFieldDelegate的方式進行防範。
- 實作適當的Web應用程式輸入安全控制機制，預設從客戶端的所有輸入應必須被視為不可信，而必須被有條件的驗證與處理。服務必須透過過濾和從應用程式和使用者驗證輸入，適當有條件的阻擋惡意輸入，輸入包括傳送前和所有的使用者輸入接收。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.4	行動應用App基本資安檢測基準	4.1.5.4.1(1) 4.1.5.4.1(2)

使用者輸入驗證(Q)-CPQ-02

CPQ-02 行動應用App需要驗證使用者輸入、伺服器端傳入及其他裝置輸入之資料，防止因Buffer overflow/underflow造成安全性漏洞。

簡介

- 因為程式設計缺陷造成緩衝區溢位或不足位，在C、Objective-C與C++程式碼中的堆疊(Stack)和堆積(heap)是安全漏洞的主要來源。雖然現今主流作業系統都有buffer overflow protection(緩衝區溢位保護/記憶體位置重新導向)，且多數程式語言都會檢查輸入防止buffer overflow或underflow，但是C、Objective-C的，和C++則無此功能。
- 建議可以遵循以下規則，讓此種攻擊的發生機率下降：

✓ 在需要使用緩衝區前，以”0”填滿所有的緩衝區。讓緩衝區的內容僅包含”0”，沒有任何敏感資訊。
✓ 經常檢查回傳值和失敗回傳。
✓ 當呼叫配置(allocation)或初始化(initialization)的函數(例如AuthorizationCopyRights)失敗時，不要使用其結果值，因為它有可能是舊的資料。
✓ 你可以使用「read」及其他類似的系統呼叫(System call)，以確認實際上所讀取的資料量，而不是使用一個預設的常數(constant)。如果執行系統呼叫後的回傳值不是預期的資料數量，就需視為執行失敗(fail)。

✓ 當程式在執行一個有字元數量限制的資料結構(data structure)時，務必驗證所輸入的字元是否如為預期。
✓ 避免將非C字串(CFStringRef物件、NSString物件、CFDataRef物件、帕斯卡串等等)轉換成C字串，儘量直接使用既有的格式就好。如果真的要轉換，請務必檢查轉換後的C字串的長度，或是檢查其中有無空字元(Null)。
✓ 避免混合緩衝操作和字串操作，如果真的有必要使用，務必檢查轉換後的C字串的長度，或是檢查其中有無空字元(Null)。
✓ 儲存檔案時應以得防止被惡意篡改或截斷的方式進行。
✓ 避免整數值(Integer)的overflow和underflow。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.4	行動應用App基本資安檢測基準	N/A

不安全程式碼範例：

iOS：Objective-C

- *strcat,strcpy,strncat,strncpy,sprintf,vsprintf,gets*

函式不會控制資料結尾，除了長度不安全以外，也會造成沒有結尾。

```
int len = （int）self.myInput.text.length;
char buf[len+1];
//strncpy
strncpy（buf,  [self.myInput.text UTF8String] , len）;
```

安全程式碼範例：

iOS：Objective-C

- *strlcat,strlcpy,strlcat,strlcpy,snprintf,asprintf,vsnprintf,vasprintf,fgets*

函式會控制資料結尾，除了長度安全以外，也會有結尾。

```
int len = （int）self.myInput.text.length;
char buf[len+1];
//strlpy
strlcpy（buf, [self.myInput.text UTF8String] , len）;
```

使用者輸入驗證(Q)-CPQ-03

CPQ-03 行動應用App提供使用者輸入值儘量可以參數化(Query parameterization)。

簡介

- 由於SQL命令，若欄位參數採用字串組合，遇到特殊字元或惡意攻擊，會有安全漏洞，必須改用元件提供之欄位參數輸入方式。如下圖程式碼範例所示，如何建構最常見的程式語言的參數化查詢。這些程式碼的目的是展示給Web開發人員如何避免在Web應用程式中建立資料庫查詢的時候遭受SQL注入攻擊。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.4	行動應用App基本資安檢測基準	N/A

使用者輸入驗證(Q)-CPQ-05

CPQ-05 行動應用App應實作過濾使用者輸入及伺服器端傳入資料中易導致SQL injection之字串。

簡介

- 由於SQL命令，若欄位參數採用字串組合，遇到特殊字元或惡意攻擊，會有安全漏洞，必須改用元件提供之欄位參數輸入方式。
- 於CPQ-03也說明相關解決方式：使用Query parameterization。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.4	行動應用App基本資安檢測基準	4.1.5.4.2(1)

使用者輸入驗證(Q)-CPQ-08

CPQ-08 行動應用App應實作過濾使用者輸入及伺服器端傳入資料中易導致XML Injection之字串。

簡介

- XML注入攻擊者會嘗試在SOAP訊息中插入各種字串，目標在對XML結構注入各種XML標籤。通常一個成功的XML注入攻擊將導致限制操作的執行。根據各種執行的操作，而衍生各種安全問題。
- 由於XML的應用，在存取XML資料的時候，也須避免如隱碼攻擊(SQL Injection)的處理方式而造成的漏洞，或基於XML的衍生功能與漏洞進行限制，例如XML External Entity (XXE)。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.4	行動應用App基本資安檢測基準	4.1.5.4.2(1)

CPE-04 GPS資訊的處理或移除

```
-(void) saveImage:(UIImage *)imageToSave withInfo:(NSDictionary *)info
{
    // Get the image metadata (EXIF & TIFF)
    NSMutableDictionary * imageMetadata = [[info
objectForKey:UIImagePickerControllerMediaMetadata] mutableCopy];
    if (!imageMetadata) {
        imageMetadata = [[NSMutableDictionary alloc] init]; }
        // add (fake) GPS data
        CLLocationCoordinate2D coordSF = CLLocationCoordinate2DMake(37.732711,-
122.45224);
        // arbitrary altitude and accuracy
        double altitudeSF = 15.0; double accuracyHorizontal = 1.0; double
accuracyVertical = 1.0;
        NSDate * nowDate = [NSDate date];
        // create CLLocation for image
       CLLocation * loc = [[CLLocation alloc] initWithCoordinate:coordSF
altitude:altitudeSF horizontalAccuracy:accuracyHorizontal
verticalAccuracy:accuracyVertical timestamp:nowDate]
        if ( loc ) {
            [imageMetadata setObject:[self gpsDictionaryForLocation:loc]
forKey:(NSString*)kCGImagePropertyGPSDictionary]; }
        ….
```

編輯或
放入假資料

```
-(void)imagePickerController:(UIImagePickerController
*)pickerdidFinishPickingMediaWithInfo:(NSDictionary *)info{
    NSString *mediaType = [info objectForKey:UIImagePickerControllerMediaType];
    [self dismissModalViewControllerAnimated:YES];
    if ([mediaType isEqualToString:(NSString *)kUTTypeImage]) {
        UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
        //got image, put image to imageView
        imageView.image = image;
        if (newMedia) UIImageWriteToSavedPhotosAlbum(image,
self,@selector(image:finishedSavingWithError:contextInfo:), nil);
    }
   ….
```

移除GPS
只保留影像

CPE-05 敏感資料的處理與顯示

當顯示敏感資料(例如帳號/身分字號)，為了保護資料，有時候會被要求改由另一個代號來處理，可降低原始資料的洩漏問題。
另一種較常見的方式是隱藏部分(例如隱藏身分字號之456碼)

```
NSString *str = self.edit_input.text;
self.label_Account.text =
     [str stringByReplacingCharactersInRange:NSMakeRange(3, 3)
withString:@"***"];
```

CPF-09連線暫存問題

手機端：清除連線cache

```
[[NSURLCache sharedURLCache] removeAllCachedResponses];
[[NSURLCache sharedURLCache] setDiskCapacity:0];
[[NSURLCache sharedURLCache] setMemoryCapacity:0];
```

瀏覽器(網頁內容/html)：

```
<meta http-equiv="Cache-Control" content="no-cache, no-
store, must-revalidate" />
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="0" />
```

CPF-10 NSLog資料洩漏

於Release版本時，覆寫NSLog函式，以達到不洩漏資訊的目的

```
//NSLog rewrite when publishing
//disable NSLog when Release (ie. When not DEBUG)

#ifndef DEBUG
  #define NSLog(...)
#endif
```

CPF-14 Keyboard資料洩漏

- 建議關閉文字輸入框的自動校正功能，以免資料洩漏

myInput.autoCorrectionType = UITextAutocorrectionTypeNo;

- 建議關閉文字輸入框的拼字校正功能，以免資料洩漏
- myInput.spellCheckingType = UITextSpellCheckingTypeNo;

- 並可由手機設定中心，清除輸入的紀錄

General > Reset > Reset Keyboard Dictionary.
一般 > 重置 > 重置鍵盤辭典

CPF-15文字輸入框禁止複製/貼上選單

監聽TextField/TextView事件

```
@interface ViewControllerPaste : UIViewController
<UITextFieldDelegate, UITextViewDelegate>

當敏感輸入元件有事件(Copy等)時，不允許Copy/Paste Menu

(BOOL)canPerformAction:(SEL)action withSender:(id)sender {
    //identy those who not allow copy/paste
    [myInput resignFirstResponder];
    return NO;
}
```

課程大綱

第1單元	iOS安全行動應用設計最佳實務	2.5小時
	共通實務(Common Practices)	
	iOS實務	
	Server實務	
第2單元	行動應用App安全檢測流程與工具	0.5小時

iOS-01

iOS-01 謹慎使用Keychain儲存密碼(Use the Keychain carefully)。

簡介

- 為保護資料儲存的安全性，iOS作業系統提供了鑰匙串(Keychain)功能。然而在一些情況下，Keychain可能會受到損害並遭到解密。
  - 使用iOS 7以上的版本時，若惡意第三方有機會存取到有加密的iTunes備份，Keychain就有可能被破解。因為當iTunes備份被啟用時，iOS會重新對Keychain進行加密，所以這時如果惡意第三方可以知道備份的加密密碼，那麼Keychain就有可能部分被解密。
  - 此外，Keychain在有進行越獄(jailbreak)的設備上，其存取控管也可能會失效。在有越獄的手機上，任何的應用程式都可能存取到別隻程式的Keychain內容。而對於那些本身就含有Bootrom漏洞(Bootrom exploit)的舊設備(例如iPhone4)，攻擊者可以藉由實體存取而破解Keychain。
- 當有使用Keychain儲存資料時，開發人員應使用最嚴格的防護類別(可參考kSecAttrAccessible屬性)，而且使用該種類別(class)完全不會影響到App本身的運作流暢度。例如，若你的應用程式並不是設計於背景(background)執行的，可使用kSecAttrAccessibleWhenUnlocked或kSecAttrAccessibleWhenUnlockedThisDeviceOnly。
- 若要避免因iTunes備份讓Keychain曝光，可以使用「ThisDeviceOnly」保護類別。
- 對於高度敏感的資料，可考慮使用Keychain所提供的另一種更安全的保護機制，即應用程式層的加密機制。例如在進入應用程式時，使用者會輸入通行密碼(passphrase)以進行認證，並在資料儲存到Keychain前，用此通行密碼對資料進行加密。

不安全的程式碼	使用其他屬性的參數，造成可能的漏洞，例如備份檔被其他機器還原。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	N/A

iOS-02

iOS-02 為保護敏感資料不被以截圖方式儲存於檔案系統，行動應用App使用API設定或編寫程式阻擋敏感資料區快照功能(Snapshots)或以覆蓋方式清除。

簡介

- 為了提供在界面上的APP切換，iOS裝置已被證明在切到背景時會儲存螢幕快照。而這個快照會被存放在一個資料夾，將可能衍生安全問題。為了保護敏感資料，於螢幕切換時進行畫面保護，或隱藏安全疑慮的欄位是建議的安全方法。

開發生命週期	開發實作階段		
不安全程式碼範例	N/A	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.2.3	行動應用App基本資安檢測基準	N/A

安全程式碼範例：Objective-C

- *提供蓋掉螢幕的範例碼、提供隱藏欄位的建議方法*
- *先製作滿版圖片 secure-image.png （320x568 for iPhone）*

```
- (void)applicationDidEnterBackground:(UIApplication *)application {
  // Use this method to release shared resources, save user data, invalidate timers, and store enough
 application state information to restore your application to its current state in case it is terminated
 later.
  // If your application supports background execution, this method is called instead of
 applicationWillTerminate: when the user quits.
      if (!self.backgroundImage) {
          UIImageView *myBanner = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"secure-
 image.png"]];
          self.backgroundImage = myBanner;
      }
      [self.window addSubview:self.backgroundImage];
  // you can hide security field here
    // 隱藏欄位
  // [viewControl.secure_field setHidden:TRUE];
}
- (void)applicationWillEnterForeground:(UIApplication *)application {
  // Called as part of the transition from the background to the inactive state; here you can undo many
 of the changes made on entering the background.
  if (self.backgroundImage)
      [self.backgroundImage removeFromSuperview];
  // you should visi security field here
  // 回復已隱藏欄位
    // [viewControl.secure_field setHidden:FALSE];
}
```

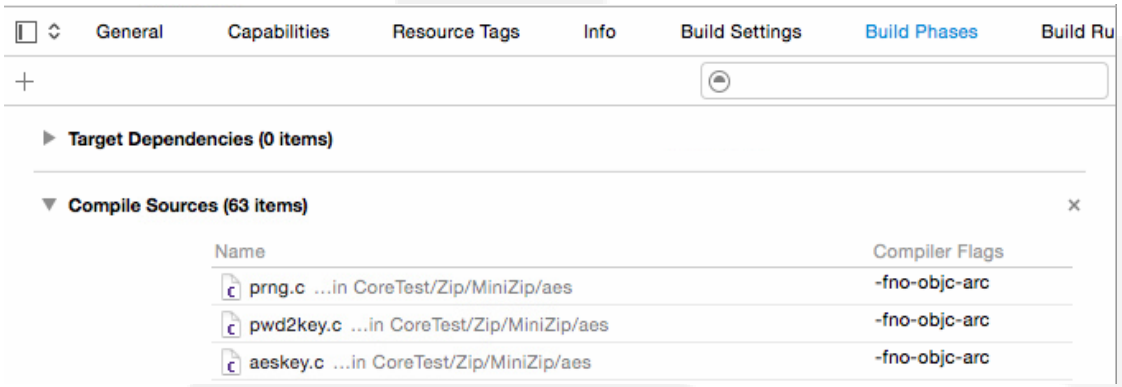
iOS-03

iOS-03 啟用自動引用計數(Automatic Reference Counting ,ARC)，避免記憶體物件弱點產生。

簡介

- 自動引用計數(ARC)是一個記憶體管理系統，在編譯/執行時會自動處理，而不是交由開發者的引用計數。此功能在iOS 5時推出，但也允許使用者自行管理記憶體釋放。而進一步，推出Switf語言時，物件與物件的強型鏈結，也造成需要注意的ARC項目。
- 【不啟用ARC】：當不啟用ARC的專案，記憶體需自行釋放。
- 【啟用ARC】：專案啟用ARC時，可定義編譯參數，以允許某些檔案-不啟用ARC。
- 強制關閉某Class方法：在專案->Build Phases->Compile Sources設定

Compiler Flags增加[-fno-obj-arc]



開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	4.1.5.1	行動應用App基本資安檢測基準	N/A

iOS-04

iOS-04 啟用App Transport Security(ATS)設定。

簡介

- 因應App網路需求，從iOS 9.0開始，增加應用程式傳輸安全(ATS)的新安全功能，並預設啟用。重新編譯的App就需考慮此設定，否則會造成無法使用非HTTPS的網路資源。
- Apple宣布2017年開始，所有App的資料連結必須啟用HTTPS才能上架。藉此要求提高了對執行基於HTTP的請求增加額外的安全性要求，並保護應用程式和網路服務之間的連接的保密性和資料完整性。
- ATS強調HTTP連接必須使用HTTPS(RFC 2818)。HTTPS請求必須使用安全通信的最佳實作，即TLS 1.2(RFC 5246)。
- ATS是由NSURLSession類和使用它的所有API執行。舊NSURLConnection的元件也受ATS管制。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	N/A
行動應用App基本資安規範	4.1.2.4	行動應用App基本資安檢測基準	N/A

課程大綱

第1單元	iOS安全行動應用設計最佳實務	2.5小時
	共通實務(Common Practices)	
	iOS實務	
	Server實務	
第2單元	行動應用App安全檢測流程與工具	0.5小時

Server-01

Server-01 與行動應用App連接之所有後端服務伺服器(包含網頁、資料庫及中介等)作業系統應有效強化及進行安全設定配置，並持續進行安全性程式修補。

簡介

- 減少資訊洩露
  - 攻擊者可因為獲得片段有用的伺服器資訊，提高攻擊成功的機率。
  - 正式環境中應避免揭露過於詳細的錯誤訊息，例如網頁的元件、版本、作業系統等。
  - 改善建議：降低Apache的版本資料、刪除某些預設的路徑或特別的安裝路徑。此外，管理者功能路徑，除非必要，否則不應提供公開存取(加入安全限制)。
- 強制使用HTTPS機制
  - 於網頁主機使用HTTPS機制(在header加上“Strict-Transport-Security”)，以保護連線。

開發生命週期	部署維運階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	N/A	行動應用App基本資安檢測基準	N/A

不安全程式碼範例：

網站主機：Server

例如：過時 OpenSSL 版本 OpenSSL/1.0.1e 與太多資訊洩漏。

```
telnet ---.---.---.tw 80

Trying 0.0.0.0……....

Connected to ---.---.---.tw.

Escape character is '^]'.

HEAD / HTTP/1.0
```

```
HTTP/1.1 200 OK
Date: Sat, 06 Aug 2016 01:57:04 GMT
Server: Apache/2.2.29 (Unix) mod_ssl/2.2.29 OpenSSL/1.0.1e-fips DAV/2 PHP/5.3.29
Last-Modified: Thu, 30 Jun 2016 06:22:49 GMT
ETag: "bc16f1-7f8-53678e6296040"
Accept-Ranges: bytes
Content-Length: 2040
Connection: close
Content-Type: text/html


遺失與主機的連線。
```

安全程式碼範例：

網站主機：Server

較少資訊洩漏

```
telnet www.------.com.tw 80

HTTP/1.0 200 OK

Date: Sat, 30 Jul 2016 05:52:56 GMT

P3P: policyref="http://info.------.com/w3c/p3p.xml", ... Cache-Control: max-age=3600, public

Vary: Accept-Encoding

Content-Type: text/html; charset=UTF-8

Server: ATS
```

Server-06

Server-06 網頁伺服器需預防網頁掛馬及點擊劫持攻擊。

簡介

- 什麼是網頁掛馬？點擊劫持？

網頁掛馬(Framing)	點擊劫持(clickjacking)
係指使用iFrame來遞送一個Web/WAP網站連線。這種攻擊可以用 “包裝(wrapper)” 網站來執行點擊劫持攻擊。	係指利用熱門的網站服務(例如Facebook)，通過諸如跨網站指令碼(Cross-Site Scripting，常簡稱為XSS) ，於頁面上嵌入iFrame或程式碼，以獲取受影響主機的控制權。此一網頁頁框的主要目的是誘騙使用者點擊嵌在iFrame的隱藏連結，將使用者重新導向到攻擊者控制的網站以竊取資訊的一種威脅。

- 防止Framing的方法
  - 不使用網頁視圖(Web View)及停用JavaScript。
- 專門設計的WebView的API可以被濫用來破解的WebView中指定的Web內容的安全性。可以實作下列機制以避免此種攻擊：
  - 阻止上傳頁框託管於其他域名的請求內容的X-Frame-Option HTTP response header。然而，與已被入侵主機連線時，這種緩解方法並不適用。
  - 利用內部防禦機制，以確保所有UI元素在頂層框架上傳;這樣就避免了在較低的層級水準，通過不信任的頁框設置服務內容。

開發生命週期	部署維運階段		
不安全程式碼範例	N/A	安全程式碼範例	N/A
行動應用App基本資安規範	N/A	行動應用App基本資安檢測基準	N/A

Server-10

Server-10 參考CPG-01~CPG-03實作TLS伺服器端設定。

簡介

- 目前符合公認安全的版本為TLS 1.2以上。

SSL/TLS更新歷程

定義	SSL 1.0	SSL 2.0	SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3
年份	NA	1995	1996	1999	2006	2008	待定

停用1024位元金鑰

美國NIST建議於2013之前，金鑰長度由1024位元，轉換成2048位元。因1024位元金鑰有可能會被破解，而2048位元金鑰於理論上並不會在有限的時間(數年內)被破解。

停用SSL 3.0

自從2014 SSL 3.0的CVE- 2014-3566漏洞會造成資料不安全，各大網站推廣停用 SSL3.0，改用TLS加密協定。

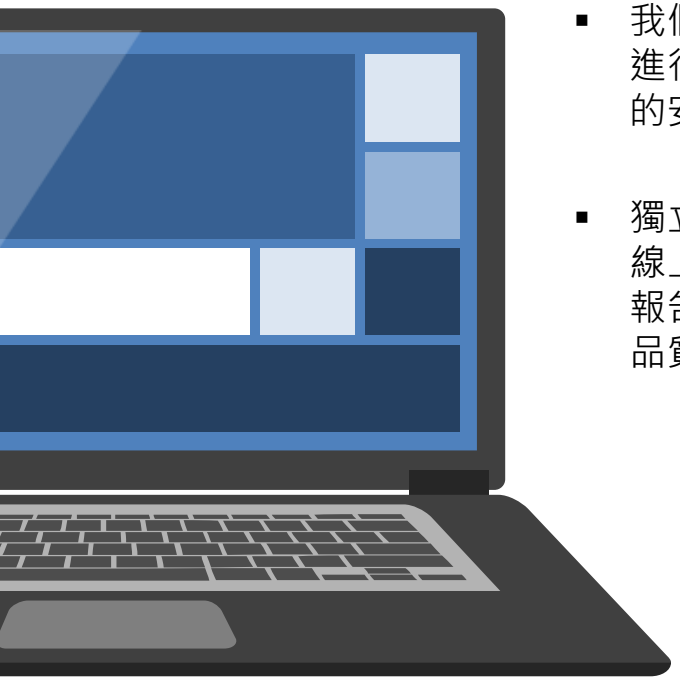
- OpenSSL heartbleed bug
  - OpenSSL大量被應用於網頁主機上，於2014/04/08發布1.0.1g修正此問題。此問題會影響包括SSL與TLS，會導致主機憑證私鑰被竊、主機資料被竊及客戶端資料被竊等議題。

開發生命週期	開發實作階段		
不安全程式碼範例	參考補充講義	安全程式碼範例	參考補充講義
行動應用App基本資安規範	N/A	行動應用App基本資安檢測基準	N/A

課程大綱

第1單元	iOS安全行動應用設計最佳實務	2.5小時
第2單元	行動應用App安全檢測流程與工具	0.5小時

行動應用App資安檢測實務

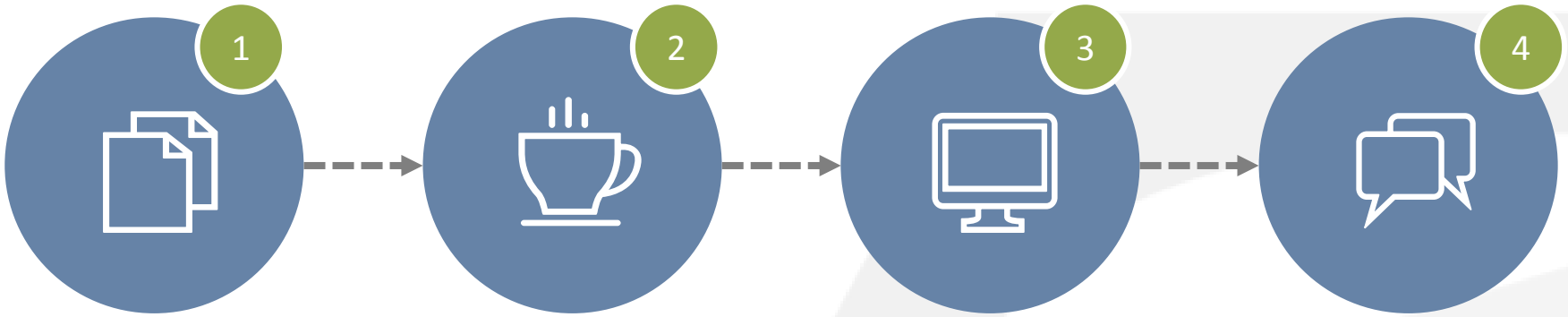


- 我們建議行動應用App安全檢測活動應由團隊測試人員或第三方單位人員進行檢測，透過靜、動態分析及黑、白箱等測試方式，針對行動應用App的安全性議題進行探討，有助於排除開發者自我開發之盲點。

- 獨立測試人員可由測試工程方法蒐集與探討相關惡意或可疑之函式庫、線上服務接口或相關第三方程式模組等，並且研擬相關測試方案、表單、報告與改善方法，有助於團隊之分工合作及提升行動應用App改版效率與品質。

行動應用App安全檢測流程

測試程序之4步驟



測試規劃

開發初期應規劃擬定相關測試計畫書，並載明測試目標與範圍及其相關方法。

測試準備

準備測試工具、環境及參考文獻，以順利執行測試計畫

App測試

執行App測試並記錄相關過程與回報問題。

評估控制

監控與檢討測試計畫，並評估其執行成效。

行動應用App安全檢測工具(1/5)

建議使用以下3個主要工具，可對iOS系統進行黑白箱測試。

Santoku　Snoop-it　MobSF

當開發人員與測試發布人員有溝通協作上的困難時，可考慮納入DevOps的開發概念，並可額外採用如HockeyApp這類的工具。

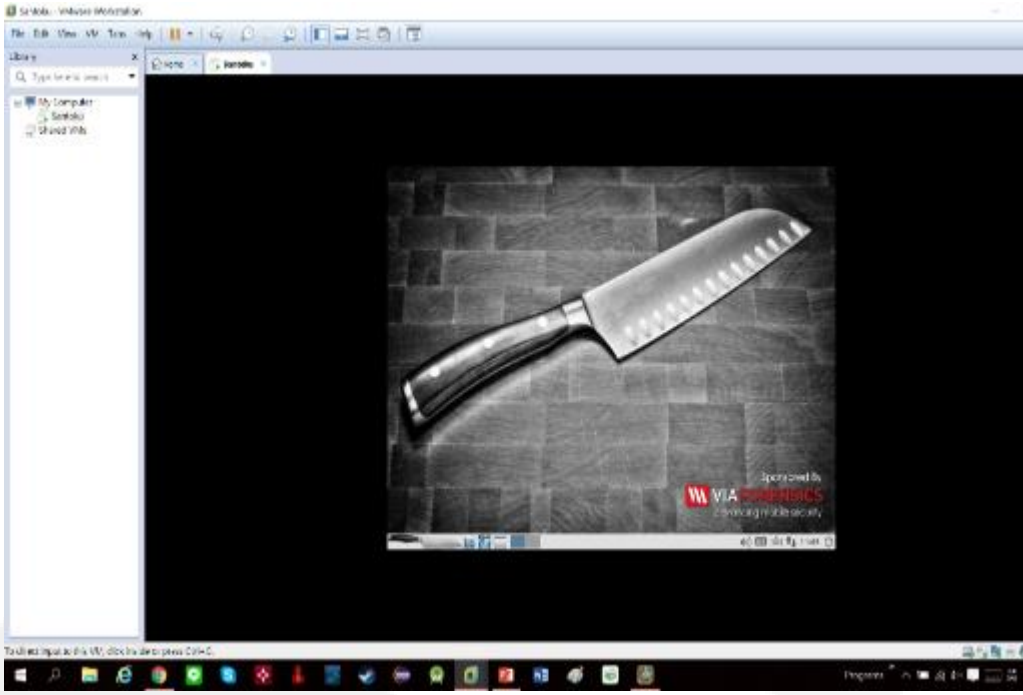
HockeyApp提供多樣性的行動開發工具，並具跨平台的行動應用程式測試功能，HockeyApp的功能包含錯誤報告(Crash Report)、App發布和回報等。

行動應用App安全檢測工具(2/5)

Santoku

- 適用於iOS及Android平台。
- 以Ubuntu Linux為基礎的整合工具環境作業系統，以提供使用者對手機或App進行檢測或分析。
- 主要功能
  - 行動裝置App鑑識
  - 行動裝置App惡意軟體分析
  - 行動應用安全檢測

主要採用其中BurpSuite進行行動應用App黑箱之網路檢測，探討其在網路資料傳輸或交換之安全項目。



Santoku操作環境

行動應用App安全檢測工具(3/5)

Santoku

https://santoku-linux.com/howto/mobile-forensics/howto-create-a-logical-backup-of-an-ios-device-using-libimobiledevice-on-santoku-linux/

1.備份/擷取整個裝置 (含iOS/Android)

A.一般衍生資料

Contacts/Call Logs/SMS/MMS/MMS Parts

B.硬體、密鑰、程式、相關歷史資料

Device info/Keychains(plist/sqlite)/白板BulletinBoard
暫存Caches(Safari/ WebAppCache)/Thumbnails/行事曆Calendar
商店購買資料(com.apple.itunesstored)
各種設定檔ConfigurationProfiles/網頁Cookies
鍵盤紀錄Keyboard/郵件Mail/地圖Maps(Bookmarks/ History)
音樂MusicLibrary/備忘錄Notes/Protocol設定檔(Preferences)
桌面(SpringBoard)/語音郵件Voicemail/短片WebClips
相簿/錄音/系統設定/無線網路/wireless

2.暴力破解Android密碼 (Brute Force Android Encryption)

行動應用App安全檢測工具(4/5)

Snoop-it

- 僅適用於iOS平台。
- 藉由debugging和runtime tracing功能檢測行動應用App，以幫助做為動態分析和黑箱安全評估的工具。
- 允許即時操作任意iOS系統架構的行動應用App，透過一個易於使用的界面，可以繞過客戶端限制或解鎖額外付費的功能和應用程序。
- 藉此測試行動應用App在安全驗證或資料安全之相關項目。

Snoop-it操作畫面

主要功能

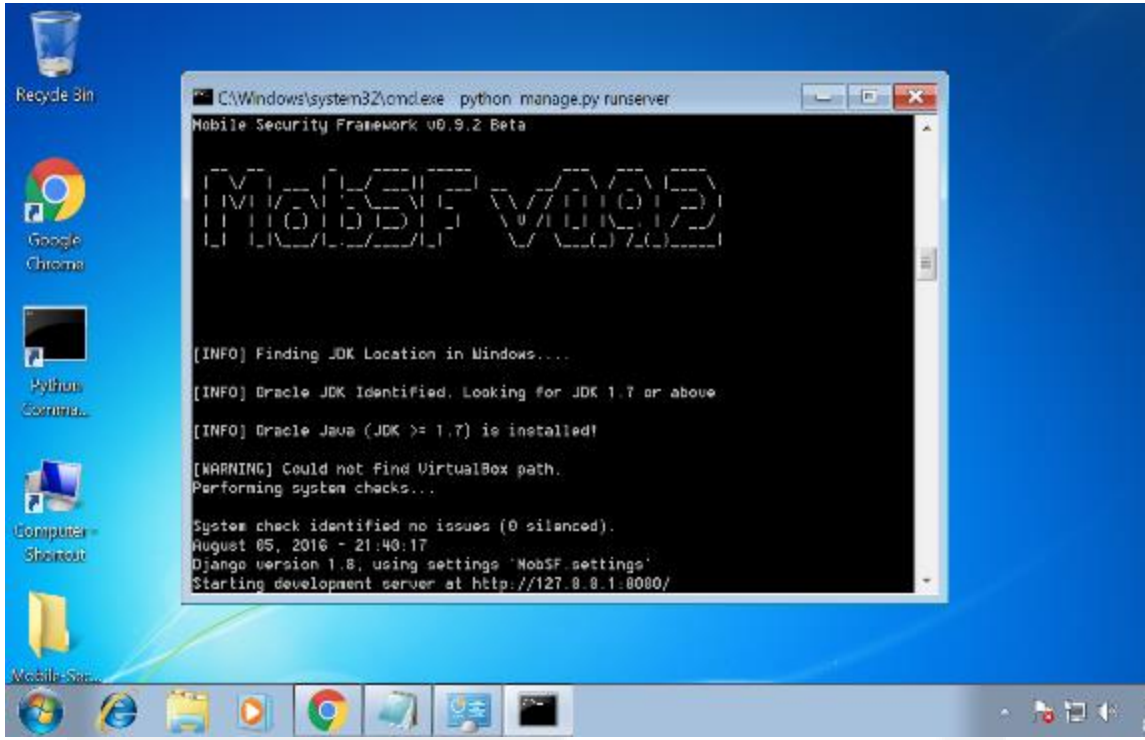
- Monitoring。
- Analysis/Manipulation。
- Simple installation and configuration。
- Easy to use graphical user interface。
- Plenty of filter and search options。
- Detailed description of the XML-RPC web service interface。

行動應用App安全檢測工具(5/5)

MobSF

- 適用於iOS及Android平台。
- 為一個行動App分析安全框架，具備多種功能可針對行動應用App(Android版/ iOS版)進行分析，這個測試框架能夠執行靜態和動態(僅適用於Android平台)分析。

建議開發者可採用由ajinabraham所發展的MobSF工具，有助於多數開發者可以較快上手進行使用。



MobSF執行畫面

行動應用App安全檢測實務

本單元以採用MobSF自動化工具檢測環境，並以某電子支付App為檢測範例。

APP基本資訊

- 檔案名稱：testsamlpe02.ipa
- 主要功能：行動支付
- 取得權限：官網未提供

本單元以透過取得該App的ipa檔案，進行實務操作分析。

自我基本檢測目標與流程

提供開發人員於行動App開發完成後的初步基本安全檢測，藉以善盡開發者安全開發的基本責任。

- 檢測開始前針對待測之行動應用App進行基本版本、權限等公開資訊分析。
- 透過自動化檢測工具/環境，分別進行靜及動態分析。
- 根據其所產生之報告中，發掘問題點的特定項目檢測。針對問題警訊，依據檢測項目不同，其選擇適用之檢測工具及方法。

檢測環境及檢測方式

工具運行作業系統

- Mac OS X EI Capitan v.10.11.6
- iOS 7.1.2

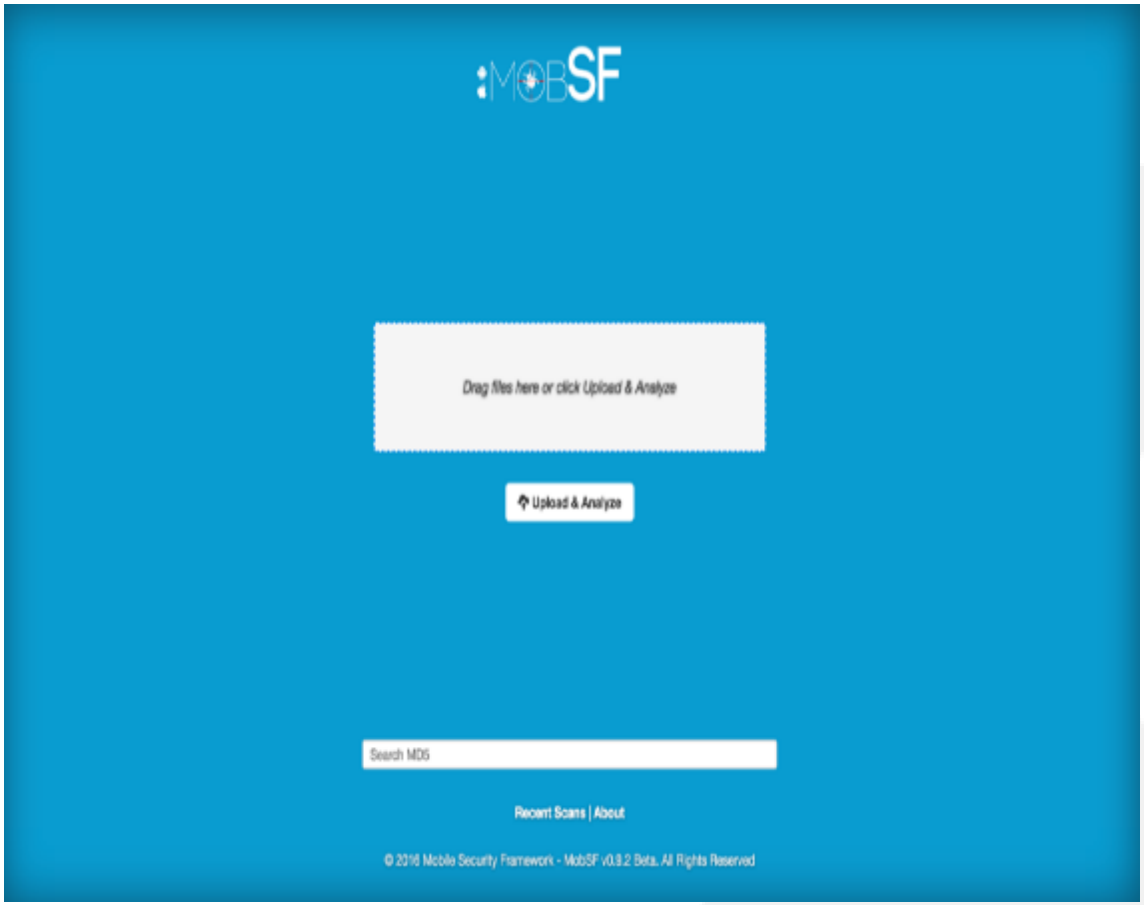
檢測工具/環境

- Mobile Security Framework(MobSF) v0.9.2 Beta(依工具要求環境建置)
- Snoop-it v.1.0.10(依工具要求環境建置)

檢測方法

- 黑箱測試
- 靜態分析

檢測操作步驟-靜態分析(MobSF)

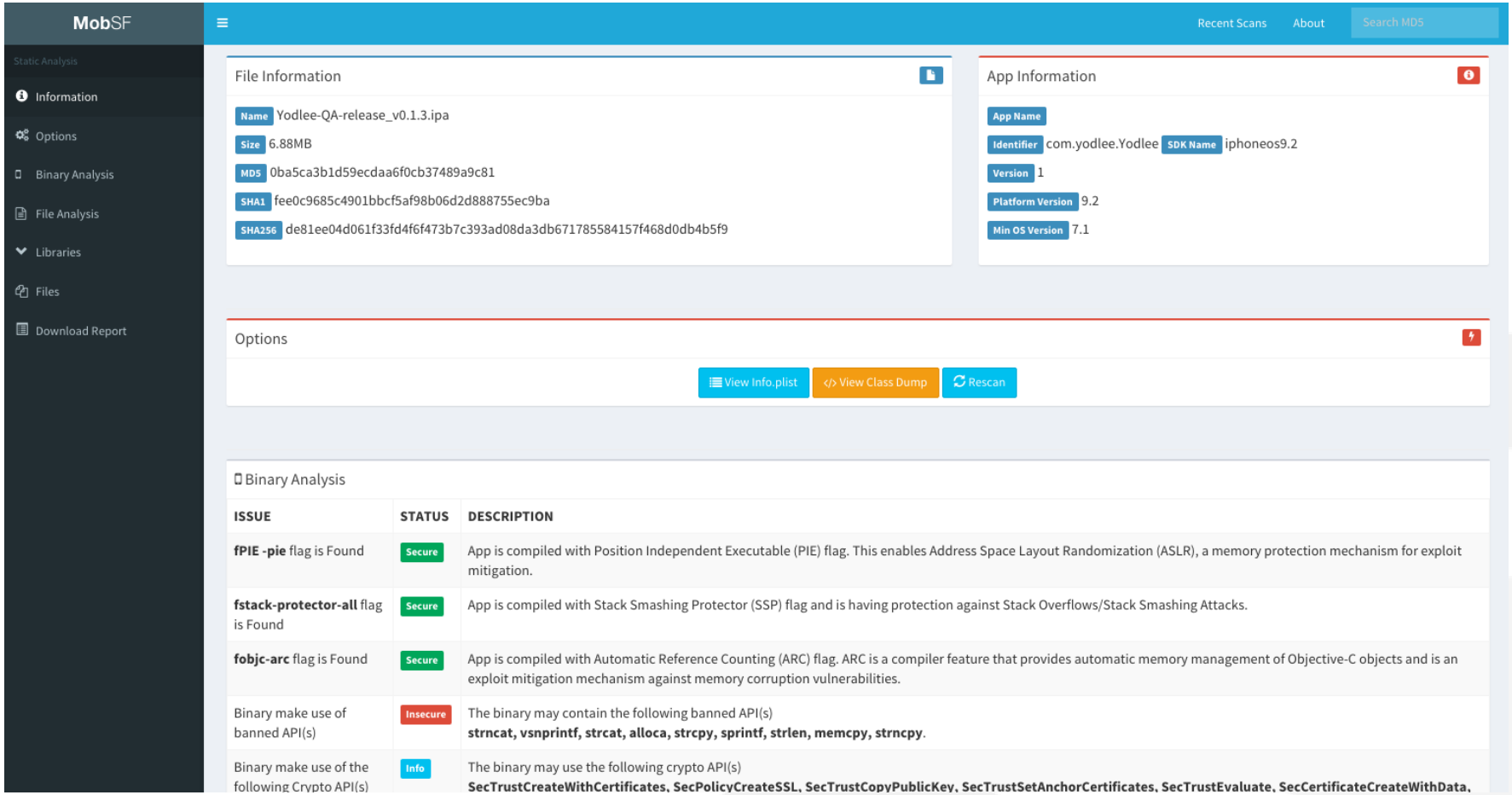


MobSF 開啟畫面

步驟一

上傳待測行動應用App

- 將欲檢測的行動App(App)上傳或拖曳至該測試平台。
- 支援檔案格式為apk、ipa檔。



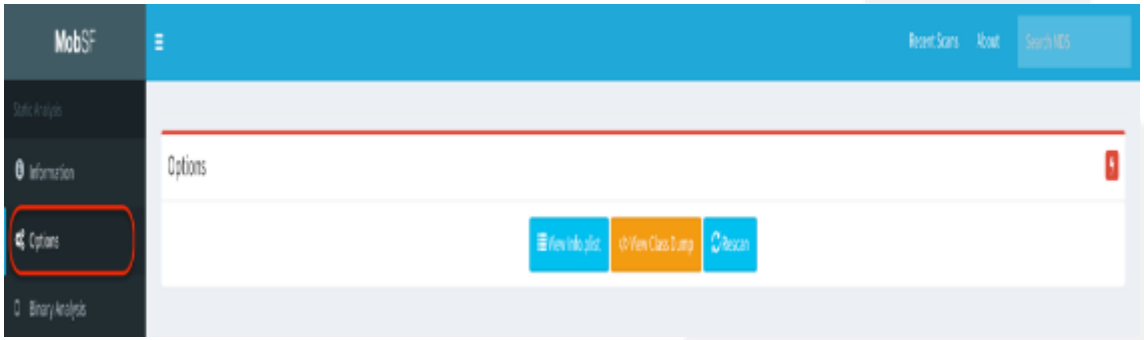
靜態分析：基本資訊、選項、原始碼分析、檔案分析、函示庫分析、報表

檢測操作步驟-靜態分析(MobSF)

靜態分析- Information

步驟二(1/3)

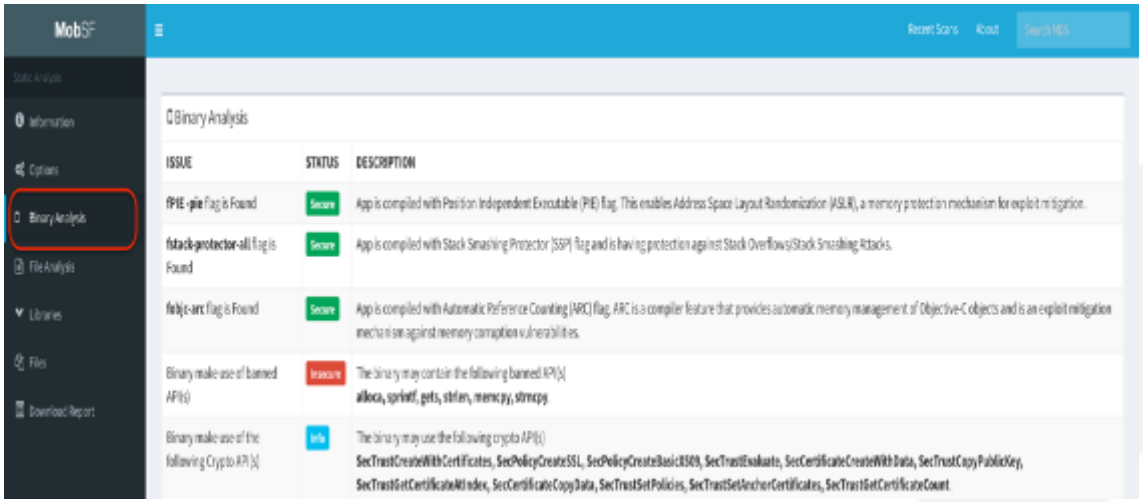
Information 內容
- 檔名
- MD5
- SALT
- SHA256
- App Name



靜態分析- Option

Option內容(進階分析)
- 下載 Info.plist
- 下載 Class Dump

檢測操作步驟-靜態分析(MobSF)

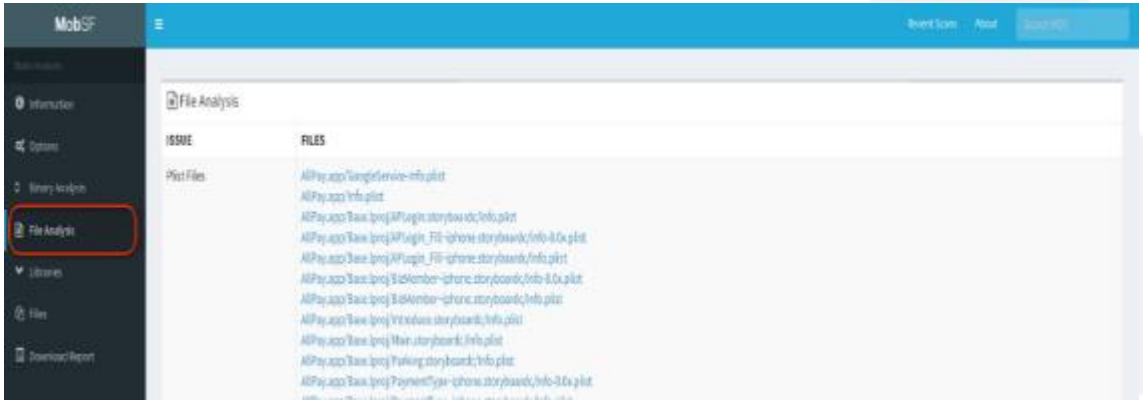


靜態分析- Binary Analysis

步驟二(2/3)

Binary Analysis 內容

安全設定
不安全函式
有使用之加密函式



靜態分析- File Analysis

File Analysis內容

列出所有檔案名稱

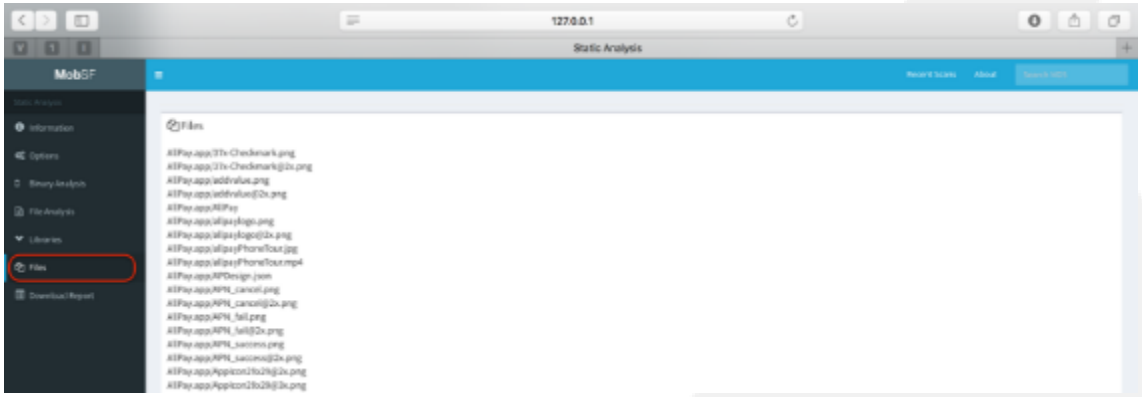
檢測操作步驟-靜態分析(MobSF)

靜態分析- Libraries

步驟二(2/3)

Libraries 內容

列出使用之元件

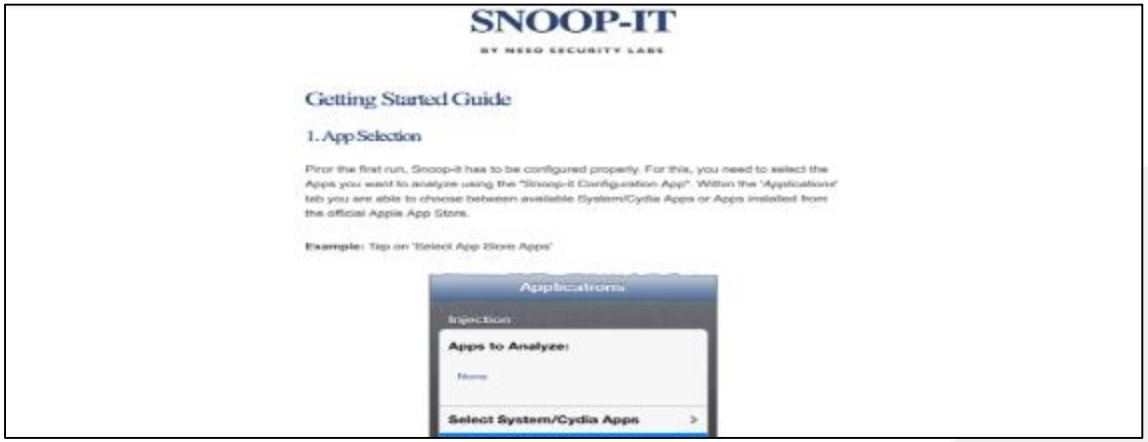


靜態分析- Files

Files內容

列出所有檔案名稱

檢測操作步驟-動態分析(Snoop-It)



動態分析-開啟畫面(瀏覽器)

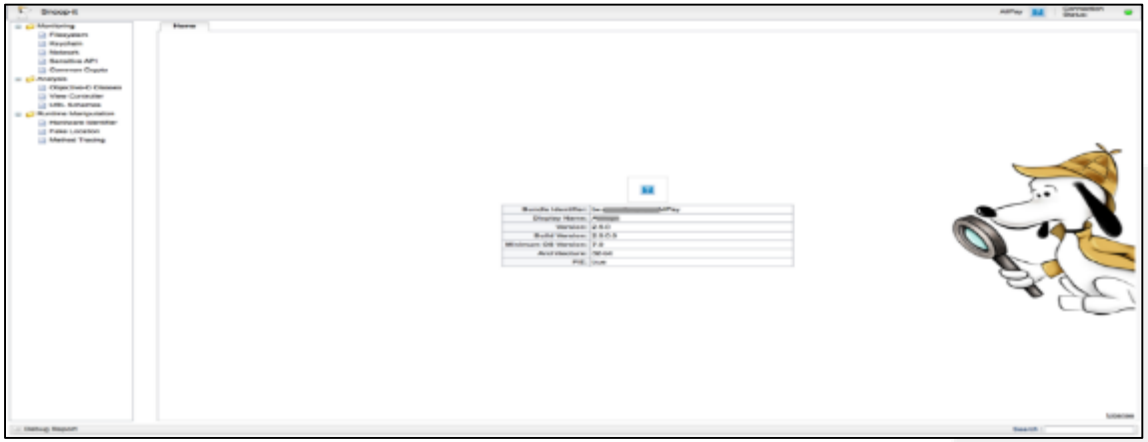
動態分析-開啟畫面(行動裝置)

步驟三

動態分析-載入測試平台

- 基於MobSF僅支援Android作業系統的動態分析，故在此提供另一套專門針對iOS平台的檢測工具Snoop-it。
  - ✓ 行動裝置：先開啟Snoop-it，進行相關環境設定，記下其所提供之IP位址及端口，同時載入待測的行動應用App。
  - ✓ 電腦端：開啟網頁瀏覽器，引導至Snoop-it所提供之IP位址及端口。

檢測操作步驟-動態分析(Snoop-It)



動態分析-測試環境

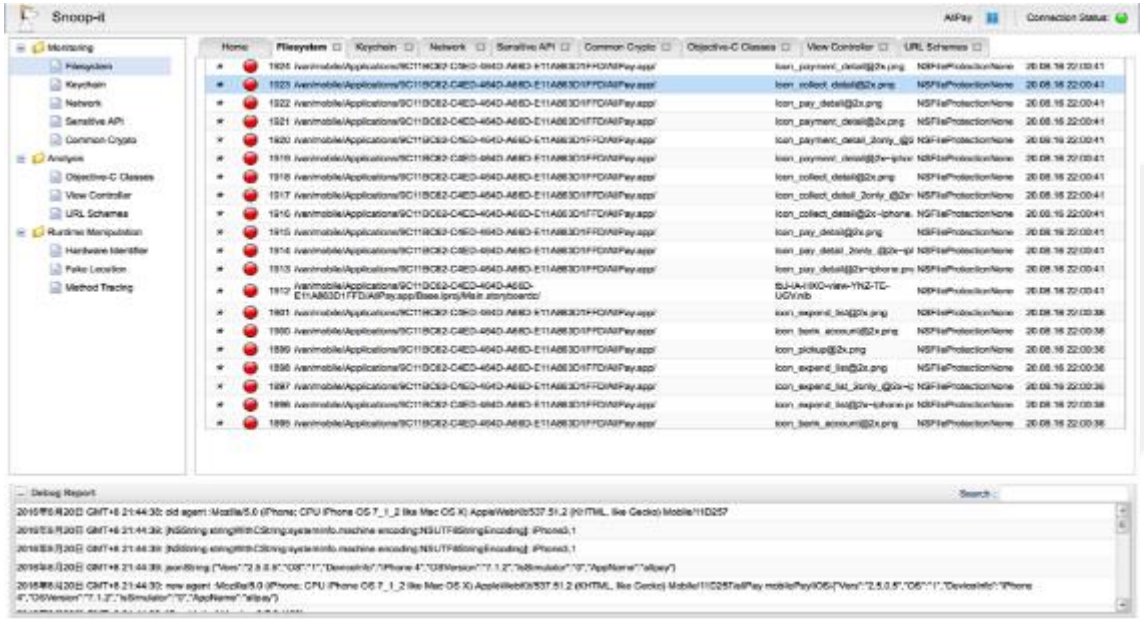
動態分析- 測試環境

步驟四(1/9)

動態分析-進行即時檢測

- 基於MobSF僅支援Android作業系統的動態分析，故在此提供另一套專門針對iOS平台的檢測工具Snoop-it。
  - ✓ 行動裝置：此時切換至待測之行動應用App的畫面。
  - ✓ 電腦端：重新整理網頁瀏覽器，此時即載入待測之行動應用App。

檢測操作步驟-動態分析(Snoop-It)



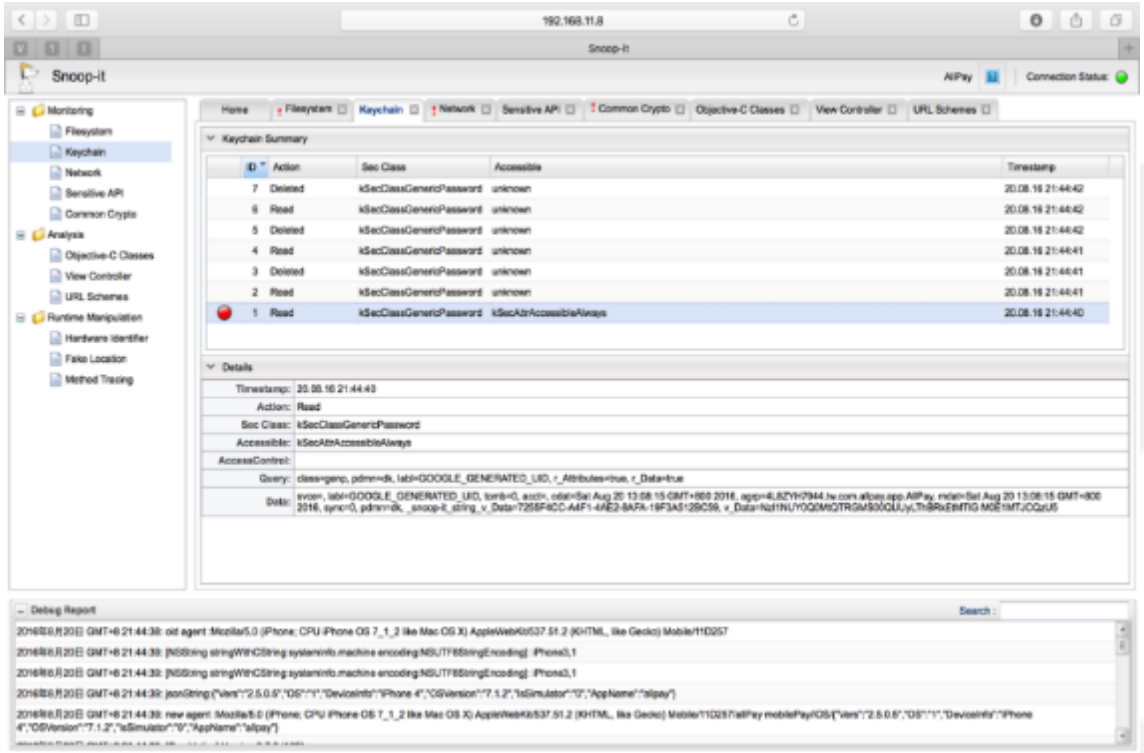
動態分析- Monitoring(Filesystem)

步驟四(2/9)

動態分析-進行即時檢測

- 檢測功能區分成Monitoring及Analysis等2大類。
- 並可運用其他工具下載Keychain等資料進行進階分析。

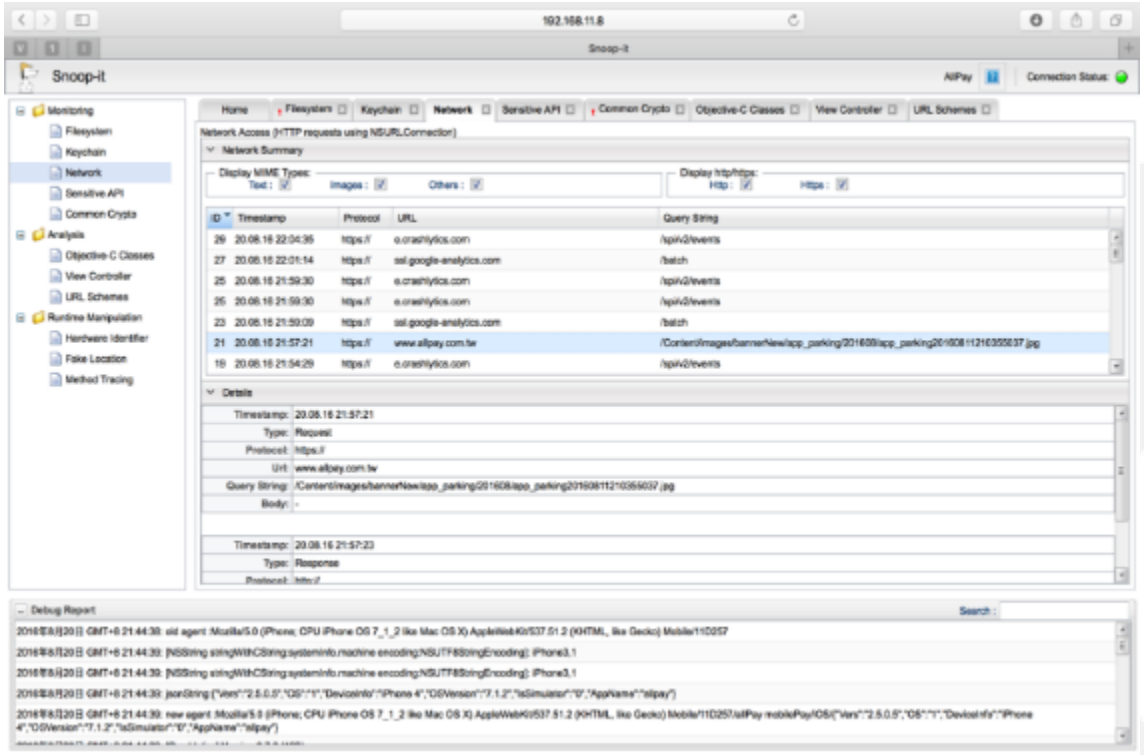
檢測操作步驟-動態分析(Snoop-It)



動態分析- Monitoring(Keychain)

步驟四(3/9)

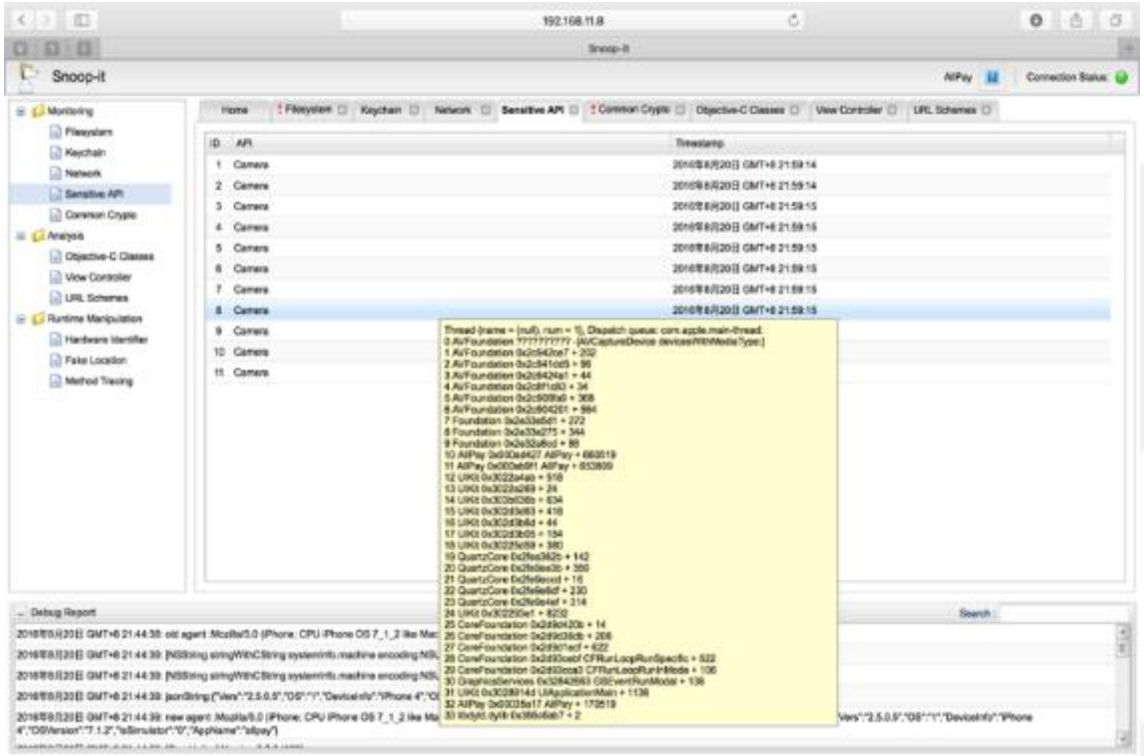
檢測操作步驟-動態分析(Snoop-It)



動態分析- Monitoring(Network)

步驟四(4/9)

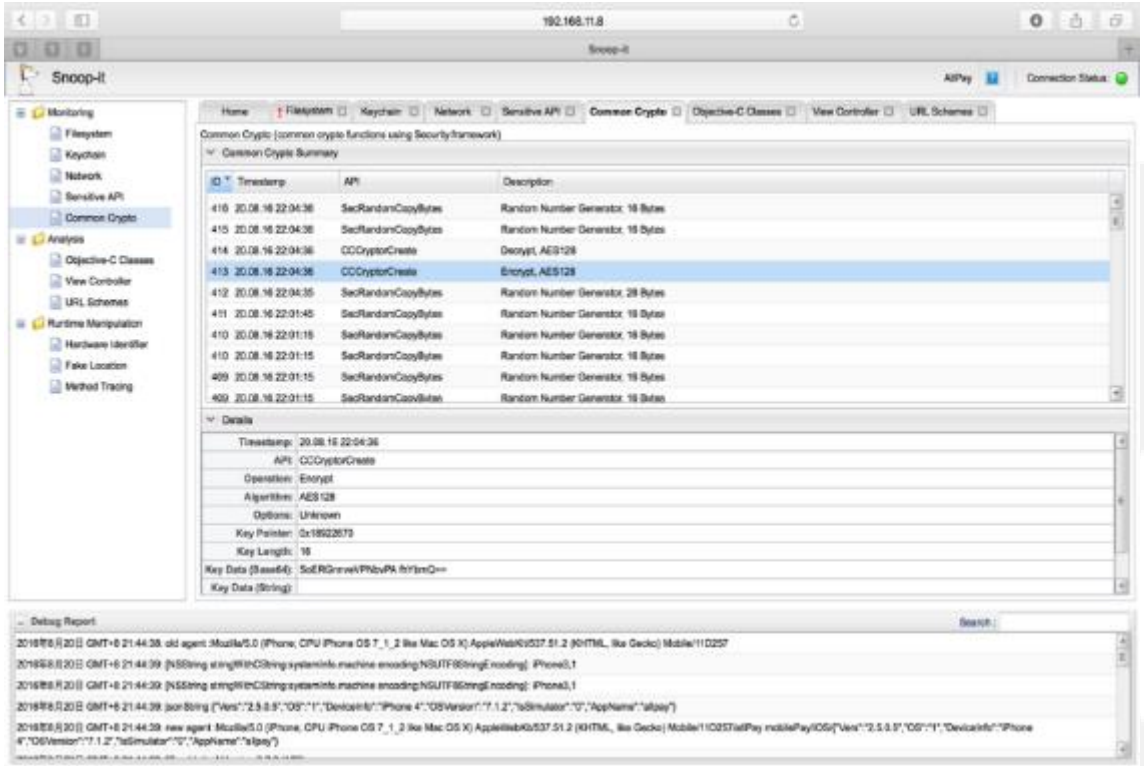
檢測操作步驟-動態分析(Snoop-It)



動態分析- Monitoring(Sensitive API)

步驟四(5/9)

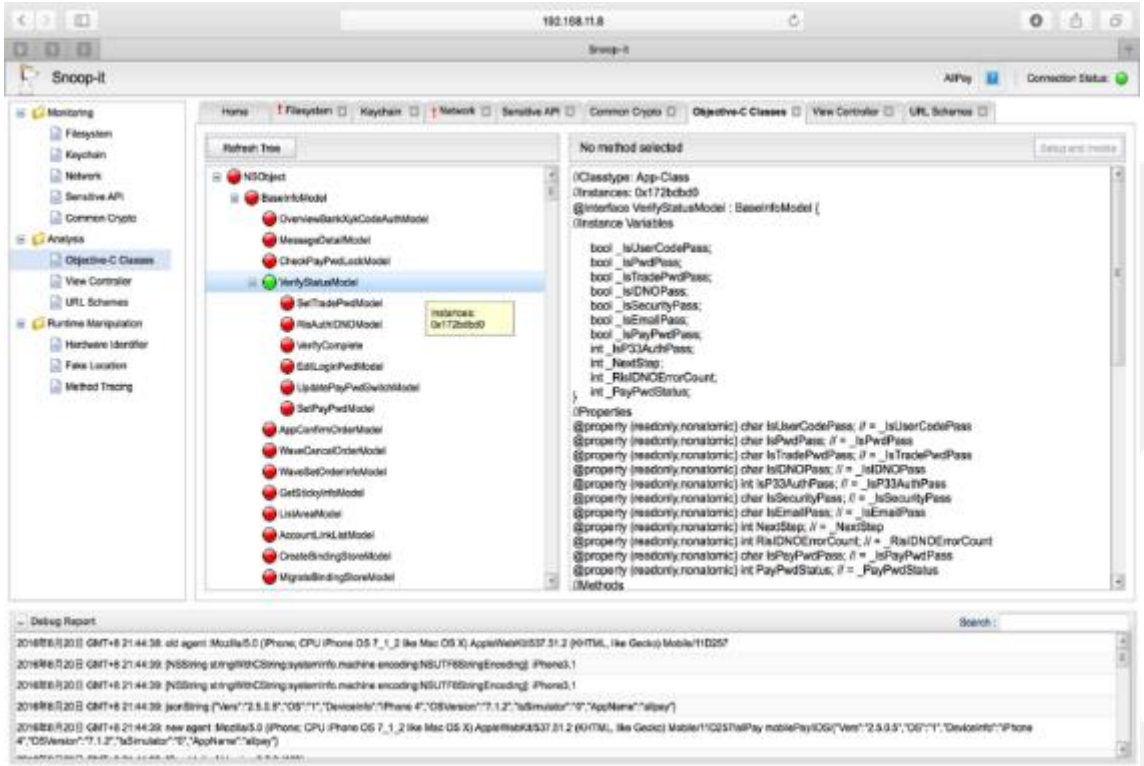
檢測操作步驟-動態分析(Snoop-It)



動態分析- Monitoring(Common Crypto)

步驟四(6/9)

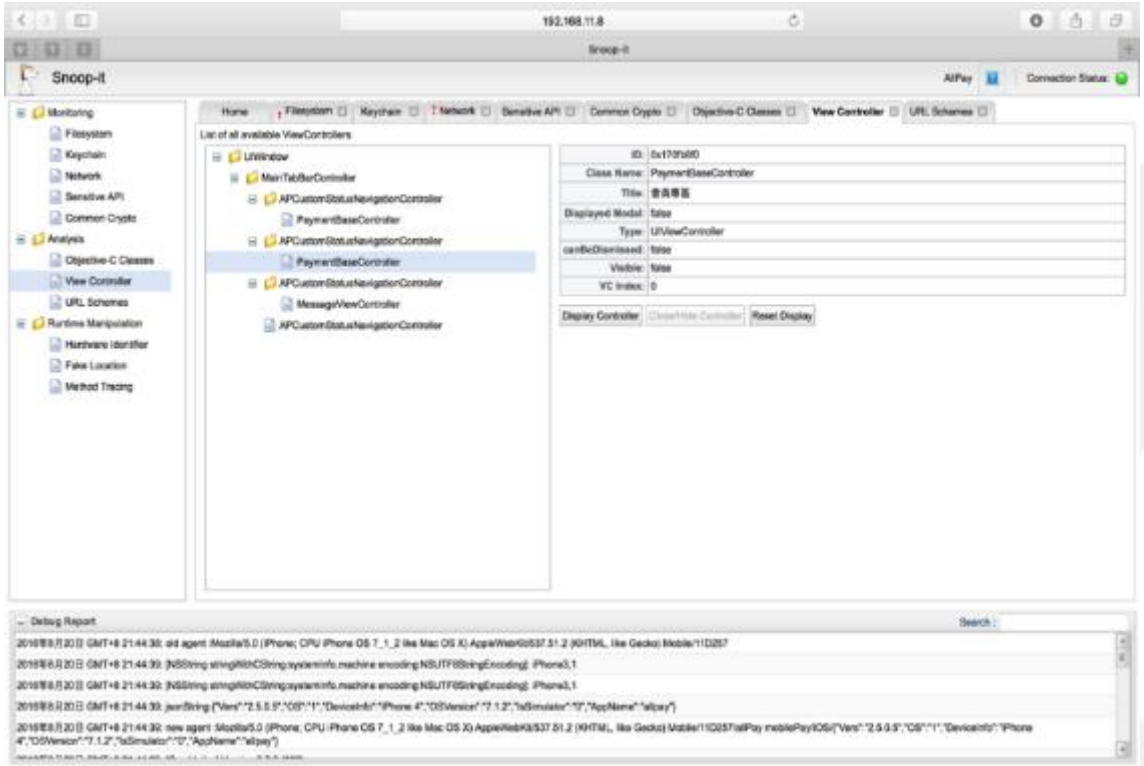
檢測操作步驟-動態分析(Snoop-It)



動態分析- Analysis(Object-C Classes)

步驟四(7/9)

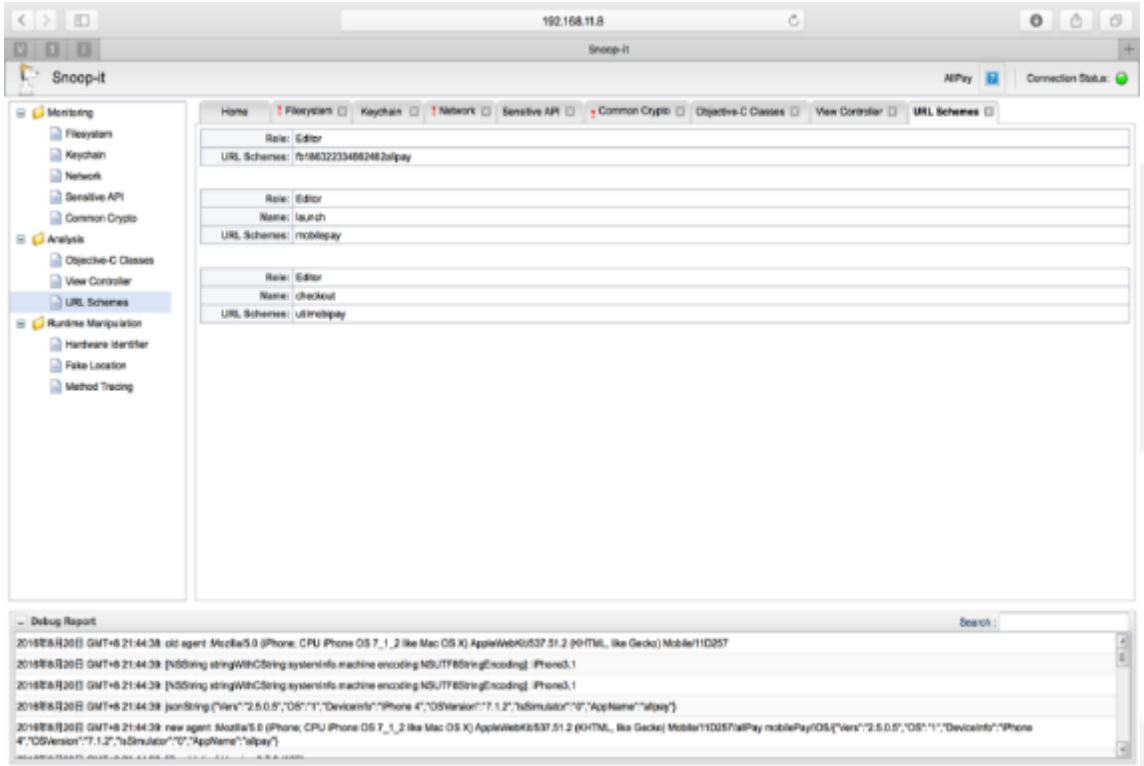
檢測操作步驟-動態分析(Snoop-It)



動態分析- Analysis(View Controller)

步驟四(8/9)

檢測操作步驟-動態分析(Snoop-It)



動態分析- Analysis(URL Schemes)

步驟四(9/9)

「行動應用App基本資安檢測基準」各構面與開發最佳實務工具對應

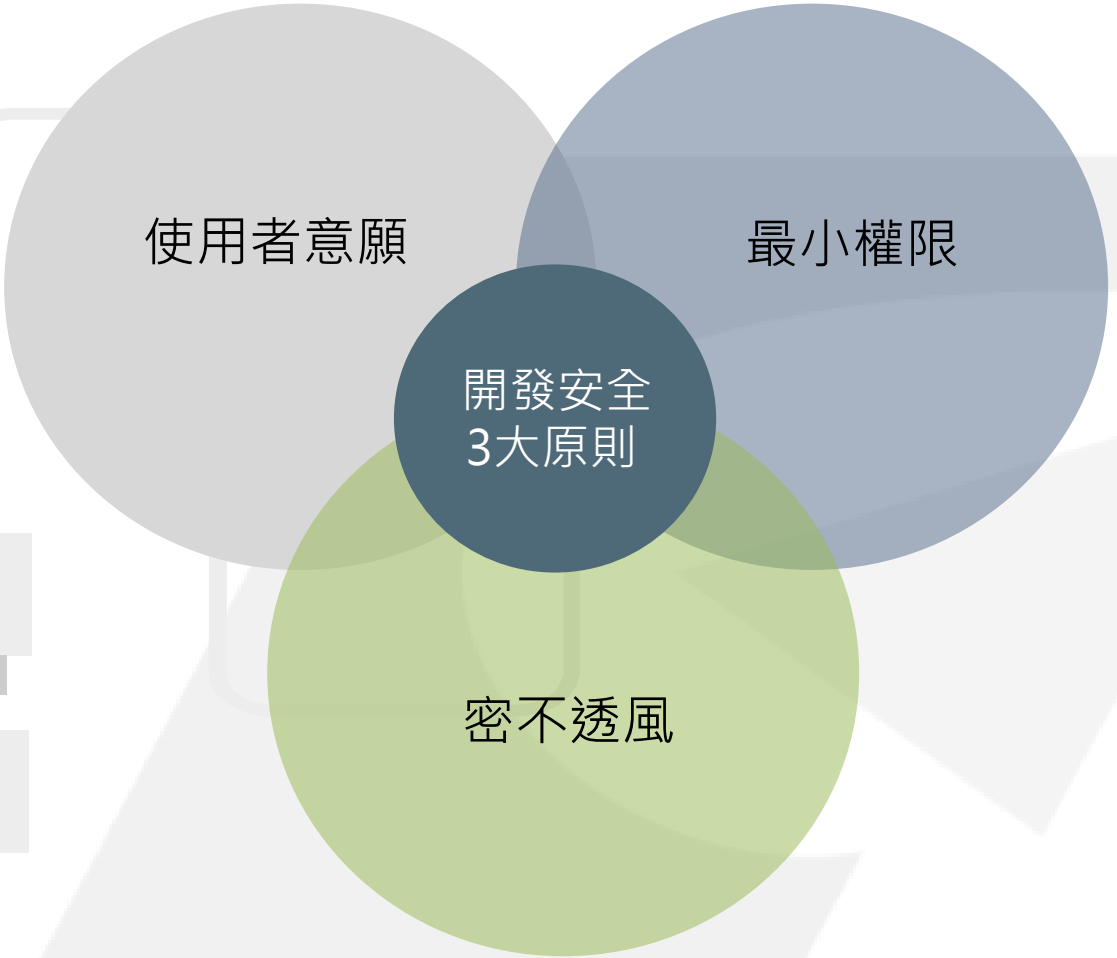
以工業局公告的「行動應用App基本資安檢測基準」與相關安全規範等之相關安全項目，對應本單元提出可應用之檢測工具，彙整成表，提供開發者參考。

檢測類別	檢測項目	基準規範	技術要求	可用檢測工具
	4.1.1.1. 行動App 發布	基本資安檢測基準	4.1.1.1.1 行動應用 App 應於可信任來源之行動 App 商店發布	文件檢核
		基本資安檢測基準	4.1.1.1.2 行動應用 App 應於發布時說明欲存取之敏感性資料、行動裝置資源及宣告之權限用途 (CPA-01)	文件檢核
		共通安全開發實務準則	CPA-01：於行動應用程式商店提供及應用程式內實作提供隱私權政策說明連結，說明欲存取之敏感性資料、行動裝置資源及宣告權限用途。	文件檢核
		共通安全開發實務準則	CPA-02：行動應用 App 實際權限與於行動平台商店提供及應用程式宣告終端使用者授權約定 (EULAs)、應用程式說明、程式內部通知及與 CPA-01 於欲存取之敏感性資料、行動智慧裝置資源及宣告權限用途一致。	文件檢核
	4.1.1. 行動App 發布安全	共通安全開發實務準則	CPA-03：應於行動應用 App 上架前確保內部軟體品質流程及版本控制均已實作完成。	文件檢核
		共通安全開發實務準則	CPA-04：確保應用程式規格遵循行動應用商店，如蘋果的 App Store 和 Google Play 規範的規	文件檢核
		共通安全開		

請參考「行動應用App安全開發指引」表20：建議工具與「行動應用App基本資安檢測基準」項目對應表。

行動應用App開發3大原則

需要有一個安全開發流程 - 安全軟體開發生命週期(SSDLC)，協助開發者藉由安全需求、安全設計、實作安全、測試安全及上線安全，開發出不會濫取使用者資料、能對抗惡意攻擊的行動應用App。



問題與討論

- 中華軟協 協同計畫主持人 陳亮宏 專案顧問
  wasavii@gmail.com
  專案顧問 蔡瑞雄
  rex.tsai@gmail.com

- 中華軟協 資安輔導組 張德維 組長
  (02)2553-3988#339
  david.chang@mail.cisanet.org.tw

- 中華軟協 資安輔導組 廖惠美 資深專員
  (02)2553-3988#626
  huimei@mail.cisanet.org.tw