

行動應用 App 資安標章全攻略

「需求、開發到實驗室檢測的合規一體化實務」



L1



L2



L3

執行單位：  台北市電腦公會

單元二 | 規範解析篇

行動應用 App 資安認驗證制度

Security by Design , 把規範化為開發控制點

建構「開發即合規」防護力

業界資深開發專家
郭凱迪 總經理

課程地圖



單元二 | 安全開發指引：建構「開發即合規」防護力

從源頭安全（Security by Design）出發，將抽象的資安規範轉化為開發製程中的具體控制點。

三單元課程地圖

單元一

制度推動篇

打造「全面性防禦」管理力

制度緣起、架構、標章與市場價值

單元二

本單元

規範解析篇

建構「開發即合規」防護力

Security by Design，把規範化為開發控制點

單元三

基準說明篇

落實「實驗室實戰」檢測實務

檢測項目、常見盲點與送驗前 Checklist



課程主軸 期

從「外部檢測」→ 到「開發即合規」

以主動防禦貫穿 App 研發全生命週

今天的三個部分

一條主線：把檢測基準，變成開發的日常

1

為什麼要「開發即合規」？

把資安項目併進開發流程——從源頭內建防護，而非事後補救

2

設計與開發的 Checklist

動工前把該注意的點一次盤點清楚，每一項列入專案流程

3

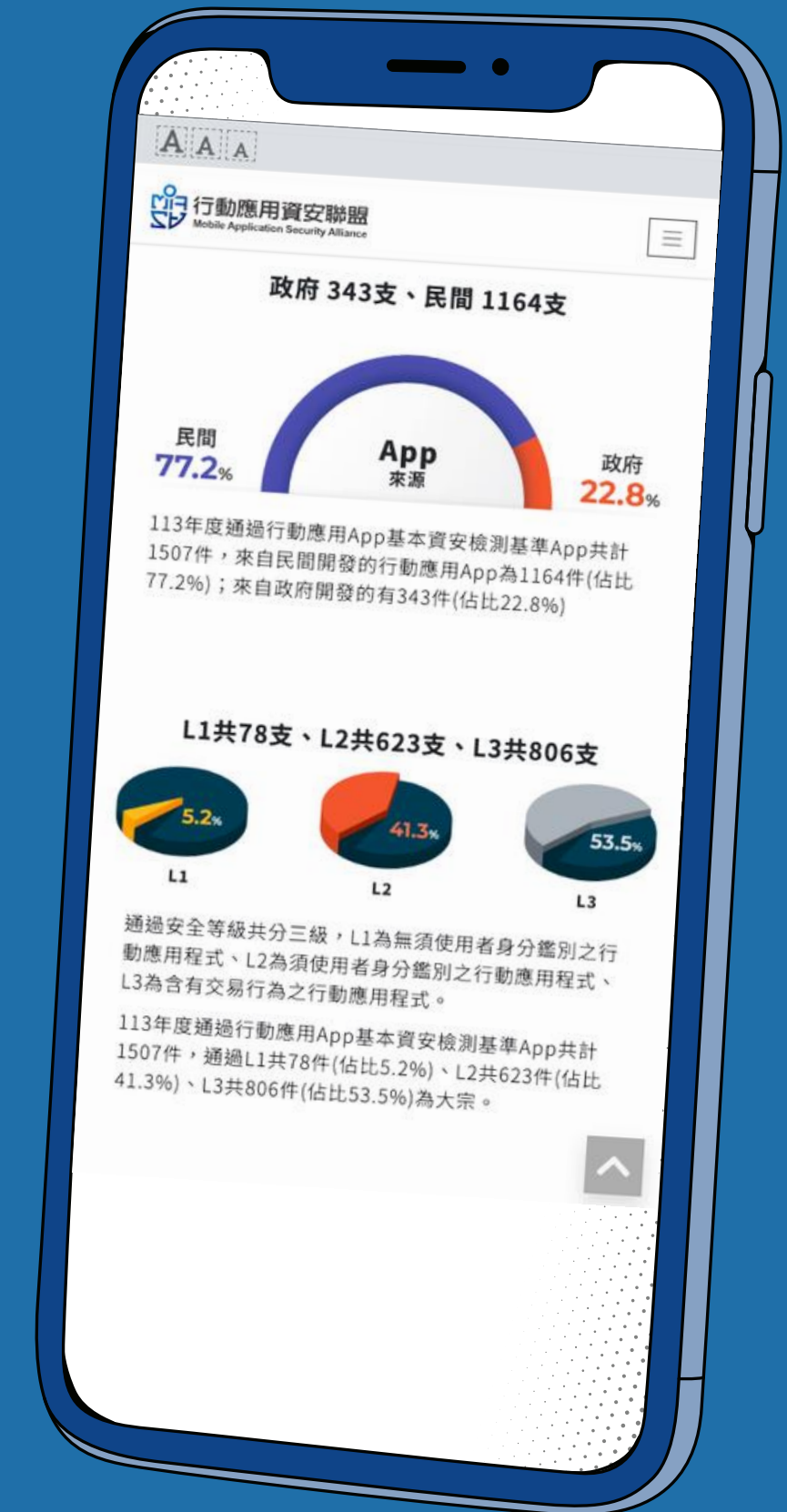
開發案例分享

分享最常見的地雷&實務案例

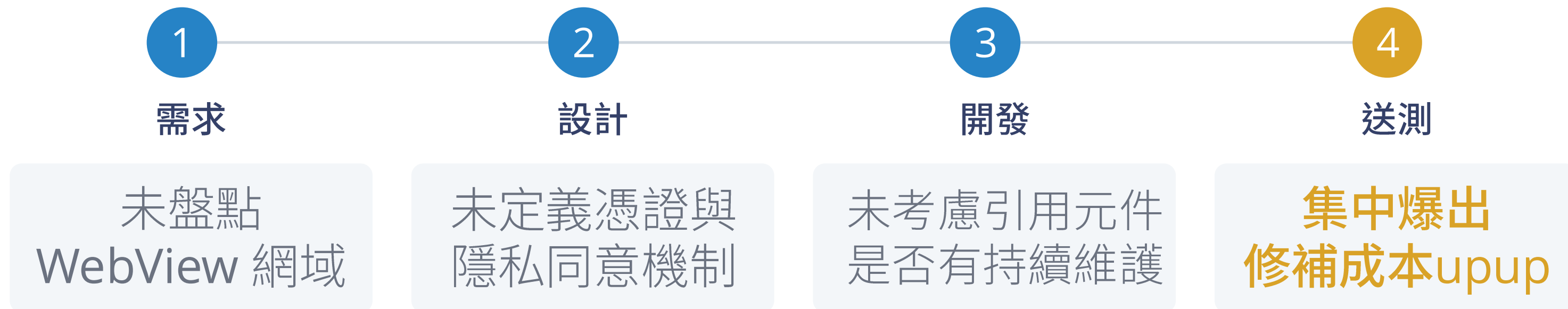
PART 1

為什麼要「開發即合規」？

把資安項目併進開發流程——從源頭內建防護，而非事後補救



檢測才發現的問題，通常在幾個月前就已經決定



問題被「發現」的位置在檢測，但問題「開始」的位置在需求與設計。

對應 V4.0：4.1.2.3.16 | 4.1.4.2.2 | 4.2.2.1.2

同一個問題，晚一步改，代價完全不同

需求階段改

改「一句話」

調整規格與資料清單，程式還沒動工

開發階段改

改「一段程式」

程式與測試同步修正，影響範圍可控

送測後改

改「一個版次」

API、App、後台、文件、測試全部牽動，還要重新送驗

越早決定，修改範圍越小；越晚發現，影響的是整個流程。

開發即合規=Shift Left

把資安控制點前移至開發製程——把基準當規格書，不是考卷

以前：外部檢測

送測前才面對基準，像交考卷，錯了扣分



現在：開發即合規

開案就把基準當規格書，照著做、留佐證

對開發商

降低事後修補成本，避免上架前得加班趕工



對需求者（發案單位）

看得懂 App 的安全底線，知道要在何時提出

「晚一天左移，上架前就多付出一天工時。」

把規範整合進專案開發階段

階段	資安控制點	V4.0 對應
需求	敏感性資料盤點、隱私權限告知同意	4.1.2.1、4.1.2.2
設計	加密選型、TLS、憑證策略、WebView 網域	4.1.2.4、4.1.4.2
開發	安全編碼、輸入驗證、套件盤點	4.1.5.3、4.1.5.4
測試	模擬測試、注入	4.2.2.1
上架	簽章驗證、發布安全	4.1.1
維運	CVE、套件版本、憑證到期	4.1.5.3

PART 2

設計與開發的 Checklist

動工前把該注意的點一次盤點清楚，每一項列入專案流程



需求訪談階段

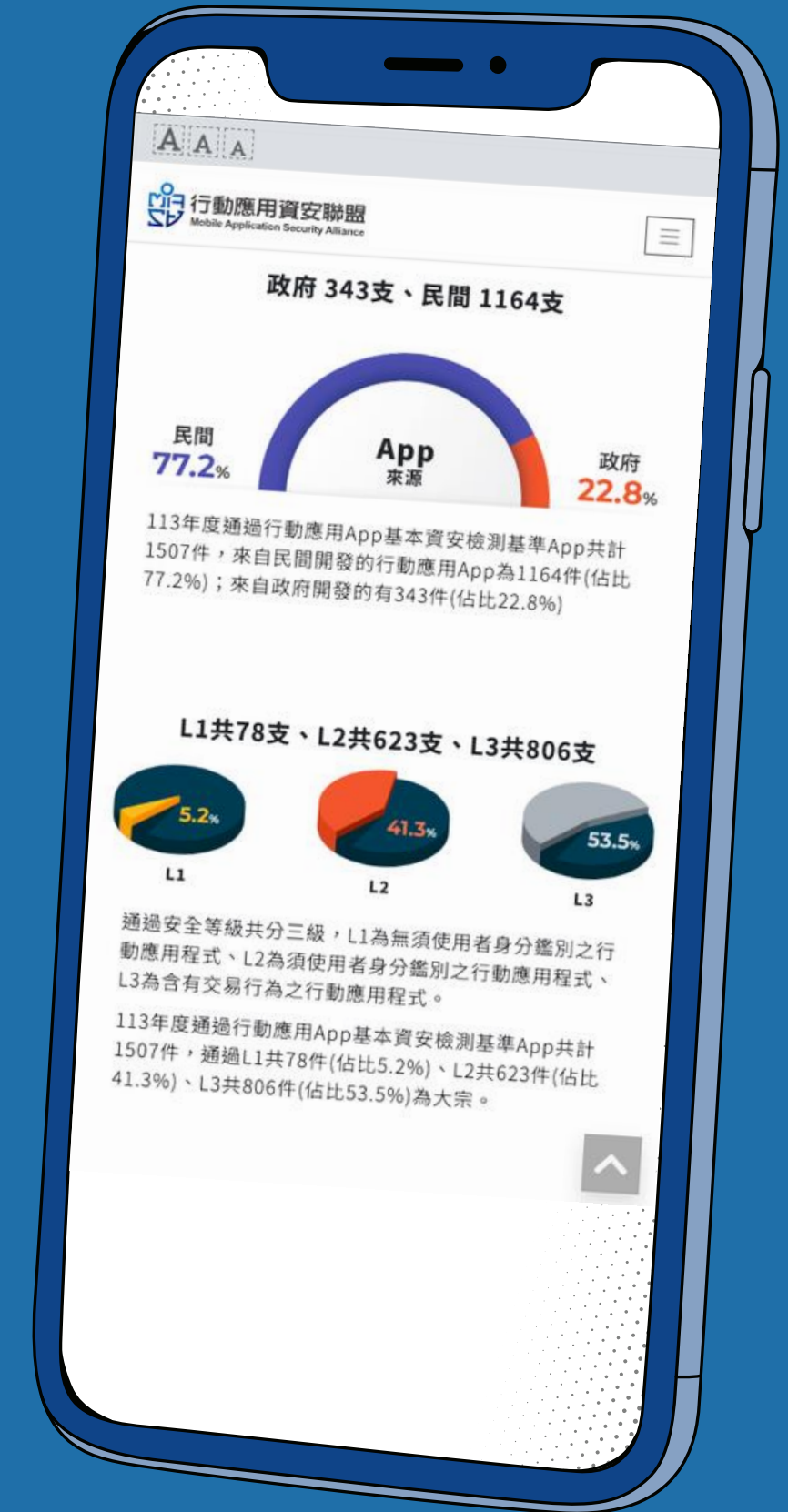
- 1 這支 App 會蒐集／儲存哪些敏感性資料？誰同意？
- 2 資料要不要分享給第三方？分享範圍如何界定？
- 3 伺服器憑證誰簽發？是否採用憑證綁定？
- 4 要用哪些第三方 SDK／函式庫？誰負責追蹤更新？
- 5 有沒有 WebView？連到哪些網域？網域安不安全？

開發階段 Checklist 範本

分類	開發前必須回答	應產出的設計	Check
資料與權限	收什麼、為何、誰能看、是否交易？	資料盤點、角色與資源	v
儲存與處理	是否存本機？Log／快取／備份？	儲存、金鑰、Log、備份策略	v
身分與連線	如何登入、授權、管理 Session 與憑證？	Session、憑證策略	v
套件與輸入	SDK 來源／版本？輸入在哪裡驗證？	套件清單、規格	v
WebView／發布	允許哪些網域？記得web要弱掃	網域清單、導向規則	v

PART 3 開發案例分享

分享最常見的類型&實務案例



開發最常遇到的 8 種類型

每個案例都看三件事：正確做法怎麼做、檢測看什麼證據、開發如何落地

- | | | | |
|----------|--------------------------------------|----------|-------------------------------------|
| 1 | 敏感性資料蒐集
同意前封包／拒絕流程 | 2 | 敏感性資料儲存
檔案／Log／備份 |
| 3 | 資料傳輸安全
HTTP／TLS／Redirect | 4 | 憑證驗證與連線加密
MITM／Pinning／網域 |
| 5 | 身分權限：交談識別／身分控管
Replay／越權測試 | 6 | 第三方套件管理
實際 Build／SBOM／CVE |
| 7 | 注入攻擊防護
異常輸入／編碼變形 | 8 | WebView 安全
網域清單／導向／掃描 |

案例 1 | 敏感性資料蒐集

V4.0 | 4.1.2.1

基準要求

- 蒐集前取得使用者同意
- 必須讓使用者能拒絕蒐集
- 資料項目、用途、權限說明與實際行為必須一致

實務做法

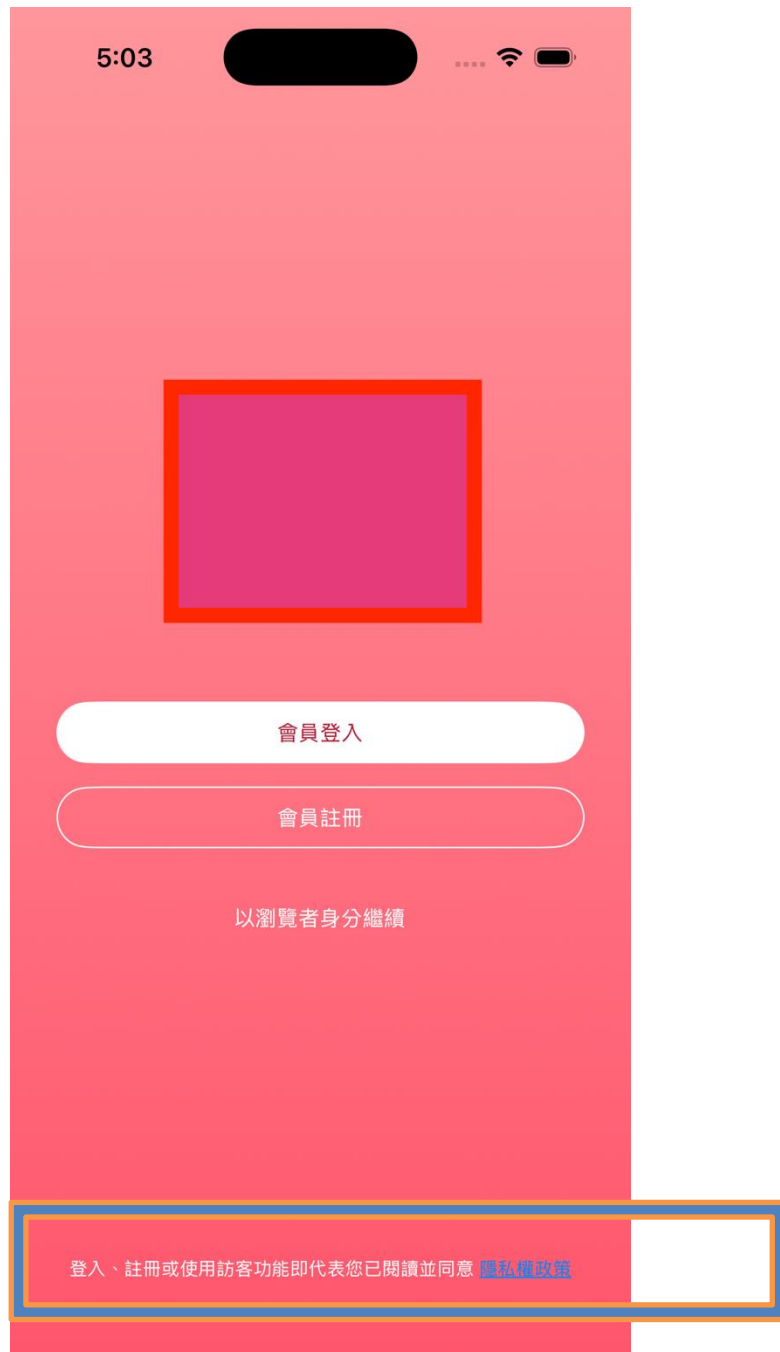
開發實作重點

敏感性資料若有蒐集需要，須在註冊與登入之前就顯示明顯文字提醒：

例如於登入或註冊按鈕下方註明，或在註冊的下一步驟顯示協議文案，並勾選同意後才能繼續往下一步驟。

依據：V4.0 4.1.2.1.1–4.1.2.1.2 | REF-6–7

案例 1 | 敏感性資料蒐集



案例 2 | 敏感性資料儲存

V4.0 | 4.1.2.3

基準要求

- 敏感資料須加密後儲存 (4.1.2.3.6)
- 存於 OS 保護區，避免其他 App 存取 (3.7)
- 不得出現在程式碼、備份、冗餘檔與系統 Log (3.8／3.14／3.16)

實務做法

開發實作重點

SessionID 與敏感個資若需儲存，必須加密儲存、取出時解密，並在登出時清除。
以儲存帳號為例有兩種做法：

一是記錄帳號本身，就必定得加密儲存，且登出時清除；

二是登入後由後台 API 將資料傳回 App，僅留在記憶體、不儲存。

依據：V4.0 4.1.2.3.6、3.7、3.8、3.14、3.16

案例 2 | 敏感性資料儲存

```
isRememberMe.toggle()  
checkbox.isSelected = isRememberMe  
prefs.set(isRememberMe, forKey: "rememberMe")  
  
let pcdKey = "pcd"  
  
if isRememberMe {  
    // 7) 儲存時：使用記憶體中的完整字串  
    let trimmed = currentTrimmedPCD()  
    let upperPCD = trimmed.uppercased()  
    guard !upperPCD.isEmpty else {  
        prefs.removeObject(forKey: pcdKey)  
        prefs.synchronize()  
        return  
    }  
    let encrypted = encryptText(upperPCD)  
    prefs.set(encrypted, forKey: pcdKey)  
    pcdBuffer.override(with: upperPCD)  
    updateTextFieldWithMaskedPCD()  
} else {  
    prefs.removeObject(forKey: pcdKey)  
}  
  
prefs.synchronize()
```

案例 2 | 敏感性資料儲存

```
/// 使用 FWEncryptorAES 以 Base64(AES) 加密 PCD (以大寫後再加密)
func encryptText(_ pcd: String, upText: Bool = true) -> String {
    let key = FWEncryptorAES.getOrCreateKey()
    let iv = Data()
    let upper = upText ? pcd.uppercased() : pcd
    if let encrypted = FWEncryptorAES.encryptStr(toBase64: upper, key: key, iv: iv) {
        return encrypted
    }
    // fallback: 回傳大寫原字串以避免崩潰
    return upper
}
```

案例 3 | 資料傳輸安全

V4.0 | 4.1.2.4.1

基準要求

- 所有網路傳輸都須採安全加密
- TLS 1.2 以上；禁用 3DES 與明文
- 演算法與金鑰長度必須符合基準

實務做法

開發實作重點

App 雖可經設定與非 HTTPS 後端連線，但有資安要求時必須：

1. 全程 TLS 1.2／1.3 加密，禁用 HTTP、SSL、TLS 1.0／1.1
2. 僅用 AES-256-GCM、ChaCha20-Poly1305 等現行演算法，禁用 3DES、DES、RC4、MD5、SHA-1
3. 金鑰交換採 ECDHE，支援 PFS
4. 憑證採 RSA 2048 或 ECC P-256 以上，由可信任 CA 簽發

依據：V4.0 4.1.2.4.1 | MASTG V1.7.0

案例 3 | 資料傳輸安全

```
private static func aesGcmEncrypt(_ plain: Data, key: Data) throws -> Data {
    let symmetricKey = SymmetricKey(data: normalizeKey(key))
    let sealed = try AES.GCM.seal(plain, using: symmetricKey)
    guard let combined = sealed.combined else {
        throw NSError(domain: "FWEncryptorAES", code: -1,
            userInfo: [NSLocalizedDescriptionKey: "AES.GCM combined output missing"])
    }
    return combined
}

private static func aesGcmDecrypt(_ combined: Data, key: Data) throws -> Data {
    let symmetricKey = SymmetricKey(data: normalizeKey(key))
    let sealedBox = try AES.GCM.SealedBox(combined: combined)
    return try AES.GCM.open(sealedBox, using: symmetricKey)
}
```

依據：V4.0 4.1.2.4.1 | MASTG V1.7.0

案例 4 | 憑證與連線驗證

V4.0 | 4.1.4.2

基準要求

- 驗證有效期、撤銷狀態與主機名稱（4.1.4.2.2）
- 驗證可信 CA；固定服務採憑證綁定（2.2／2.3）
- 實際封包流向須與宣告一致（2.4）

實務做法

開發實作重點

憑證需有效驗證：例如驗證伺服器憑證的有效期限，確認憑證未過期；

登出時也要清除本機儲存的認證資訊（如 Access Token、Session ID）。

更進階的做法如某些網銀 App：

即使已用 SSL 溝通，仍先下載站台 SSL 憑證，協議溝通時先比對憑證是否吻合，防止中間人攻擊。

依據：V4.0 4.1.4.2.2–4.1.4.2.4

案例 4 | 憑證與連線驗證

```
func webView(_ webView: WKWebView,
             didReceive challenge: URLAuthenticationChallenge,
             completionHandler: @escaping (URLSession.AuthChallengeDisposition, URLCredential?) -> Void) {
    let host = challenge.protectionSpace.host
    let method = challenge.protectionSpace.authenticationMethod
    Dlog("🔒 challenge host=\(host), method=\(method)")
    Dlog("web end --- \(Date().timeIntervalSince1970)")
    guard UATServerTrustEvaluator.canHandle(challenge),
          let trust = challenge.protectionSpace.serverTrust else {
        Dlog("🔒 challenge default handling for host=\(host)")
        completionHandler(.performDefaultHandling, nil)
        return
    }

    if UATServerTrustEvaluator.evaluate(challenge) {
        Dlog("✅ trust pass host=\(host)")
        completionHandler(.useCredential, URLCredential(trust: trust))
    } else {
        Dlog("❌ trust fail host=\(host)")
        completionHandler(.cancelAuthenticationChallenge, nil)
    }
}
```

案例 5 | 身分權限：交談識別與身分控管

V4.0 | 4.1.4.1–2

基準要求

- Session ID $\geq 128\text{bit}$ ，且不可由時間或資料推算（4.1.4.2.1）
- 登出後交談識別碼必須失效（2.1）
- 依身分與資源進行授權（4.1.4.1.1–1.2）

實務做法

開發實作重點

某些 App 會製作自動登入：

登入過後，下次打開 App 即自動登入。

最好的方式是只記錄 Access Token，跟後端 API 溝通後，由後端驗證 Token 是否有效，並回傳新的 Access Token，作為安全控管

若後端為客戶開發，記得事先提醒

依據：V4.0 4.1.4.2.1 | 4.1.4.1.1–4.1.4.1.2

案例 5 | 身分權限：交談識別與身分控管

```
func loginSuccess(data:JSON) {  
    hasLogin = true  
    token = data["access_token"].stringValue  
    let expires_in = data["expires_in"].intValue  
    expiredDate = Date().addingTimeInterval(TimeInterval(expires_in))  
    Dlog("token \ \(token), expiredDate =  
        \ \(expiredDate.dateAndTimeSectoString())")  
  
    UserDefaults.standard.set("1", forKey: "hasLogin")  
    UserDefaults.standard.set(token, forKey: "token")  
}
```

依據：V4.0 4.1.4.2.1 | 4.1.4.1.1–4.1.4.1.2

案例 6 | 第三方套件管理

V4.0 | 4.1.5.3.1

基準要求

- 函式庫有更新時，須備妥對應版本
- CVE 且 CVSS v3.0 ≥ 7 視為不符合
- 調查表需列明函式庫名稱、版本與來源

實務做法

開發實作重點

使用第三方套件，需使用已具知名度、廣被認可的套件，

並盡量採用仍在維護中的套件，避免時間久遠後，因手機系統資安要求提升而讓套件失效。

依據：V4.0 4.1.5.3.1

案例 6 | 第三方套件管理

```
pod 'SwiftyJSON', '~> 4.0'  
pod 'Alamofire', '~> 4.7'  
pod 'AlamofireImage'  
pod 'Kingfisher', '~> 4.8.1'  
pod 'iProgressHUD', '~> 1.1.1'  
pod 'Lightbox'  
pod 'SwiftAlertView', '~> 1.3.0'  
pod 'VersaPlayer'  
pod 'FMDB'  
pod 'Firebase/Core'  
pod 'Firebase/Messaging'  
pod 'FSPagerView'  
pod 'Parchment', '~> 3.2'  
pod 'CryptoSwift'  
pod 'ProgressHUD'  
pod 'Charts'
```

案例 6 | 第三方套件管理

APP版本與套件說明							
平台	APP名稱	開發語言	開發版本	第三方套件現行版本	說明	更新狀態	來源 網址
Android	XXXXX	java	Android studio XX	androidx.appcompat:appcompat:1.3.0-alpha01	Android相容套件	持續更新	Google https://developer.android.com/jetpack/androidx/releases/appcompat
				com.android.support.design:28.0.0	Android UI套件	持續更新	Google https://developer.android.com/topic/libraries/support-library/package
				com.google.android.gms:play-services-maps:17.0.0	Android Google 地圖套件	持續更新	Google https://developers.google.com/android/guides/releases
				com.google.code.gson:gson:2.8.6	Google JSON資料格式	持續更新	Google https://mvnrepository.com/artifact/com.google.code.gson/gson
				com.makeramen:roundedimageview:2.3.0	圓角圖像元件	持續更新	Github https://github.com/vinc3m1/RoundedImageView
				com.jakewharton:butterknife:10.2.1	開發輔助工具	持續更新	Github https://github.com/TakeWharton/butterknife
				com.github.bumptech.glide:glide:4.11.0	照片儲存元件	持續更新	Github https://github.com/bumptech/glide
				com.scottyab:rootbeer-lib:0.0.8	檢測Root狀態套件	持續更新	Github https://github.com/scottyab/rootbeer
				com.7heaven.ioswidget:iosswitch:0.6	仿iOS開關套件	持續更新	Github https://github.com/7heaven/SHSwitchView
iOS	XXXXX	Objective-C	Xcode XX	com.astuetz:pageslidingtabstrip:1.0.1	左右滑動播動元件	持續更新	Github https://github.com/astuetz/PagerSlidingTabStrip
				FMDatabase (2.7)	SQLite 資料庫操作元件	持續更新	github https://github.com/ccgus/fmdb
				MWPhotoViewer (2.1.2)	圖片瀏覽元件	持續更新	github https://github.com/mwaterfall/MWPhotoBrowser

案例 7 | 注入攻擊防護

V4.0 | 4.1.5.4

基準要求

- 檢核型別、長度、格式與邊界（4.1.5.4.1）
- 提供 7 類注入攻擊防護（4.1.5.4.2）
- 特殊字元要正確處理，不能只靠黑名單

實務做法

開發實作重點

表單、輸入框、搜尋框這類一般輸入欄位，多數都會記得做驗證；但 Deep Link 常被當成「自己人呼叫」而漏掉——它其實是什麼 App 都能發起的外部輸入，其驗證也要記得做：

參數一律檢核型別、長度、格式，並採白名單驗證，不符即拒絕

參數不可直接帶入 WebView、SQL 或指令執行，須先過濾或參數化處理

敏感操作由後端重新驗證身分與授權；

單號設有效期限或用過即失效，防止重放攻擊

依據：V4.0 4.1.5.4.1–4.1.5.4.2

案例 7 | 注入攻擊防護

```
func analyzeDeepLinkURL(Str str:String) {  
    if isInVIPVerify, vipVC != nil {  
        //showAlert(vipVC, title: "", msg: str)  
    }  
    Dlog("analyzeDeepLinkURL str = \(str)")  
    if str.hasPrefix("dpapp://mydataAuth") {  
        if isInVIPVerify, vipVC != nil {  
            let ary = str.components(separatedBy: "mydataAuth/")  
            if ary.count == 2 {  
                if ary[1] == "success" {  
                    vipVC.goFinishVC()  
                } else {  
                    vipVC.showFailStr(str: "MyData認證未成功")  
                }  
            }  
        }  
    }  
    return
```

依據：V4.0 4.1.5.4.1-4.1.5.4.2

案例 8 | WebView 安全

V4.0 | 4.2.2.1.2

基準要求

- WebView 實際連線網域都必須納入
- 非安全網域須完成 Injection Flaws 掃描
- 實際網域須與調查表宣告一致

實務做法

開發實作重點

WebView 使用時盡量設定不在 local 端儲存資料，甚至每次使用後清除 cache

若須以 JavaScript 在 WebView 與 App 間傳資料，應避開敏感個資，甚至加密後再傳輸。

更安全的做法如網銀 App：先下載站台憑證，溝通時先驗證是否吻合，防止中間人攻擊。

依據：V4.0 4.2.2.1.2

案例 8 | WebView 安全

```
private func performLogoutAndResetWebSession() {
    isAuthenticated = false
    SensitiveStore.appToken = nil
    AppDelegate.shared().jwt = ""
    prefs.synchronize()
    applIdBuffer.clear()

    // 2) Stop the current webView and detach delegates/handlers
    tearDownCurrentWebView()

    // 3) Fully reset WebKit session (cookies/cache/storage) + process pool, then rebuild webView
    clearAllWebData { [weak self] in
        guard let self = self else { return }
        DispatchQueue.main.async {
            self.clearURLSessionCacheAndCookies()
            self.removeLegacyCacheDBFiles()

            // New process pool = new web content process (prevents stale session state)
            self.webProcessPool = WKProcessPool()
            self.initWebView()
            self.goToLogin()
        }
    }
}
```

'WKProcessPool' was deprecated in iOS

案例 9 | 懸浮遮罩/蓋板處理

V4.0 | 4.1.5.1.3

基準要求

- 敏感畫面避免被遮蔽、誘導點擊或誤觸
- 登入、交易、OTP、授權確認須降低外洩風險
- 外部輸入與觸控事件不得直接信任

實務做法

開發實作重點

常見情境：惡意 App 開啟懸浮遮罩或透明蓋板，遮住 OTP、付款確認或授權按鈕，誘導使用者誤點。

開發者怎麼改：

1. 敏感流程偵測遮蔽觸控，必要時阻擋操作
2. 登入／交易／OTP 頁顯示警示，要求關閉懸浮視窗
3. Android 可搭配 `filterTouchesWhenObscured/onFilterTouchEventForSecurity`。

案例 9 | 懸浮遮罩/蓋板處理

```
override fun onFilterTouchEventForSecurity(ev: MotionEvent): Boolean {  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {  
        return true  
    }  
    val checkIsObscured = ev.flags and MotionEvent.FLAG_WINDOW_IS_OBSCURED != 0  
    val checkIsPartiallyObscured = ev.flags and MotionEvent.FLAG_WINDOW_IS_PARTIALLY_OBSCURED != 0  
  
    if (checkIsObscured || checkIsPartiallyObscured) {  
        Log.d("ObscureCheck", "Obscured: $checkIsObscured, Partially: $checkIsPartiallyObscured")  
  
        AlertDialog.Builder(context)  
            .setTitle("安全警告")  
            .setMessage("偵測到畫面有異常遮罩，為了安全已暫停此操作。")  
            .setPositiveButton("確定") { dialog, _ -> dialog.dismiss() }  
            .show()  
    }  
  
    return super.onFilterTouchEventForSecurity(event = ev)  
}
```

案例 9 | 懸浮遮罩/蓋板處理



案例 10 | 退到背景避免敏感資料任務快照被截取

V4.0 | V4.0 4.1.2.3.13 (L2/L3) 、4.1.2.3.9 (L3)

基準要求

- 敏感資料顯示時須避免被截圖、快照或外洩
- App 切到背景時，系統可能留下最近任務快照
- 回前景時須正確移除遮罩，不影響正常操作

實務做法

開發實作重點

常見情境：使用者停在餘額、OTP、身分證或病歷畫面，切到背景後，App Switcher／最近任務仍看得到敏感資訊。

開發者怎麼改：

1. App 進背景前加全黑或品牌遮罩
2. 回前景後再移除遮罩
3. Android 可搭配 FLAG_SECURE／最近任務截圖控管；iOS 可在 sceneWillResignActive 加 Privacy View。

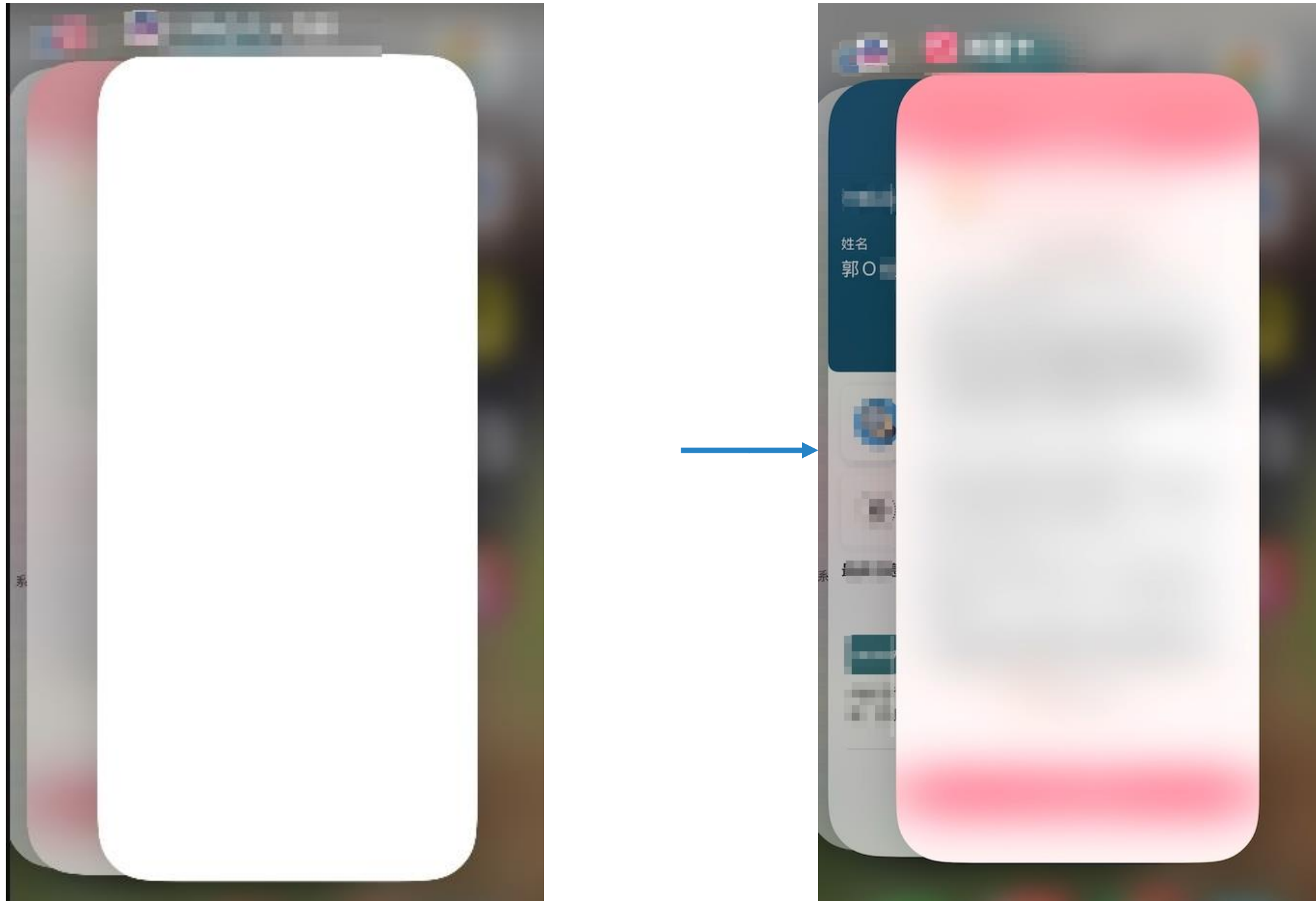
測試佐證：切背景錄影、App Switcher 截圖全黑、回前景流程正常。

依據：V4.0 4.1.2.3；實務延伸：背景快照與截圖保護

案例 10 | 退到背景避免敏感資料任務快照被截取

```
class SceneDelegate: UIResponder, UIWindowSceneDelegate {  
      
    func sceneWillResignActive(_ scene: UIScene) {  
        // Called when the scene will move from an active state to an inactive state.  
        // This may occur due to temporary interruptions (ex. an incoming phone call).  
        AppDelegate.shared().showBlackWindow()  
    }  
}
```

案例 10 | 退到背景避免敏感資料任務快照被截取



第二單元回顧

- 1 讓資安不是上架前的驗收關卡，而是規格書的一部分。
- 2 V4.0 的每一條，列入專案的各項流程。
- 3 團隊 Checklist，就是「開發即合規」的第一步。

「Security by Design，把規範化為開發控制點。」

十年淬一盾

打造「全面性防禦」管理力

讓每一個 App 標章，都值得被信任

感謝聆聽，敬請指教

行動應用資安聯盟

mas.org.tw



十年淬鍊．全面守護．值得信任