CPF-07:敏感性資料或包含敏感性資料的日誌檔，除非已加密，應避免儲存於與其他行動應用 App 共用或全域可讀寫儲存區域或外部儲存媒體。

不安全程式碼範例(Android：Java)

- *在 API Level 17 Android 4.2 (JELLY_BEAN_MR1)之前的版本，可以允許開發人員的 APP 將資料輸出到 APP 公有區中，下列範例即為將資料輸出到公有區中，並設其權限為共同寫入。*

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_WORLD_READABLE, );
fos.write(string.getBytes());
fos.close();
```

安全程式碼範例(Android：Java)

- *下列範例為安全的範列，此範例改將資料輸出到私有區中，並設其權限為私有，以進一步確保資訊安全。*

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

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。

安全程式碼範例(Android：Java)

```
//定義使用 ACCOUNT_MANAGER 與 INTERNET 權限
<manifest ... >
    <uses-permission android:name="android.permission.ACCOUNT_MANAGER" />
    <uses-permission android:name="android.permission.INTERNET" />
    ...
</manifest>
//實作 CallBack
AccountManager am = AccountManager.get(this);
Bundle options = new Bundle();
am.getAuthToken(
    myAccount_,                     // Account retrieved using getAccountsByType()
    "Manage your tasks",            // Auth scope
    options,                        // Authenticator-specific options
    this,                           // Your activity
    new OnTokenAcquired(),          // Callback called when a token is successfully acquired
    new Handler(new OnError()));    // Callback called if an error occurs

private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
    @Override
    public void run(AccountManagerFuture<Bundle> result) {
        ...
        Intent launch = (Intent) result.getResult().get(AccountManager.KEY_INTENT);
        if (launch != null) {
            startActivityForResult(launch, 0);
            return;
        }
    }
}
//獲取 token
private class OnTokenAcquired implements AccountManagerCallback<Bundle> {
    @Override
    public void run(AccountManagerFuture<Bundle> result) {
        // Get the result of the operation from the AccountManagerFuture.
        Bundle bundle = result.getResult();

        // The token is a named value in the bundle. The name of the value
        // is stored in the constant AccountManager.KEY_AUTHTOKEN.
        token = bundle.getString(AccountManager.KEY_AUTHTOKEN);
        ...
    }
}
//呼叫 OAuth2 服務
URL url = new URL("https://www.googleapis.com/tasks/v1/users/@me/lists?key=" +
your_api_key);
URLConnection conn = (HttpURLConnection) url.openConnection();
conn.addRequestProperty("client_id", your client id);
conn.addRequestProperty("client_secret", your client secret);
conn.setRequestProperty("Authorization", "OAuth " + token);
```

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。

安全程式碼範例(Android：Java)

可以在 Activity 內加上下面二個方法來達成 local-session-timeout

```
@Override public boolean dispatchTouchEvent(MotionEvent ev) {
        lastActivity = new Date().getTime();  //取得最後動作的時間
        return super.dispatchTouchEvent(ev);
}
@Override public void onResume() {
        long now = new Date().getTime();
        if (now - lastActivity > 300000) {     //超過五分鐘則自動登出
        // startActivity and force logon     } }
```

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。

安全程式碼範例(Android：Java)

假設你有由知名 CA 頒發的證書的 Web 伺服器，你可以用簡易程式碼安全要求：

```
URL url = new URL ( "https://wikipedia.org" );
URLConnection urlConnection = url . openConnection ();
InputStream in = urlConnection . getInputStream ();
copyInputStreamToOutputStream ( in , System . out );
```

如果你想客製設定 HTTP 請求，可以改使用 HttpsURLConnection 或是強制轉型為 HttpURLConnection

```
public class KeyPinStore {

    private static KeyPinStore instance = null;
    private SSLContext sslContext = SSLContext.getInstance("TLS");

    public static synchronized KeyPinStore getInstance() throws CertificateException,
IOException, KeyStoreException, NoSuchAlgorithmException, KeyManagementException{
        if (instance == null){
            instance = new KeyPinStore();
        }
        return instance;
    }

    private KeyPinStore() throws CertificateException, IOException, KeyStoreException,
NoSuchAlgorithmException, KeyManagementException{
        // Load CAs from an InputStream
        // (could be from a resource or ByteArrayInputStream or ...)
        CertificateFactory cf = CertificateFactory.getInstance("X.509");
        // randomCA.crt should be in the Assets directory
        InputStream caInput = new
BufferedInputStream(MainActivity.context.getAssets().open("randomCA.crt"));
        Certificate ca;
        try {
            ca = cf.generateCertificate(caInput);
            System.out.println("ca=" + ((X509Certificate) ca).getSubjectDN());
        } finally {
            caInput.close();
        }

        // Create a KeyStore containing our trusted CAs
        String keyStoreType = KeyStore.getDefaultType();
        KeyStore keyStore = KeyStore.getInstance(keyStoreType);
        keyStore.load(null, null);
        keyStore.setCertificateEntry("ca", ca);

        // Create a TrustManager that trusts the CAs in our KeyStore
        String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
```

```
        TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
        tmf.init(keyStore);

        // Create an SSLContext that uses our TrustManager
        // SSLContext context = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
    }

    public SSLContext getContext(){
        return sslContext;
    }
}
```

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。

不安全程式碼範例(Android：Java)

SQL 語法直接使用+的法式將使用者的輸入串連起來

```
tring commandString = "INSERT INTO mytable (idno, name) VALUES
'"+userInputIdno+"','"+userInputName+"'";
try {
    mSampleDB.execSQL(commandString);
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

安全程式碼範例(Android：Java)

使用參數化方式避免使用者的惡意輸入

```
String commandString = "INSERT INTO mytable (idno, name) VALUES (?, ?)";
//使用參數化的SQLiteStatement以防止SQL Injection
SQLiteStatement sqlStmt = mSampleDB.compileStatement(commandString);
sqlStmt.bindString(1, userInputIdno);
sqlStmt.bindString(2, userInputName);
try {
    sqlStmt.executeInsert();
}catch (SQLException e) {
    ......
} finally {
    sqlStmt.close();
}
```

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。

不安全程式碼範例(網頁主機：Server .NET C#)

使用字串加減維護 XML

```
xmlstr = "<username>" + "foo<" + "</username>";
```

安全程式碼範例(Android；Java)

```
libxml_disable_entity_loader(true);
```

安全程式碼範例(網頁主機：Server .NET C#)

使用元件維護 XML

```
XmlNode node = doc.CreateNode("username");
node.Value = "foo<";
```

Android-01:Android 版本行動應用 App 應謹慎實作 Intents，避免資訊不慎外洩遭惡意運用。

不安全程式碼範例(Java)

使用不指定接收 Class 的 Intent 來傳遞機密資訊，是不安全的也可能會被攔截，而使得資料外洩。

```
Intent intent = new Intent();
intent.putExtra("RESULT", "Sensitive Info");
startActivity(intent);
```

安全程式碼範例(Java)

當使用 Intent 來傳遞資料給 Activity, 為維持資料安全，請使用明文指定接收 Class 的名稱。

```
//明文指定接收此 intent 的類別，以避免其它程式攔截取得資訊，或改變程式流程
Intent intent = new Intent(this, LoginActivity.class);
intent.putExtra("RESULT", "Not Sensitive Info");
startActivity(intent);
```

Android-09: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 Intent Injection 之字串。

不安全程式碼範例(Java)

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    isr = new InputStreamReader(url.openConnection().getInputStream());
      //此處並無防止"file://..." 這種型式的。在此情形之下，本機端的檔案會被開啟，並且將內容秀在 TextView 中，而非原先預請的遠端網頁

    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ...
```

安全程式碼範例(Java)

```
TextView tv = (TextView) findViewById(R.id.textview);
InputStreamReader isr = null;
char[] text = new char[1024];
int read;
try {
    String urlstr = getIntent().getStringExtra("WEBPAGE_URL");
    URL url = new URL(urlstr);
    //加入明確檢查是否是使用 http 或 https protocol, 如不是則產生例外
    String prot = url.getProtocol();
    if (!"http".equals(prot) && !"https".equals(prot)) {
        throw new MalformedURLException("invalid protocol");
    }
    isr = new InputStreamReader(url.openConnection().getInputStream());
    while ((read=isr.read(text)) != -1) {
        tv.append(new String(text, 0, read));
    }
} catch (MalformedURLException e) { ..
```

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。

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
```

安全程式碼範例(網站主機：Server .NET)

啟用 HSTS 機制

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <system.webServer>
        <rewrite>
            <rules>
                <rule name="HTTP to HTTPS redirect" stopProcessing="true">
                    <match url="(.*)" />
                    <conditions>
                        <add input="{HTTPS}" pattern="off" ignoreCase="true" />
                    </conditions>
                    <action type="Redirect" url="https://{HTTP_HOST}/{R:1}"
                        redirectType="Permanent" />
                </rule>
            </rules>
            <outboundRules>
                <rule name="Add Strict-Transport-Security when HTTPS" enabled="true">
                    <match serverVariable="RESPONSE_Strict_Transport_Security"
                        pattern=".*" />
                    <conditions>
                        <add input="{HTTPS}" pattern="on" ignoreCase="true" />
                    </conditions>
                    <action type="Rewrite" value="max-age=31536000" />
                </rule>
            </outboundRules>
        </rewrite>
    </system.webServer>
</configuration>
```