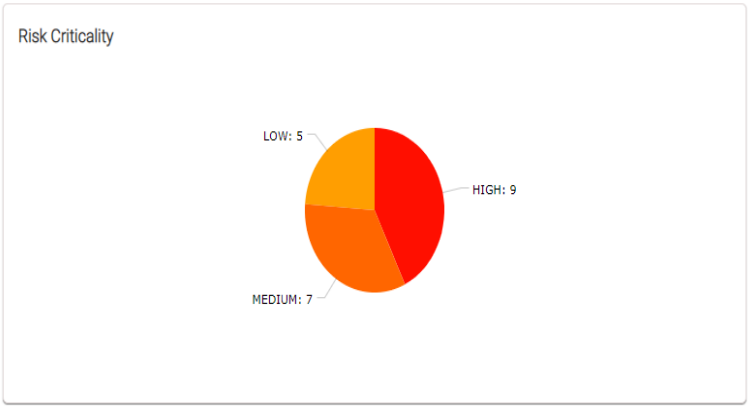
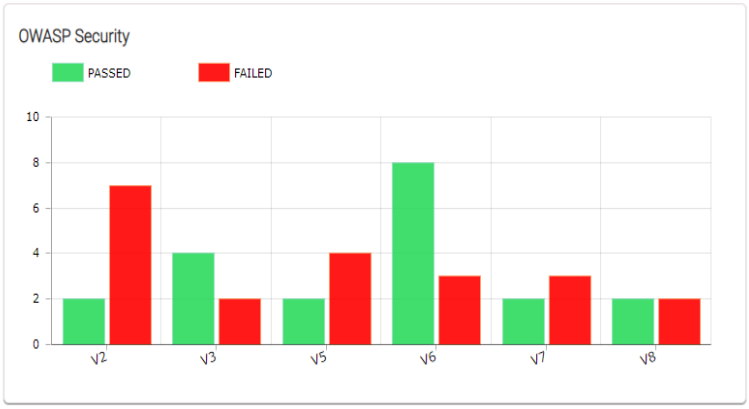
Meera Rewards

Mobile App Security Assessment - Android





Application Name	Platform	Package Name
AI Meera	Android	googlecom.codelab.meera
Package Version	**Total Vulnerabilities Detected**	
7.0	21	
High Severity Threats	**Medium Severity Threats**	**Low Severity Threats**
9	7	5

VULNERABILITY SCAN - FAILED SCENARIOS

Severity
High

VULNERABILITY

Improper Export of your Android Services

OWASP MASVS	Common Weakness Enumeration
6.2 L2	CWE-926
Known Exploits	**Common Vulnerability Scoring System**
CAPEC-501	

Reference URL[s]:

https://developer.android.com/guide/components/services
https://developer.android.com/about/versions/oreo/background.html#services
https://developer.android.com/guide/background
https://developer.android.com/guide/topics/permissions/defining
https://developer.android.com/guide/topics/manifest/permission-element
https://cwe.mitre.org/data/definitions/926.html
https://developer.android.com/training/articles/security-tips

CVSS BaseScore and Vector

CAPEC-501

THREAT

An Android Service is an application Component that can perform long-running operations in the background, and it doesn't provide a user interface. Another application Component can start a Service, and it continues to run in the background even if the user switches to another application. Additionally, a Component can bind to a Service to interact with it and even perform interprocess communication [IPC]. For example, a Service can handle network transactions, play music, perform file I/O or interact with a content provider, all from the background

RISK

If the access to a sensitive Service is misconfigured or not restricted then any application may start and bind to it. Moreover the user may not be aware that a Service was started by an unauthorized third-party

Depending on the exposed functionality, this may allow a malicious application to perform unauthorized actions, gain access to sensitive information or corrupt the internal state of the application

FIX

There are a few solutions that help mitigate the issue: by default, Android Services are not exported and cannot be invoked by any other application. Therefore, if you don't want other applications to invoke the Service or interact with it please delete any Intent Filters because their declaration is enough to turn that Service into an exported one do not assume the input is from a trusted source and validate each time to avoid your app to crash if you need to restrict the access [start, stop, bind] to your Service to specific callers then use an appropriate 'android:permission'. For example, consider using the signature 'android:protectionLevel' on permissions for IPC communication between applications provided by a single developer. Since Android 5.0 and superior, when naming a custom <permission> please bear in mind that the system does not permit the user to install other packages with the same permission name, unless those packages are signed with the same certificate as the first package. To avoid naming collisions, it is recommended to use reverse-domain-style naming for custom permissions, for example com.example.myapp.MY_PERMISSION

The following Service is not protected and export flag is set(android:exported=true).

```
<service android:name="com.google.firebase.messaging.FirebaseMessagingService" android:exported="true">
        <intent-filter android:priority="-500">
            <action android:name="com.google.firebase.MESSAGING_EVENT"/>
        </intent-filter>
    </service>
<service android:name="com.google.firebase.iid.FirebaseInstanceIdService" android:exported="true">
        <intent-filter android:priority="-500">
            <action android:name="com.google.firebase.INSTANCE_ID_EVENT"/>
        </intent-filter>
    </service>
```

Severity
High

VULNERABILITY

Improper Export of your Android Broadcast Receiver

OWASP MASVS

6.2 L2

Common Weakness Enumeration

CWE-927

CWE-925

Known Exploits

CAPEC-499

CAPEC-501

Common Vulnerability Scoring System

Reference URL[s]:

https://developer.android.com/guide/topics/manifest/receiver-element.html#exported
http://blog.palominolabs.com/2013/05/13/android-security/index.html
https://stackoverflow.com/a/38376435
https://developer.android.com/guide/topics/permissions/defining
https://developer.android.com/guide/topics/manifest/permission-element
https://cwe.mitre.org/data/definitions/926.html
https://developer.android.com/training/articles/security-tips

CVSS BaseScore and Vector

CAPEC-499

CAPEC-501

THREAT

Broadcast Receivers enable applications to receive Intents that are broadcast by the system or by other applications, even when other Components of the application are not running. Many Android applications use them to receive Intents without properly verifying if coming from an authorized source

Explicit Intents explicitly name the class of the target Android Component that will handle the Intent. Implicit Intents are used without a class name and in this case Android will help determine an appropriate Component to handle the Intent

RISK

A major problem with respect to intent spoofing is that registering an Intent Filter to receive implicit Intents makes that Component exported by default. This default "exported" behavior allows all the applications to send both implicit and explicit Intents to that Component

And for an exported=true Component if the Intents are explicit then they don’t even have to match the Intent Filter. So - ironically - creating an Intent Filter for a Component greatly widens the scope of Intents that Android will allow to be sent to it. Developers must not rely on Intent Filters for security

FIX

There are a few solutions that help mitigate the issue: if you don't want the Broadcast Receiver to receive messages from sources outside its application please delete any Intent Filters because their declaration is enough to turn that Broadcast Receiver into exported by default, even if you explicitly declared it as android:exported=false the current Android behavior is that a Broadcast Receiver without Intent Filters is not exported but if you want you can also explicit this adding android:export="false" do not assume the input is from a trusted source and validate each time to avoid your app to crash if you need to restrict the access to your Broadcast Receiver to specific callers then use an appropriate 'android:permission'. For example, consider using the signature 'android:protectionLevel' on permissions for IPC communication between applications provided by a single developer. Since Android 5.0 and superior, when naming a custom <permission> please bear in mind that the system does not permit the user to install other packages with the same permission name, unless those packages are signed with the same certificate as the first package. To avoid naming collisions, it is recommended to use reverse-domain-style naming for custom permissions, for example com.example.myapp.MY_PERMISSION

The following Broadcast Receiver is not protected. An intent-filter exists.

```
<receiver android:name="googlecom.codelab.meera.DownloadBroadcastReceiver">
        <intent-filter>
            <action android:name="android.intent.action.DOWNLOAD_COMPLETE"/>
        </intent-filter>
    </receiver>
```

The following Broadcast Receiver protected by permission but the protection level of the permission should be checked.The permission contains Google's package. We advise to check the permissions granted in the partner app/library and ensure that it is implemented correctly in the scanned app. Permission: com.google.android.c2dm.permission.SEND and exported flag is set(android:exported=true).

```
<receiver android:name="com.google.firebase.iid.FirebaseInstanceIdReceiver" android:permission="com.google.android.c2dm.permission.SEND
" android:exported="true">
        <intent-filter>
            <action android:name="com.google.android.c2dm.intent.RECEIVE"/>
            <category android:name="googlecom.codelab.meera"/>
        </intent-filter>
    </receiver>
```

Severity
High

VULNERABILITY

Clear text traffic is allowed in application

OWASP MASVS	Common Weakness Enumeration
-	CWE-319
Known Exploits	**Common Vulnerability Scoring System**
-	-

Reference URL[s]:

https://mas.owasp.org/MASTG/Android/0x05g-Testing-Network-Communication/#android-network-security-configuration

CVSS BaseScore and Vector

-

THREAT

Sending sensitive information in clear text traffic may help the attacker to capture the information, It is disabled by default in latest versions of android but still there are many ways to enable it.

It can be enabled by setting the value to true for the attribute android:usesCleartextTraffic="true" in application tag of AndroidManifest.xml

It can also be enabled by setting the attribute cleartextTrafficPermitted to true in domain-config or base-config tag of network_security_config.xml

RISK

Sensiitive information in clear text can be capture by attacker and it can be misused. It is easy for the attackers to perform MITM attack

FIX

It can be fixed by verifying following two configurations.
1. Ensure the attribute android:usesCleartextTraffic is set to "false" in AndroidManifest.xml
2. Ensure the attribute cleartextTrafficPermitted to false in domain-config and base-config if it is not necessary for the domain.

Clear text traffic is Enabled For App [android:usesCleartextTraffic=true]

<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/meera_app_icon" android:name="googlecom.codelab.meera.utils.MyApplication" android:allowBackup="true" android:supportsRtl="true" android:usesCleartextTraffic="true" android:roundIcon="@mipmap/meera_app_icon" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:requestLegacyExternalStorage="true">

Severity
High

VULNERABILITY

Cleartext Storage of Sensitive Information in app source code

OWASP MASVS

2.14 L2

Common Weakness Enumeration

CWE-312

Known Exploits

CVE-2018-19981

Common Vulnerability Scoring System

CVE-2018-19981-CVSS 3.0 Score 7.2

Best Practices:

https://aws-amplify.github.io/aws-sdk-android/docs/reference/com/amazonaws/auth/CognitoCachingCredentialsProvider.html

CVSS BaseScore and Vector

CVE-2018-19981
Version & Base Score : CVSS 3.0 Score 7.2
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

THREAT

App source may contain hardcoded sensitive information such as Oauth tokens, API keys, passwords any known intellectual property that maybe considered sensitive.

RISK

Sometimes mobile application developers leave sensitive data in the mobile app source code. They are not the companys crown jewels per se, but they are potential clues for malicious hackers to find them. It is a mistake to hardcode security components, such as security tokens or encryption keys, or privileged bits of code, such as API keys or proprietary algorithms, on the mobile device. Doing so may give malicious hackers the opportunity to steal those secrets by reverse-engineering the mobile app

FIX

If sensitive data is still required to be stored locally, it should be encrypted using a key derived from hardware backed storage which requires authentication

We have detected '_key = "url_key"' in the file androidmads/library/qrgenearator/QRGContents.java

```
line : 3 public class qrgcontents {

line : 4     public static final string note_key = "note_key";

line : 5     public static final string url_key = "url_key";

line : 6     public static final string[] phone_keys = {"phone", "secondary_phone", "tertiary_phone"};
```

We have detected '_key = "sharedaccesskey"' in the file com/microsoft/windowsazure/messaging/Connection.java

line : 24 private static final string api_version = "2014-09";

line : 25 private static final string api_version_key = "api-version";

line : 26 private static final string authorization_header = "authorization";

line : 27 private static final string endpoint_key = "endpoint";

line : 28 private static final int expire_minutes = 5;

line : 29 private static final string sdk_version = "2014-09";

line : 30 private static final string shared_access_key = "sharedaccesskey";

line : 31 private static final string shared_access_key_name = "sharedaccesskeyname";

We have detected '_key = "storage_version"' in the file com/microsoft/windowsazure/messaging/NotificationHub.java

line : 25 private static final string new_registration_location_header = "location";

line : 26 private static final string pns_handle_key = "pns_handle";

line : 27 private static final string registration_name_storage_key = "reg_name_";

line : 28 private static final string storage_prefix = "__nh_";

line : 29 private static final string storage_version = "1.0.0";

line : 30 private static final string storage_version_key = "storage_version";

line : 31 private static final string xml_content_type = "application/atom+xml";

We have detected '_password = "password"' in the file googlecom/codelab/meera/storage/DataBaseHelper.java

line : 86 private static final string key_offers_status = "status";

line : 87 private static final string key_password = "password";

line : 88 private static final string key_points = "points";

We have detected '_password = "logged-in-user-password"' in the file googlecom/codelab/meera/utils/Flags.java

line : 7 public static final string logged_in_user_name = "logged-in-user-name";

line : 8 public static final string logged_in_user_password = "logged-in-user-password";

line : 9 public static final string my_list = "my-list";

Severity
High

VULNERABILITY

Unsafe TrustManager implementation

OWASP MASVS	Common Weakness Enumeration
5.3 L2	CWE-295
Known Exploits	**Common Vulnerability Scoring System**
CVE-2020-5523	CVE-2020-5523-CVSS 3.0 Score 7.4

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05g-Testing-Network-Communication.md#testing-endpoint-identify-verification-mstg-network-3

Reference URL[s]:

https://developer.android.com/reference/javax/net/ssl/TrustManager
https://find-sec-bugs.github.io/bugs.htm#WEAK_TRUST_MANAGER
https://support.google.com/faqs/answer/6346016?hl=en
https://stackoverflow.com/questions/35545126/an-unsafe-implementation-of-the-interface-x509trustmanager-from-google
https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=134807561
https://www.jssec.org/dl/android_securecoding_en.pdf

CVSS BaseScore and Vector

CVE-2020-5523
Version & Base Score : CVSS 3.0 Score 7.4
CVSS Scoring Vector :
CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:N

THREAT

TrustManagers are responsible for managing the trust material that is used when making trust decisions and for deciding whether credentials presented by a peer should be accepted

RISK

The TrustManager interface might have been configured to trust all the server certificates, regardless of who signed it

An implementation ignoring all the SSL certificate validation errors when establishing an HTTPS connection to a remote host makes your app vulnerable to MitM attacks

Beginning 17 May 2016, Google Play started to block publishing any new apps or updates containing an unsafe implementation of the interface X509TrustManager

FIX

To properly handle SSL certificate validation, change your code in the checkServerTrusted method of your custom X509TrustManager interface to raise either CertificateException or IllegalArgumentException whenever the certificate presented by the server does not meet your expectations

We have detected 'X509Certificate[] getAcceptedIssuers()' in the file com/koushikdutta/async/AsyncSSLSocketWrapper.java

```
line : 20 import javax.net.ssl.TrustManagerFactory;

line : 21 import javax.net.ssl.X509TrustManager;

line : 22 import org.apache.http.conn.ssl.StrictHostnameVerifier;

line : 74                @Override

line : 75                public X509Certificate[] getAcceptedIssuers() {

line : 76                    return new X509Certificate[0];
```

Severity
High

VULNERABILITY

Unsafe files deletion

OWASP MASVS	Common Weakness Enumeration
2.10	CWE-200
Known Exploits	**Common Vulnerability Scoring System**
CVE-2018-3987	CVE-2018-3987-CVSS 3.0 Score 5.5

Best Practices:

https://www.youtube.com/watch?v=tGw1fxUD-uY

Reference URL[s]:

https://stackoverflow.com/a/22372329

CVSS BaseScore and Vector

CVE-2018-3987
Version & Base Score : CVSS 3.0 Score 5.5
CVSS Scoring Vector : CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

THREAT

When you delete a file using file.delete(), only the reference to the file is removed from the file system table. The file still exists on disk until other data overwrites it, leaving it vulnerable to recovery

RISK

Because of what just explained everything you delete may be recovered by any user or attacker, especially on rooted devices

FIX

First of all the developer must consider if there is an absolute need to save sensitive data in files that will be kept on customer side

If this is really required then - whenever there will be the need to remove a file - it is strongly suggested to first empty it [e.g. writing an empty string and saving it] before calling the actual deletion. This will fix the vulnerability

Additionally, please consider to encrypt any sensitive file in such a way that its content is protected against snooping also during its known life on device, not just after deletion

We have detected '.delete()' in the file com/koushikdutta/async/http/AsyncHttpClient.java

line : 28 import java.io.BufferedOutputStream;

line : 29 import java.io.File;

line : 30 import java.io.FileNotFoundException;

line : 537 }

line : 538 file.delete();

line : 539 }

line : 573 }

line : 574 this.val$file.delete();

line : 575 AsyncHttpClient.this.invoke(this.val$callback, this.val$ret, asyncHttpResponse, exc, null);

line : 596 if (e != null) {

line : 597 AnonymousClass9.this.val$file.delete();

line : 598 AsyncHttpClient.this.invoke(AnonymousClass9.this.val$callback, AnonymousClass9.this.val$ret, asyncHttpResponse, e, null);

We have detected '.delete()' in the file com/koushikdutta/async/util/FileCache.java

line : 3 import android.support.v4.media.session.PlaybackStateCompat;

line : 4 import java.io.File;

line : 5 import java.io.FileInputStream;

line : 137 for (File file : fileArr) {

line : 138 file.delete();

line : 139 }

line : 210 if (partFile.exists()) {

line : 211 partFile.delete();

line : 212 i++;

line : 242 if (cacheEntry2 == null && !FileCache.this.loading) {

line : 243 new File(FileCache.this.directory, str).delete();

line : 244 }

We have detected '.delete()' in the file com/koushikdutta/async/util/FileUtility.java

```
line : 2

line : 3 import java.io.File;

line : 4

line : 12                } else {

line : 13                    listFiles[i].delete();

line : 14                }

line : 16        }

line : 17        return file.delete();

line : 18    }
```

We have detected '.delete()' in the file com/koushikdutta/ion/FileCacheStore.java

```
line : 16 import com.koushikdutta.ion.gson.GsonSerializer;

line : 17 import java.io.File;

line : 18 import org.w3c.dom.Document;

line : 44                    if (exc != null) {

line : 45                        tempFile.delete();

line : 46                        simpleFuture.setComplete(exc);
```

We have detected '.delete()' in the file com/koushikdutta/ion/BitmapCallback.java

```
line : 11 import com.koushikdutta.ion.bitmap.PostProcess;

line : 12 import java.io.File;

line : 13 import java.io.FileOutputStream;

line : 33            } catch (Throwable th) {

line : 34                tempFile.delete();

line : 35                throw th;

line : 36            }

line : 37            tempFile.delete();

line : 38        }
```

We have detected '.delete()' in the file com/koushikdutta/ion/IonRequestBuilder.java

```
line : 64 import java.io.ByteArrayInputStream;

line : 65 import java.io.File;

line : 66 import java.io.InputStream;

line : 833            public void run() {

line : 834                file.delete();

line : 835            }
```

Severity
High

VULNERABILITY

Raw SQL queries used for SQLLite database

OWASP MASVS	Common Weakness Enumeration
6.2 [L1, L2]	CWE-89

Known Exploits	Common Vulnerability Scoring System
CVE-2019-5454	CVE-2019-5454-CVSS 3.0 Score 9.8
CVE-2020-0060	CVE-2020-0060-CVSS 3.0 Score 4.4

Best Practices:

https://hackerone.com/reports/291764

Reference URL[s]:

https://mobile-security.gitbook.io/mobile-security-testing-guide/general-mobile-app-testing-guide/0x04h-testing-code-quality#sql-injection

CVSS BaseScore and Vector

CVE-2019-5454
Version & Base Score : CVSS 3.0 Score 9.8
CVSS Scoring Vector :
CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

CVE-2020-0060
Version & Base Score : CVSS 3.0 Score 4.4
CVSS Scoring Vector : CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:N/A:N

THREAT

App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.

RISK

A SQL injection attack involves integrating SQL commands into input data, mimicking the syntax of a predefined SQL command. A successful SQL injection attack allows the attacker to read or write to the database and possibly execute administrative commands, depending on the permissions granted by the server.

FIX

Verify that the following best practices have been followed: -Untrusted inputs are type-checked and/or validated using a list of acceptable values. -Prepared statements with variable binding (i.e. parameterized queries) are used when performing database queries. If prepared statements are defined, user-supplied data and SQL code are automatically separated.

We have detected 'rawQuery(' in the file googlecom/codelab/meera/storage/DataBaseHelper.java

line : 5 import android.database.Cursor;

line : 6 import android.database.sqlite.SQLiteDatabase;

line : 7 import android.database.sqlite.SQLiteOpenHelper;

line : 8 import androidx.core.app.NotificationCompat;

line : 160 ArrayList<User> arrayList = new ArrayList<>();

line : 161 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM customer_info", null);

line : 162 if (!rawQuery.moveToFirst()) {

line : 229 ArrayList<Offer> arrayList = new ArrayList<>();

line : 230 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM offers", null);

line : 231 if (!rawQuery.moveToFirst()) {

line : 300 ArrayList<BestDeal> arrayList = new ArrayList<>();

line : 301 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM best_deals", null);

line : 302 if (!rawQuery.moveToFirst()) {

line : 358 ArrayList<Catalog> arrayList = new ArrayList<>();

line : 359 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM catalogs", null);

line : 360 if (!rawQuery.moveToFirst()) {

line : 410 ArrayList<FAQ> arrayList = new ArrayList<>();

line : 411 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM faqs", null);

line : 412 if (!rawQuery.moveToFirst()) {

line : 453 ArrayList<SLA> arrayList = new ArrayList<>();

line : 454 Cursor rawQuery = getReadableDatabase().rawQuery("SELECT * FROM sla", null);

line : 455 try {

Severity
High

VULNERABILITY

Read/Write access to External Storage

OWASP MASVS	Common Weakness Enumeration
2.14 L2	CWE-276
Known Exploits	**Common Vulnerability Scoring System**
CVE-2018-6599	CVE-2018-6599-CVSS 3.0 Score 5.5

Best Practices:

https://blog.checkpoint.com/2018/08/12/man-in-the-disk-a-new-attack-surface-for-android-apps

CVSS BaseScore and Vector

CVE-2018-6599
Version & Base Score : CVSS 3.0 Score 5.5
CVSS Scoring Vector : CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

THREAT

App can read/write to External Storage. Any App can read data written to External Storage.

RISK

Files created on external storage, such as SD Cards, are globally readable and writable. Because external storage can be removed by the user and also modified by any application, you should not store sensitive information using external storage.

FIX

As with data from any untrusted source, you should perform input validation when handling data from external storage. We strongly recommend that you not store executables or class files on external storage prior to dynamic loading. If your app does retrieve executable files from external storage, the files should be signed and cryptographically verified prior to dynamic loading.

We have detected '.getExternalStorage' in the file googlecom/codelab/meera/utils/FileUtils.java

```
line : 22       } else {
line : 23           file = new File(Environment.getExternalStorageDirectory() + "/Meera/" + replaceAll);
line : 24       }
```

We have detected '.getExternalStorage' in the file googlecom/codelab/meera/services/DownloadFile.java

```
line : 33       }
line : 34       File file = new File(Environment.getExternalStorageDirectory().toString(), "Meera");
line : 35       file.mkdir();
```

Severity
High

VULNERABILITY

Missing Certificate Pinning

OWASP MASVS	Common Weakness Enumeration
5.4 L2	CWE-295
	CWE-254
Known Exploits	**Common Vulnerability Scoring System**
CVE-2017-9968	CVE-2017-9968-CVSS 3.0 Score 5.9
CVE-2018-20200	CVE-2018-20200-CVSS 3.0 Score 5.9

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05g-Testing-Network-Communication.md#testing-custom-certificate-stores-and-certificate-pinning-mstg-network-4

Reference URL[s]:

https://medium.com/@appmattus/android-security-ssl-pinning-1db8acb6621e
https://www.owasp.org/index.php/Certificate_and_Public_Key_Pinning#Android
https://www.veracode.com/security/man-middle-attack
https://www.jssec.org/dl/android_securecoding_en.pdf

CVSS BaseScore and Vector

CVE-2017-9968
Version & Base Score : CVSS 3.0 Score 5.9
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N

CVE-2018-20200
Version & Base Score : CVSS 3.0 Score 5.9
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N

THREAT

An app can further protect itself from communicating with a wrong recipient by a technique known as Certificate Pinning. The general concept is that the client is configured to know the certificate expected to be received from the server. If the certificate presented doesn't match with the assigned one then the client will prevent the session to start

Some websites often rotate their certificates, so pinning against a specific certificate may discontinue the service. Typically though the public key inside these rotated certificates remains the same, so by pinning against it you are reducing the chances of bricking your app

RISK

If the Certificate Pinning is not implemented, an attacker [MITM - Man In The Middle] can position himself between the client and the real server. If the Certificate Authority is victim of a fraud they can issue a valid certificate to a criminal. Or the user can be induced to add a new trusted certificate authority. In this situation the handshake procedure for the client would occur with the attacker mimicking the server

This will cause a different public key to be sent to the client who - thinking to have received it from the original server - will send back its pre-master secret to start the communication. The MITM will complete the hack sending the pre-master secret to the original server. At this point the client and the server will be connected in a just apparently-secure way because the MITM has the same pre-master key to decrypt the traffic between the two parties

FIX

Quixxi App Shield can automatically implement the Certificate Pinning technique - pinning in reality against the SubjectPublicKeyInfo [SPKI] of the X.509 certificate - by enabling the "SSL certificate validation via SSL pinning" Shield entry. We support different solutions for this issue depending on the developer's original code [OkHttp, HttpURLConnection, Retrofit]. Any mismatch will cause the immediate termination of the app [fail hard strategy]

Implementing pinning validation from scratch should be avoided, as mistakes in this realization are extremely likely and usually lead to severe vulnerabilities

We have detected 'Retrofit.Builder()' in the file googlecom/codelab/meera/services/RetrofitClient.java

```
line : 4 import okhttp3.OkHttpClient;

line : 5 import retrofit2.Retrofit;

line : 6 import retrofit2.converter.gson.GsonConverterFactory;

line : 14        if (retrofit == null || ((str2 = url) != null && !str2.equals(str))) {

line : 15            retrofit = new Retrofit.Builder().baseUrl(str).client(new OkHttpClient.Builder().connectTimeout(60L, TimeUnit.SECONDS).writeTime
out(60L, TimeUnit.SECONDS).readTimeout(60L, TimeUnit.SECONDS).build()).addConverterFactory(GsonConverterFactory.create()).build();

line : 16            url = str;
```

Severity
Medium

VULNERABILITY

ADB Backup allowed

OWASP MASVS

2.8 L2

Common Weakness Enumeration

CWE-530

CWE-312

Known Exploits

CVE-2017-16835

Common Vulnerability Scoring System

CVE-2017-16835-CVSS 3.0 Score 7.5

Best Practices:

https://resources.infosecinstitute.com/android-hacking-security-part-15-hacking-android-apps-using-backup-techniques/
https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05d-Testing-Data-Storage.md#testing-backups-for-sensitive-data-mstg-storage-8
https://securitygrind.com/exploiting-android-backup/

Reference URL[s]:

https://androidquest.wordpress.com/2014/09/18/backup-applications-on-android-phone-with-adb/
https://developer.android.com/guide/topics/data/autobackup#IncludingFiles

CVSS BaseScore and Vector

CVE-2017-16835
Version & Base Score : CVSS 3.0 Score 7.5
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

THREAT

The Android operating system offers a backup/restore mechanism of installed packages through the ADB utility. The full backup of applications including the private files stored on data partition is allowed by default

RISK

The attacker can access the backup and any sensitive data included [e.g. passwords] is exposed

FIX

In order to fix this issue please consider to disable ADB Backup for any apps that might include sensitive content. You can also customize your backup by implementing a BackupAgent class

ADB Backup is enabled for this app (default: ENABLED)

<application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/meera_app_icon" android:name="googlecom.codelab.meera.utils.MyApplication" android:allowBackup="true" android:supportsRtl="true" android:usesCleartextTraffic="true" android:roundIcon="@mipmap/meera_app_icon" android:appComponentFactory="androidx.core.app.CoreComponentFactory" android:requestLegacyExternalStorage="true">

Severity
Medium

VULNERABILITY

Debugging Information Provision

OWASP MASVS

7.4 L2

Common Weakness Enumeration

CWE-215

Known Exploits

CAPEC-133

CVE-2018-6599

Common Vulnerability Scoring System

CVE-2018-6599-CVSS 3.0 Score 5.5

Reference URL[s]:

https://github.com/b66l/OASAM/blob/master/oasam-leak-information-leak/oasam-leak-002-information-leak-to-log-files.md

CVSS BaseScore and Vector

CAPEC-133

CVE-2018-6599
Version & Base Score : CVSS 3.0 Score 5.5
CVSS Scoring Vector : CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N

THREAT

Applications can output runtime information using the android.util.Log class. Logcats are usually collected [e.g. via adb logcat -v long > logcat.txt] in order to bugfix the app but they should be removed once done

RISK

If the app is configured to check the behavior in code sections where the developer is dealing with users sensitive data then this info can leak. Besides this, any log line can theoretically give unrequired details about the app's logic

For example an app log might reveal a strategy for detecting rooted phones based on two different methods. This will tell a hacker that in order to run the app on rooted phones he will need to overcome two checks

FIX

Quixxi App Shield can automatically remove the debug logs [Log.i(), Log.d(), Log.v(), System.out.println()] preventing them to provide information by enabling the "Remove app logs" Shield entry

One common manual solution is declaring and using a custom log class, so that the output log is automatically turned on/off based on Debug/Release. If the app includes any third-party libraries it is developer's responsibility to guarantee safety conditions also for the additional files included

We have detected 'Log.d(' in the file androidmads/library/qrgenearator/QRGSaver.java

```
line : 3 import android.graphics.Bitmap;
line : 4 import android.util.Log;
line : 5 import androidmads.library.qrgenearator.QRGContents;
line : 17 } else {
line : 18 Log.v("QRGSaver", "Folder Exists");
line : 19 }
line : 26 } catch (IOException e) {
line : 27 Log.d("QRGSaver", e.toString());
line : 28 return false;
```

We have detected 'Log.i(' in the file com/koushikdutta/async/AsyncServer.java

line : 4 import android.os.Handler;
line : 5 import android.util.Log;
line : 6 import com.koushikdutta.async.callback.CompletedCallback;
line : 158 } catch (Exception unused) {
line : 159 Log.i(AsyncServer.LOGTAG, "Selector Exception? L Preview?");
line : 160 }
line : 232 } catch (InterruptedException e) {
line : 233 Log.e(LOGTAG, "run", e);
line : 234 }
line : 459 } catch (IOException e) {
line : 460 Log.e(AsyncServer.LOGTAG, "Datagram error", e);
line : 461 StreamUtility.closeQuietly(open);
line : 485 } catch (IOException e) {
line : 486 Log.e(AsyncServer.LOGTAG, "Datagram error", e);
line : 487 StreamUtility.closeQuietly(open);
line : 532 if (this.mSelector != null) {
line : 533 Log.i(LOGTAG, "Reentrant call");
line : 534 z2 = true;
line : 573 } catch (AsyncSelectorException e) {
line : 574 Log.i(LOGTAG, "Selector closed", e);
line : 575 try {
line : 591 } catch (AsyncSelectorException e) {
line : 592 Log.i(LOGTAG, "Selector exception, shutting down", e);
line : 593 try {
line : 745 } else {
line : 746 Log.i(LOGTAG, "wtf");
line : 747 throw new RuntimeException("Unknown key state.");
line : 763 if (AsyncServer.this.mSelector == null) {
line : 764 Log.i(AsyncServer.LOGTAG, "Server dump not possible. No selector?");
line : 765 return;
line : 766 }
line : 767 Log.i(AsyncServer.LOGTAG, "Key Count: " + AsyncServer.this.mSelector.keys().size());
line : 768 Iterator<SelectionKey> it = AsyncServer.this.mSelector.keys().iterator();
line : 769 while (it.hasNext()) {
line : 770 Log.i(AsyncServer.LOGTAG, "Key: " + it.next());
line : 771 }

We have detected 'Log.e(' in the file com/koushikdutta/async/AsyncNetworkSocket.java

line : 2
line : 3 import android.util.Log;
line : 4 import com.koushikdutta.async.callback.CompletedCallback;
line : 219 } else if (exc != null) {
line : 220 Log.e(AsyncServer.LOGTAG, "Unhandled exception", exc);
line : 221 }

We have detected 'Log.e(' in the file com/koushikdutta/async/PushParser.java

line : 2
line : 3 import android.util.Log;
line : 4 import com.koushikdutta.async.callback.DataCallback;
line : 245 } catch (Exception e) {
line : 246 Log.e("PushParser", "Error while invoking tap callback", e);
line : 247 }

We have detected 'System.out.print' in the file com/koushikdutta/async/ByteBufferList.java

line : 346 public void spewString() {
line : 347 System.out.println(peekString());
line : 348 }

We have detected 'Log.e(' in the file com/koushikdutta/async/http/HybiParser.java

line : 2
line : 3 import android.util.Log;
line : 4 import com.koushikdutta.async.ByteBufferList;
line : 377 } catch (Exception e) {
line : 378 Log.e(TAG, "inflater.end failed", e);
line : 379 }

We have detected 'Log.e(' in the file com/koushikdutta/async/http/AsyncHttpRequest.java

line : 3 import android.net.Uri;
line : 4 import android.util.Log;
line : 5 import com.koushikdutta.async.AsyncSSLException;
line : 211 if (str2 != null && this.logLevel <= 4) {
line : 212 Log.i(str2, getLogMessage(str));
line : 213 }
line : 218 if (str2 != null && this.logLevel <= 2) {
line : 219 Log.v(str2, getLogMessage(str));
line : 220 }
line : 225 if (str2 != null && this.logLevel <= 5) {
line : 226 Log.w(str2, getLogMessage(str));
line : 227 }
line : 232 if (str2 != null && this.logLevel <= 3) {
line : 233 Log.d(str2, getLogMessage(str));
line : 234 }
line : 239 if (str2 != null && this.logLevel <= 3) {
line : 240 Log.d(str2, getLogMessage(str));
line : 241 Log.d(this.LOGTAG, exc.getMessage(), exc);
line : 242 }
line : 247 if (str2 != null && this.logLevel <= 6) {
line : 248 Log.e(str2, getLogMessage(str));
line : 249 }
line : 254 if (str2 != null && this.logLevel <= 6) {
line : 255 Log.e(str2, getLogMessage(str));
line : 256 Log.e(this.LOGTAG, exc.getMessage(), exc);
line : 257 }

We have detected 'System.out.print' in the file com/koushikdutta/async/http/server/AsyncHttpServerRequestImpl.java

line : 75 protected void onNotHttp() {
line : 76 System.out.println("not http!");
line : 77 }

We have detected 'Log.w(' in the file com/koushikdutta/ion/IonRequestBuilder.java

line : 9 import android.util.Base64;
line : 10 import android.util.Log;
line : 11 import android.widget.ImageView;
line : 118 if (isAlive != null) {
line : 119 Log.w("Ion", "Building request with dead context: " + isAlive);
line : 120 }

We have detected 'Log.i(' in the file com/koushikdutta/ion/Ion.java

line : 9 import android.text.TextUtils;
line : 10 import android.util.Log;
line : 11 import android.widget.ImageView;
line : 181 } catch (IOException e) {
line : 182 IonLog.w("unable to set up response cache, clearing", e);
line : 183 FileUtility.deleteDirectory(file);
line : 186 } catch (IOException unused) {
line : 187 IonLog.w("unable to set up response cache, failing", e);
line : 188 }
line : 330 String str = this.logtag;
line : 331 Log.i(str, "Pending bitmaps: " + this.bitmapsPending.size());
line : 332 String str2 = this.logtag;
line : 333 Log.i(str2, "Groups: " + this.inFlight.size());
line : 334 Iterator<FutureSet> it = this.inFlight.values().iterator();
line : 336 String str3 = this.logtag;
line : 337 Log.i(str3, "Group size: " + it.next().size());
line : 338 }

We have detected 'Log.w(' in the file com/koushikdutta/ion/IonLog.java

```
line : 2
line : 3 import android.util.Log;
line : 4
line : 13 if (debug) {
line : 14 Log.d(LOGTAG, str, exc);
line : 15 }
line : 18 public static void e(String str, Exception exc) {
line : 19 Log.e(LOGTAG, str, exc);
line : 20 }
line : 22 public static void i(String str, Exception exc) {
line : 23 Log.i(LOGTAG, str, exc);
line : 24 }
line : 26 public static void w(String str, Exception exc) {
line : 27 Log.w(LOGTAG, str, exc);
line : 28 }
line : 31 if (debug) {
line : 32 Log.d(LOGTAG, str);
line : 33 }
line : 36 public static void e(String str) {
line : 37 Log.e(LOGTAG, str);
line : 38 }
line : 40 public static void i(String str) {
line : 41 Log.i(LOGTAG, str);
line : 42 }
line : 44 public static void w(String str) {
line : 45 Log.w(LOGTAG, str);
line : 46 }
```

We have detected 'Log.e(' in the file com/koushikdutta/ion/bitmap/Exif.java

```
line : 2
line : 3 import android.util.Log;
line : 4
line : 31 } else {
line : 32 Log.e(TAG, "Invalid length");
line : 33 return 0;
line : 67 }
line : 68 Log.i(TAG, "Unsupported orientation");
line : 69 return 0;
line : 75 } else {
line : 76 Log.e(TAG, "Invalid offset");
line : 77 }
line : 78 } else {
line : 79 Log.e(TAG, "Invalid byte order");
line : 80 return 0;
```

We have detected 'Log.i(' in the file com/koushikdutta/ion/bitmap/IonBitmapCache.java

```
line : 12 import android.util.DisplayMetrics;
line : 13 import android.util.Log;
line : 14 import android.view.WindowManager;
line : 85 if (bitmapInfo.bitmap != null && bitmapInfo.bitmap.isRecycled()) {
line : 86 Log.w(IonLog.LOGTAG, "Cached bitmap was recycled.");
line : 87 Log.w(IonLog.LOGTAG, "This may happen if passing Ion bitmaps directly to notification builders or remote media clients.");
line : 88 Log.w(IonLog.LOGTAG, "Create a deep copy before doing this.");
line : 89 this.cache.remove(str);
line : 99 public void dump() {
line : 100 Log.i("IonBitmapCache", "bitmap cache: " + this.cache.size());
line : 101 Log.i("IonBitmapCache", "freeMemory: " + Runtime.getRuntime().freeMemory());
line : 102 }
```

We have detected 'Log.w(' in the file com/koushikdutta/ion/conscrypt/ConscryptMiddleware.java

```
line : 3 import android.content.Context;
line : 4 import android.util.Log;
line : 5 import com.google.android.gms.security.ProviderInstaller;
line : 56 } catch (Throwable th) {
line : 57 Log.w(LOGTAG, "Conscrypt initialization failed.", th);
line : 58 }
```

We have detected 'Log.w(' in the file com/koushikdutta/ion/gif/GifDecoder.java

line : 3 import android.graphics.Bitmap;
line : 4 import android.util.Log;
line : 5 import androidx.core.view.ViewCompat;
line : 575 } else if (i != 3) {
line : 576 Log.w("lon", "Unknown gif dispose code: " + this.lastDispose);
line : 577 }

We have detected 'Log.e(' in the file com/koushikdutta/ion/cookie/CookieMiddleware.java

line : 4 import android.text.TextUtils;
line : 5 import android.util.Log;
line : 6 import com.koushikdutta.async.http.AsyncHttpClientMiddleware;
line : 59 } catch (Exception e) {
line : 60 Log.e("lon", "unable to load cookies", e);
line : 61 }

We have detected 'Log.e(' in the file com/shockwave/pdfium/PdfiumCore.java

line : 5 import android.os.ParcelFileDescriptor;
line : 6 import android.util.Log;
line : 7 import android.view.Surface;
line : 198 e = e3;
line : 199 Log.e(TAG, "mContext may be null");
line : 200 e.printStackTrace();
line : 202 e2 = e4;
line : 203 Log.e(TAG, "Exception throw from native");
line : 204 e2.printStackTrace();
line : 235 e = e3;
line : 236 Log.e(TAG, "mContext may be null");
line : 237 e.printStackTrace();
line : 239 e2 = e4;
line : 240 Log.e(TAG, "Exception throw from native");
line : 241 e2.printStackTrace();

We have detected 'Log.e(' in the file com/microsoft/windowsazure/notifications/NotificationsManager.java

line : 6 import android.preference.PreferenceManager;
line : 7 import android.util.Log;
line : 8 import com.google.android.gms.gcm.GoogleCloudMessaging;
line : 29 } catch (Exception e) {
line : 30 Log.e("NotificationsManager", e.toString());
line : 31 return null;
line : 50 } catch (Exception e) {
line : 51 Log.e("NotificationsManager", e.toString());
line : 52 }

We have detected 'Log.e(' in the file com/github/barteksc/pdfviewer/PDFView.java

line : 14 import android.util.AttributeSet;
line : 15 import android.util.Log;
line : 16 import android.view.SurfaceView;
line : 497 } else {
line : 498 Log.e(TAG, "load pdf error", th);
line : 499 }

We have detected 'Log.i(' in the file googlecom/codelab/meera/activities/RegisterActivity.java

line : 10 import android.text.TextWatcher;
line : 11 import android.util.Log;
line : 12 import android.util.Patterns;
line : 124 while (it.hasNext()) {
line : 125 System.out.println((String) it.next());
line : 126 }
line : 427 String lowerCase = str.toLowerCase();
line : 428 Log.i("ErrrrrrrrorrrRR", lowerCase);
line : 429 if (lowerCase.contains("duplicate email address")) {

We have detected 'Log.i(' in the file googlecom/codelab/meera/activities/MainActivity.java

line : 10 import android.text.style.AlignmentSpan;
line : 11 import android.util.Log;
line : 12 import android.view.Menu;
line : 374 public void registerWithNotificationHubs() {
line : 375 Log.i("MainActivity", " Registering with Notification Hubs");
line : 376 if (checkPlayServices()) {
line : 390 }
line : 391 Log.i("MainActivity", "This device is not supported by Google Play Services.");
line : 392 finish();

We have detected 'Log.i(' in the file googlecom/codelab/meera/fragments/TransactionsFragment.java

line : 6 import android.os.Bundle;
line : 7 import android.util.Log;
line : 8 import android.view.LayoutInflater;
line : 103 LoaderUtils.hide();
line : 104 Log.i("FFFFFFF", "fail");
line : 105 }
line : 123 LoaderUtils.hide();
line : 124 Log.i("SSSSSSSSS", response.body().toString());
line : 125 if (response.body().getAsJsonObject().get("returnCode").toString().contains("Success")) {

We have detected 'Log.i(' in the file googlecom/codelab/meera/fragments/ContactUsFragment.java

line : 5 import android.os.Bundle;
line : 6 import android.util.Log;
line : 7 import android.view.LayoutInflater;
line : 95 LoaderUtils.hide();
line : 96 Log.i("SSSSSSSSS", response.body().toString());
line : 97 if (response.body().getAsJsonObject().get("returnCode").toString().contains("Success")) {

We have detected 'Log.e(' in the file googlecom/codelab/meera/fragments/MyCardsFragment.java

line : 6 import android.os.Handler;
line : 7 import android.util.Log;
line : 8 import android.view.Display;
line : 229 } catch (WriterException e) {
line : 230 Log.e("Tag", e.toString());
line : 231 }

We have detected 'Log.i(' in the file googlecom/codelab/meera/fragments/EditProfileFragment.java

line : 8 import android.os.Bundle;
line : 9 import android.util.Log;
line : 10 import android.util.Patterns;
line : 241 String lowerCase = str.toLowerCase();
line : 242 Log.i("ErrrrrrrrorrrRR", lowerCase);
line : 243 if (lowerCase.contains("duplicate email address")) {

Severity
Medium

VULNERABILITY

Weak Hashing Algorithms

OWASP MASVS	Common Weakness Enumeration
3.4 [L1, L2]	CWE-326

Known Exploits	Common Vulnerability Scoring System
CVE-2018-14992	CVE-2018-14992-CVSS 3.0 Score 5.5
CVE-2017-15999	CVE-2017-15999-CVSS 3.0 Score 9.8

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x04g-Testing-Cryptography.md#identifying-insecure-andor-deprecated-cryptographic-algorithms-mstg-crypto-4

Reference URL[s]:

https://developer.android.com/reference/java/security/MessageDigest.html
https://valerieaurora.org/hash.html
https://www.computerworld.com/article/3173616/the-sha1-hash-function-is-now-completely-unsafe.html

CVSS BaseScore and Vector

CVE-2018-14992
Version & Base Score : CVSS 3.0 Score 5.5
CVSS Scoring Vector : CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:N

CVE-2017-15999
Version & Base Score : CVSS 3.0 Score 9.8
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

THREAT

The mobile application leverages weak hashing algorithms. Weak hashing algorithms [e.g. MD2, MD4, MD5, SHA-0 or SHA-1] can be vulnerable to hash collisions, so they should not be used when reliable data hashing is required

RISK

Hashing algorithms are widely used to validate credit card transactions, electronic documents, email PGP/GPG signatures, open-source software repositories, backups and software updates

If a weakness is found in a hash function that allows for two files to have the same digest, the function is considered cryptographically broken, because digital fingerprints generated with it can be forged and cannot be trusted. Attackers could, for example, create a rogue software update that would be accepted and executed by an update mechanism that validates updates by checking digital signatures

FIX

Please be sure to put in place a strong hashing algorithm, like SHA-2 [the SHA-256 function is a very popular choice] or - better - SHA-3

We have detected 'MessageDigest.getInstance("SHA-1")' in the file com/koushikdutta/async/http/WebSocketImpl.java

```
line : 55 try {
line : 56 MessageDigest instance = MessageDigest.getInstance("SHA-1");
line : 57 instance.update(str.getBytes("iso-8859-1"), 0, str.length());
```

We have detected 'MessageDigest.getInstance("MD5")' in the file com/koushikdutta/async/http/spdy/ByteString.java

```
line : 236 try {
line : 237 return String.format(Locale.ENGLISH, "ByteString[size=%s md5=%s]", Integer.valueOf(this.data.length), of(MessageDigest.getInstance("
MD5").digest(this.data)).hex());
line : 238 } catch (NoSuchAlgorithmException unused) {
```

We have detected 'MessageDigest.getInstance("MD5")' in the file com/koushikdutta/async/util/FileCache.java

```
line : 92 try {
line : 93 messageDigest = MessageDigest.getInstance("MD5");
line : 94 } catch (NoSuchAlgorithmException e) {
```

Severity
Medium

VULNERABILITY

Weak Random Number Generator

OWASP MASVS

3.6 [L1, L2]

Common Weakness Enumeration

CWE-1241

Known Exploits

Multiple vulnerabilities

Common Vulnerability Scoring System

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05e-Testing-Cryptography.md#testing-random-number-generation-mstg-crypto-6

Reference URL[s]:

https://developer.android.com/reference/java/util/Random.html
https://developer.android.com/reference/java/security/SecureRandom.html
https://arstechnica.com/information-technology/2013/08/google-confirms-critical-android-crypto-flaw-used-in-5700-bitcoin-heist/

CVSS BaseScore and Vector

Multiple vulnerabilities

THREAT

Developers generally implement random number generators [RNGs] where the random number is fully determined by the seed knowledge. This is the reason why they are called pseudo-random number generators [PRNGs]

When it comes to cryptography, random numbers play a fundamental role in: key generation nonces one-time pads salts in certain signature schemes

RISK

Using standard PRNGs is a bad practice when implementing security mechanisms, since the attacker may be able to guess the logic behind and predict the generated random numbers. In this case the confidentiality and/or integrity of the vulnerable app might be undermined

FIX

Don't use standard random() method or Pseudo-Random Number Generators [PRNGs] because they will NOT really return non-predictable random numbers

Use cryptographically secure pseudorandom number generators [CSPRNG] such a directly provided by Google. CSPRNG are able to pass the "next-bit" test and to hold up well under serious attack, even when part of their initial or running state becomes available to an attacker

We have detected ' Random()' in the file com/koushikdutta/async/dns/Dns.java

```
line : 18 import java.nio.ByteOrder;
line : 19 import java.util.Random;
line : 20
line : 61 ByteBuffer order = ByteBufferList.obtain(1024).order(ByteOrder.BIG_ENDIAN);
line : 62 short nextInt = (short) new Random().nextInt();
line : 63 short query = (short) setQuery(0);
```

We have detected ' Random()' in the file com/koushikdutta/async/util/FileCache.java

```
line : 16 import java.util.Iterator;
line : 17 import java.util.Random;
line : 18 import java.util.Set;
line : 26 long size;
line : 27 Random random = new Random();
line : 28 long blockSize = PlaybackStateCompat.ACTION_SKIP_TO_QUEUE_ITEM;
```

We have detected ' Random()' in the file googlecom/codelab/meera/utils/NotificationUtil.java

```
line : 15 import java.util.Date;
line : 16 import java.util.Random;
line : 17
line : 50 PendingIntent activity = PendingIntent.getActivity(context, 0, intent, BasicMeasure.EXACTLY);
line : 51 int time = ((int) ((new Date().getTime() / 1000) % 2147483647L)) + new Random().nextInt(8999) + 1000;
line : 52 String string = context.getString(2131689544);
```

Severity
Medium

VULNERABILITY

Missing copy&paste protection from EditText fields

OWASP MASVS	Common Weakness Enumeration
2.10	CWE-200
Known Exploits	**Common Vulnerability Scoring System**
CVE-2018-12481	CVE-2018-12481-CVSS 3.0 Score 9.8

Best Practices:

https://pastebin.com/sp5nMhvc

Reference URL[s]:

https://www.researchgate.net/publication/300578051_Attacks_on_Android_Clipboard
https://github.com/grepx/android-clipboard-security
https://stackoverflow.com/questions/19963785/service-android-clipboard-listener

CVSS BaseScore and Vector

CVE-2018-12481
Version & Base Score : CVSS 3.0 Score 9.8
CVSS Scoring Vector :
CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

THREAT

The clipboard is a powerful framework to support various types of data copy and paste within an app as well as among Android apps. There is a flaw in Android's API that allows any installed application to listen to changes to the clipboard [listen to everything that is copied and pasted]

RISK

Clipboard data manipulation may lead to common code injection attacks, like JavaScript injection and command injection. Furthermore, it can also cause phishing attacks, including web phishing and app phishing. Data stealing happens when sensitive data copied into the clipboard is accessed by malicious applications

FIX

Quixxi App Shield can automatically prevent the app EditText fields to be subject to copy and paste by enabling the "Disable copy & paste functionality" Shield option

In general, please try to disable this functionality whenever your user may be required to insert sensitive information

We have detected '(EditText)' in the file googlecom/codelab/meera/activities/RegisterActivity.java

line : 18 import android.widget.DatePicker;
line : 19 import android.widget.EditText;
line : 20 import android.widget.RadioButton;
line : 43
line : 44 public class RegisterActivity extends AppCompatActivity {
line : 45 RadioButton arRadioButton;
line : 75 public void initLayoutElements() {
line : 76 this.firstNameField = (EditText) findViewById(R.id.firstnameField);
line : 77 this.lastNameField = (EditText) findViewById(R.id.lastnameField);
line : 78 this.birthdayField = (EditText) findViewById(R.id.birthdayField);
line : 79 this.mobileNumberField = (EditText) findViewById(R.id.mobileNumberField);
line : 80 this.emailAddressField = (EditText) findViewById(R.id.emailAddressField);
line : 81 this.passwordField = (EditText) findViewById(R.id.passwordField);
line : 82 this.confirmPasswordField = (EditText) findViewById(R.id.confirmPasswordField);
line : 83 this.maleRadioButton = (RadioButton) findViewById(R.id.male_radioButton);
line : 88 this.confessRadioButton = (RadioButton) findViewById(R.id.confessRadioButton);
line : 89 this.qatarilDField = (EditText) findViewById(R.id.qatarilDField);
line : 90 this.nationalityField = (EditText) findViewById(R.id.nationalityField);
line : 91 this.emailAddressField.setOnFocusChangeListener(new View.OnFocusChangeListener() {
line : 212 builder.setView(inflate);
line : 213 final EditText editText = (EditText) inflate.findViewById(R.id.firstFiled);
line : 214 final EditText editText2 = (EditText) inflate.findViewById(R.id.secondFiled);
line : 215 final EditText editText3 = (EditText) inflate.findViewById(R.id.thirdFiled);
line : 216 final EditText editText4 = (EditText) inflate.findViewById(R.id.fourthFiled);
line : 217 editText.addTextChangedListener(new TextWatcher() {

We have detected '(EditText)' in the file googlecom/codelab/meera/activities/MainActivity.java

line : 16 import android.widget.Button;
line : 17 import android.widget.EditText;
line : 18 import android.widget.FrameLayout;
line : 65
line : 66 public class MainActivity extends AppCompatActivity {
line : 67 Activity activity;
line : 415 button2.setText(R.string.cancel);
line : 416 EditText editText = (EditText) dialog.findViewById(R.id.dialogAccountID);
line : 417 this.dialogSpinner = (Spinner) dialog.findViewById(R.id.dialogSpinner);

We have detected '(EditText)' in the file googlecom/codelab/meera/activities/LoginActivity.java

line : 13 import android.widget.Button;
line : 14 import android.widget.EditText;
line : 15 import android.widget.TextView;
line : 49
line : 50 public class LoginActivity extends AppCompatActivity {
line : 51 private static final String KEY_NAME = "yourKey";
line : 83 currentContext = this;
line : 84 this.emailOrMobileEditText = (EditText) findViewById(R.id.emailOrMobileNumber);
line : 85 this.passwordEditText = (EditText) findViewById(R.id.password);
line : 86 TextView textView = (TextView) findViewById(R.id.forgotPassword_textView);

We have detected '(EditText)' in the file googlecom/codelab/meera/activities/ForgotPasswordActivity.java

line : 12 import android.widget.Button;
line : 13 import android.widget.EditText;
line : 14 import android.widget.Toast;
line : 27
line : 28 public class ForgotPasswordActivity extends AppCompatActivity {
line : 29 AuthenticationsService authenticationsService;
line : 41 this.authenticationsService = APIUtils.getAuthenticationsService();
line : 42 this.email = (EditText) findViewById(R.id.forgotEmail);
line : 43 this.password = (EditText) findViewById(R.id.forgotPassword);
line : 44 this.confirmPassword = (EditText) findViewById(R.id.forgotConfirmPassword);
line : 45 Button button = (Button) findViewById(R.id.forgotSubmit);
line : 58 builder.setView(inflate);
line : 59 final EditText editText = (EditText) inflate.findViewById(R.id.firstFiled);
line : 60 final EditText editText2 = (EditText) inflate.findViewById(R.id.secondFiled);
line : 61 final EditText editText3 = (EditText) inflate.findViewById(R.id.thirdFiled);
line : 62 final EditText editText4 = (EditText) inflate.findViewById(R.id.fourthFiled);
line : 63 editText.addTextChangedListener(new TextWatcher() {

Severity
Medium

VULNERABILITY

Missing protection against screenshots & screensharing

OWASP MASVS

2.7 L2

Common Weakness Enumeration

CWE-200

Known Exploits

CVE-2015-6630

Common Vulnerability Scoring System

CVE-2015-6630-CVSS 2.0 Score 4.3

Best Practices:

https://www.securityweek.com/screenaudio-capture-vulnerability-impacts-lions-share-android-devices

Reference URL[s]:

https://www.xda-developers.com/android-screen-recording-vulnerability/
https://threatpost.com/mobile-malware-captures-keystrokes-screengrabs/103973/

CVSS BaseScore and Vector

CVE-2015-6630
Version & Base Score : CVSS 2.0 Score 4.3
CVSS Scoring Vector : AV:N/AC:M/Au:N/C:P/I:N/A:N

THREAT

Screenshots and screensharing are useful and powerful features whenever they can provide an additional value to the target app

RISK

In specific cases though, not protecting the app against screenshot captures and screensharing exposes the user to the leakage of sensitive information [e.g. healthcare apps]

FIX

Quixxi App Shield can automatically prevent against screenshots and screensharing functionalities by enabling the "Disable screenshots capture & screen sharing" Shield option. The same option will automatically provide protection to blur the app preview when it is placed in the background

Severity
Medium

VULNERABILITY

No blurring for the app in background

OWASP MASVS

2.9 L2

Common Weakness Enumeration

CWE-200

Known Exploits

CVE-2015-6630

Common Vulnerability Scoring System

CVE-2015-6630-CVSS 2.0 Score 4.3

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05d-Testing-Data-Storage.md#finding-sensitive-information-in-auto-generated-screenshots-mstg-storage-9

CVSS BaseScore and Vector

CVE-2015-6630
Version & Base Score : CVSS 2.0 Score 4.3
CVSS Scoring Vector : AV:N/AC:M/Au:N/C:P/I:N/A:N

THREAT

When a user puts an app in background, the same information shown in the foreground is displayed in the background as preview

RISK

The lack of background screen blurring can - as a minimum - cause data leakage [e.g. banking amounts]. Such previews can be also potentially exported via screenshots

FIX

Quixxi App Shield can automatically blur the app preview when it is put in background by enabling the "Disable screenshots capture & screen sharing" Shield option. The same option will automatically provide protection against screenshots and screen sharing

Severity
Low

VULNERABILITY

Application uses HTTPURLConnection

OWASP MASVS	**Common Weakness Enumeration**
-	-
Known Exploits	**Common Vulnerability Scoring System**
-	-

Reference URL[s]:

CVSS BaseScore and Vector

-

THREAT

HttpURLConnection class allows to send information in clear text (http) without any encryption. Sending any sensitive information over a http protocol is unsafe. It may help attacker to perform MITM attack

RISK

Sensiitive information in clear text can be capture by attacker and it can be misused. It is easy for the attackers to perform MITM attack

FIX

Use HTTPSURLConnection to enable https connection to servers. This will help to prevent MITM attacks. Additionally add SSLPinning Prevention to prevent the MITM attacks.

We have detected 'connect' in the file com/koushikdutta/async/http/cache/ResponseHeaders.java

```
line : 291 if (j3 >= computeFreshnessLifetime) {
line : 292 this.headers.add("Warning", "110 HttpURLConnection \"Response is stale\"");
line : 293 }
line : 294 if (computeAge > 86400000 && isFreshnessLifetimeHeuristic()) {
line : 295 this.headers.add("Warning", "113 HttpURLConnection \"Heuristic expiration\"");
line : 296 }
line : 18 private int ageSeconds;
line : 19 private String connection;
line : 20 private String contentEncoding;
line : 100 } else if ("Connection".equalsIgnoreCase(fieldName)) {
line : 101 this.connection = value;
line : 102 } else if ("Proxy-Authenticate".equalsIgnoreCase(fieldName)) {
line : 127 public boolean hasConnectionClose() {
line : 128 return "close".equalsIgnoreCase(this.connection);
line : 129 }
line : 191 public String getConnection() {
line : 192 return this.connection;
line : 193 }
```

We have detected 'openConnection' in the file com/microsoft/windowsazure/messaging/Connection.java

```
line : 8 import java.io.UnsupportedEncodingException;
line : 9 import java.net.HttpURLConnection;
line : 10 import java.net.URI;
line : 50 }
line : 51 HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(AddApiVersionToUrl(str6 + str)).openConnection();
line : 52 httpURLConnection.setRequestMethod(str4);
line : 75
line : 76 private java.lang.String executeRequest(java.net.HttpURLConnection r6, java.lang.String r7, java.lang.String r8) throws java.lang.Excepti
on {
line : 77
line : 78 throw new UnsupportedOperationException("Method not decompiled: com.microsoft.windowsazure.messaging.Connection.executeReque
st(java.net.HttpURLConnection, java.lang.String, java.lang.String):java.lang.String");
line : 79 }
line : 80
line : 81 private String getResponseContent(HttpURLConnection httpURLConnection) throws IOException {
line : 82 try {
line : 99
line : 100 private void addAuthorizationHeader(HttpURLConnection httpURLConnection) throws InvalidKeyException {
line : 101 httpURLConnection.setRequestProperty(AUTHORIZATION_HEADER, generateAuthToken(httpURLConnection.getURL().toString()));
```

We have detected 'connect' in the file googlecom/codelab/meera/model/FileDownloader.java

```
line : 7 import java.io.InputStream;
line : 8 import java.net.HttpURLConnection;
line : 9 import java.net.MalformedURLException;
line : 16 try {
line : 17 HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(str).openConnection();
line : 18 httpURLConnection.connect();
line : 19 InputStream inputStream = httpURLConnection.getInputStream();
```

Severity
Low

VULNERABILITY

Native binaries contains debugging symbols

OWASP MASVS	Common Weakness Enumeration
7.3	-
Known Exploits	**Common Vulnerability Scoring System**
-	-

Reference URL[s]:

https://github.com/OWASP/owasp-mastg/blob/1.1.3-excel/Document/0x05i-Testing-Code-Quality-and-Build-Settings.md#testing-for-debugging-symbols-mstg-code-3

CVSS BaseScore and Vector

-

THREAT

Compiled code of application must be with as little explanation as possible. Some metadata, such as debugging information, line numbers, and descriptive function or method names, make the binary or byte-code easier for the reverse engineer to understand.

These metadata informations are not needed in a release build and it can be safely omitted without impacting the apps functionality.

RISK

Application binaries with debug symbols helps attackers to debug the application, understand the source code after reverse engineering it.

FIX

Dynamic symbols can be stripped via the visibility compiler flag. Adding this flag causes gcc to discard the function names while preserving the names of functions declared as JNIEXPORT.

Make sure that the following has been added to build.gradle:
externalNativeBuild {
cmake {
cppFlags "-fvisibility=hidden"
}
}

Library libjniPdfium.so contains following dynamic debug symbols

1. U abort
2. U ANativeWindow_fromSurface
3. U ANativeWindow_getFormat
4. U ANativeWindow_getHeight
5. U ANativeWindow_getWidth
6. U ANativeWindow_lock
7. U ANativeWindow_release
8. U ANativeWindow_setBuffersGeometry
9. U ANativeWindow_unlockAndPost
10. U AndroidBitmap_getInfo
11. U AndroidBitmap_lockPixels
12. U AndroidBitmap_unlockPixels
13. U __android_log_print
14. U asprintf
15. 000000000009b648 B __bss_end__
16. 000000000009b648 B _bss_end__
17. 0000000000088158 B __bss_start
18. 0000000000088158 B __bss_start__
19. U btowc
20. U _ctype_
21. U __ctype_get_mb_cur_max
22. 0000000000012c04 T __cxa_allocate_dependent_exception
23. 0000000000012a84 T __cxa_allocate_exception
24. U __cxa_atexit
25. 0000000000053c34 T __cxa_bad_cast
26. 0000000000053c68 T __cxa_bad_typeid
27. 0000000000012d8c T __cxa_begin_catch
28. 00000000000139bc T __cxa_call_unexpected
29. 000000000002bf10 T __cxa_current_exception_type
30. 0000000000013eb4 T __cxa_deleted_virtual
31. 000000000002babc T __cxa_demangle
32. 0000000000012e2c T __cxa_end_catch
33. U __cxa_finalize
34. 0000000000012cd4 T __cxa_free_dependent_exception

4817. 00000000000aa490 T _ZNK14CPDF_PageLabel14GetPageByLabelERK15CFX_ByteStringC
4818. 00000000000aa650 T _ZNK14CPDF_PageLabel14GetPageByLabelERK15CFX_WideStringC
4819. 00000000000aa084 T _ZNK14CPDF_PageLabel8GetLabelEi
4820. 00000000000e17a4 T _ZNK14CPDF_PatternCS6GetRGBEPfRfS1_S1_
4821. 00000000000ffd30 T _ZNK14CPDF_StreamAcc7GetDataEv
4822. 00000000000ffd50 T _ZNK14CPDF_StreamAcc7GetSizeEv
4823. 00000000000ea1d8 T _ZNK14CPDF_TextState12GetFontSizeHEv
4824. 00000000000ea184 T _ZNK14CPDF_TextState12GetFontSizeVEv
4825. 00000000000ea258 T _ZNK14CPDF_TextState13GetShearAngleEv
4826. 00000000000ea22c T _ZNK14CPDF_TextState16GetBaselineAngleEv
4827. 0000000000158914 T _ZNK15CFX_ByteStringC5GetIDEi
4828. 000000000018a974 T _ZNK15CFX_FilteredDIB11GetScanlineEi
4829. 000000000018a9cc T _ZNK15CFX_FilteredDIB18DownSampleScanlineEiPhiiii
4830. 000000000015cec0 T _ZNK15CFX_MapPtrToPtr10GetAssocAtEPvRj
4831. 000000000015cf6c T _ZNK15CFX_MapPtrToPtr10GetValueAtEPv
4832. 000000000015cddc T _ZNK15CFX_MapPtrToPtr12GetNextAssocERPvS1_S1_
4833. 000000000015cf38 T _ZNK15CFX_MapPtrToPtr6LookupEPvRS0_
4834. 000000000015cdd4 T _ZNK15CFX_MapPtrToPtr7HashKeyEPv
4835. 000000000015fa40 T _ZNK15CFXMEM_FixedMgr7GetSizeEPv
4836. 0000000000163b44 T _ZNK15CFX_WideStringL10GetIntegerEv
4837. 0000000000163b54 T _ZNK15CFX_WideStringL8GetFloatEv
4838. 0000000000159990 T _ZNK15CFX_WideTextBuf13GetWideStringEv
4839. 00000000001599d4 T _ZNK15CFX_WideTextBuf14GetWideStringLER15CFX_WideStringL
4840. 00000000000e3cf8 T _ZNK15CPDF_ColorSpace10GetBufSizeEv
4841. 00000000000e3f7c T _ZNK15CPDF_ColorSpace11GetMaxIndexEv
4842. 00000000000e3ef0 T _ZNK15CPDF_ColorSpace15GetDefaultColorEPf
4843. 00000000000e4b14 T _ZNK15CPDF_ColorSpace18TranslateImageLineEPhPKhiiii
4844. 00000000000e3d5c T _ZNK15CPDF_ColorSpace4sRGBEv
4845. 00000000000e3d98 T _ZNK15CPDF_ColorSpace7GetCMYKEPfRfS1_S1_S1_
4846. 00000000000e3e54 T _ZNK15CPDF_ColorSpace7SetCMYKEPfffff
4847. 000000000010093c T _ZNK15CPDF_Dictionary10GetBooleanERK15CFX_ByteStringCi
4848. 00000000000fecf0 T _ZNK15CPDF_Dictionary10GetElementERK15CFX_ByteStringC
4849. 00000000001008b0 T _ZNK15CPDF_Dictionary10GetIntegerERK15CFX_ByteStringC
4850. 00000000001008e8 T _ZNK15CPDF_Dictionary10GetIntegerERK15CFX_ByteStringCi
4851. 00000000000fecb8 T _ZNK15CPDF_Dictionary11GetStartPosEv
4852. 0000000000100448 T _ZNK15CPDF_Dictionary14GetConstStringERK15CFX_ByteStringC
4853. 00000000001004a4 T _ZNK15CPDF_Dictionary14GetConstStringERK15CFX_ByteStringCS2_
4854. 00000000000fecc0 T _ZNK15CPDF_Dictionary14GetNextElementERPvR14CFX_ByteString
4855. 0000000000100c9c T _ZNK15CPDF_Dictionary14GetUnicodeTextERK15CFX_ByteStringCP11CFX_CharMap
4856. 0000000000100af8 T _ZNK15CPDF_Dictionary15GetElementValueERK15CFX_ByteStringC
4857. 0000000000100b24 T _ZNK15CPDF_Dictionary7GetDictERK15CFX_ByteStringC
4858. 0000000000100b84 T _ZNK15CPDF_Dictionary7GetRectERK15CFX_ByteStringC
4859. 0000000000100b60 T _ZNK15CPDF_Dictionary8GetArrayERK15CFX_ByteStringC
4860. 00000000000fed18 T _ZNK15CPDF_Dictionary8KeyExistERK15CFX_ByteStringC
4861. 0000000000100c28 T _ZNK15CPDF_Dictionary9GetMatrixERK15CFX_ByteStringC
4862. 000000000010079c T _ZNK15CPDF_Dictionary9GetNumberERK15CFX_ByteStringC
4863. 0000000000100c78 T _ZNK15CPDF_Dictionary9GetStreamERK15CFX_ByteStringC
4864. 0000000000100270 T _ZNK15CPDF_Dictionary9GetStringERK15CFX_ByteStringC
4865. 00000000001002c8 T _ZNK15CPDF_Dictionary9GetStringERK15CFX_ByteStringCS2_
4866. 0000000000100f10 T _ZNK15CPDF_Dictionary9IdenticalEPS_
4867. 00000000000e6a6c T _ZNK15CPDF_ExpIntFunc6v_CallEPfS0_
4868. 00000000000e25d0 T _ZNK15CPDF_ICCBasedCS18TranslateImageLineEPhPKhiiii
4869. 00000000000e24bc T _ZNK15CPDF_ICCBasedCS6GetRGBEPfRfS1_S1_
4870. 00000000000e1794 T _ZNK15CPDF_ICCBasedCS6SetRGBEPfffff
4871. 00000000000e230c T _ZNK15CPDF_ICCBasedCS9v_GetCMYKEPfRfS1_S1_S1_
4872. 00000000000df380 T _ZNK15CPDF_PageObject5CloneEv
4873. 00000000000ddcfc T _ZNK15CPDF_PageObject7GetBBoxEPK10CFX_Matrix
4874. 00000000000d26e4 T _ZNK15CPDF_SimpleFont19IsUnicodeCompatibleEv
4875. 00000000000e8ec8 T _ZNK15CPDF_StitchFunc6v_CallEPfS0_
4876. 00000000000ddec4 T _ZNK15CPDF_TextObject10CountCharsEv
4877. 00000000000dee78 T _ZNK15CPDF_TextObject11CalcCharPosEPf
4878. 00000000000ddfb4 T _ZNK15CPDF_TextObject11GetCharInfoEiP19CPDF_TextObjectItem
4879. 00000000000ddf14 T _ZNK15CPDF_TextObject11GetCharInfoEiRjRf
4880. 00000000000de48c T _ZNK15CPDF_TextObject11GetCharRectEiR13CFX_FloatRect
4881. 00000000000ddda8 T _ZNK15CPDF_TextObject11GetItemInfoEiP19CPDF_TextObjectItem
4882. 00000000000de314 T _ZNK15CPDF_TextObject12GetCharWidthEj

4883. 00000000000de01c T _ZNK15CPDF_TextObject13GetTextMatrixEP10CFX_Matrix
4884. 00000000000de3c0 T _ZNK15CPDF_TextObject17GetSpaceCharWidthEv
4885. 0000000000193080 T _ZNK16CFX_RenderDevice13GetDeviceCapsEi
4886. 0000000000193114 T _ZNK16CFX_RenderDevice22CreateCompatibleBitmapEP12CFX_DIBitmapii
4887. 00000000001930a4 T _ZNK16CFX_RenderDevice6GetCTMEv
4888. 00000000000eba0c T _ZNK16CPDF_ContentMark10LookupMarkERK15CFX_ByteStringCRP15CPDF_Dictionary
4889. 00000000000eb97c T _ZNK16CPDF_ContentMark7HasMarkERK15CFX_ByteStringC
4890. 000000000012427c T _ZNK16CPDF_LinkExtract10CountLinksEv
4891. 0000000000124bb0 T _ZNK16CPDF_LinkExtract17GetBoundedSegmentEiRiS0_
4892. 0000000000124990 T _ZNK16CPDF_LinkExtract6GetURLEi
4893. 0000000000124be8 T _ZNK16CPDF_LinkExtract8GetRectsEiR17CFX_ArrayTemplateI13CFX_FloatRectE
4894. 00000000000df694 T _ZNK16CPDF_PageObjects14GetObjectIndexEP15CPDF_PageObject
4895. 00000000000df744 T _ZNK16CPDF_PageObjects15CalcBoundingBoxEv
4896. 00000000000df6dc T _ZNK16CPDF_PageObjects16GetObjectByIndexEi
4897. 00000000000df65c T _ZNK16CPDF_PageObjects21EstimateParseProgressEv
4898. 00000000000e70d4 T _ZNK16CPDF_SampledFunc6v_CallEPfS0_
4899. 00000000000a0df8 T _ZNK17CPDF_ActionFields12GetAllFieldsER17CFX_ArrayTemplateIPvE
4900. 00000000000a0cd4 T _ZNK17CPDF_ActionFields14GetFieldsCountEv
4901. 00000000000a0fe4 T _ZNK17CPDF_ActionFields8GetFieldEj
4902. 00000000000a18fc T _ZNK17CPDF_DocJSActions11GetJSActionEiR14CFX_ByteString
4903. 00000000000a1988 T _ZNK17CPDF_DocJSActions11GetJSActionERK14CFX_ByteString
4904. 00000000000a1a00 T _ZNK17CPDF_DocJSActions12FindJSActionERK14CFX_ByteString
4905. 00000000000a18b4 T _ZNK17CPDF_DocJSActions14CountJSActionsEv
4906. 00000000000d635c T _ZNK17CPDF_FontEncoding11IsIdenticalEPS_
4907. 00000000000d62cc T _ZNK17CPDF_FontEncoding19CharCodeFromUnicodeEw
4908. 0000000000110b3c T _ZNK17CPDF_RenderStatus11GetFillArgbEPK15CPDF_PageObjecti
4909. 0000000000110c40 T _ZNK17CPDF_RenderStatus13GetStrokeArgbEPK15CPDF_PageObject
4910. 000000000010ff48 T _ZNK17CPDF_RenderStatus15GetScaledMatrixER10CFX_Matrix
4911. 0000000000110b0c T _ZNK17CPDF_RenderStatus15GetTransferFuncEP11CPDF_Object
4912. 000000000010fd58 T _ZNK17CPDF_RenderStatus20GetObjectClippedRectEPK15CPDF_PageObjectPK10CFX_MatrixiR
7FX_RECT
4913. 00000000000e474c T _ZNK17CPDF_SeparationCS6GetRGBEPfRfS1_S1_
4914. 000000000012b148 T _ZNK17CPDF_TextPageFind11GetCurOrderEv
4915. 000000000012b124 T _ZNK17CPDF_TextPageFind12GetCharIndexEi
4916. 0000000000124a04 T _ZNK17CPDF_TextPageFind12GetRectArrayER17CFX_ArrayTemplateI13CFX_FloatRectE
4917. 000000000012b150 T _ZNK17CPDF_TextPageFind15GetMatchedCountEv
4918. 00000000000bdcb8 T _ZNK17CPDF_VariableText13GetTotalWordsEv
4919. 00000000000bd4ac T _ZNK17CPDF_VariableText14GetContentRectEv
4920. 00000000000c24bc T _ZNK17CPDF_VariableText14GetUpWordPlaceERK14CPVT_WordPlaceRK10CPDF_Point
4921. 00000000000bdaa8 T _ZNK17CPDF_VariableText15AjustLineHeaderERK14CPVT_WordPlacei
4922. 00000000000be354 T _ZNK17CPDF_VariableText15GetLineEndPlaceERK14CPVT_WordPlace
4923. 00000000000c2070 T _ZNK17CPDF_VariableText15SearchWordPlaceERK10CPDF_Point
4924. 00000000000bf710 T _ZNK17CPDF_VariableText15UpdateWordPlaceER14CPVT_WordPlace
4925. 00000000000c25d0 T _ZNK17CPDF_VariableText16GetDownWordPlaceERK14CPVT_WordPlaceRK10CPDF_Point
4926. 00000000000bf460 T _ZNK17CPDF_VariableText16GetNextWordPlaceERK14CPVT_WordPlace
4927. 00000000000bf150 T _ZNK17CPDF_VariableText16GetPrevWordPlaceERK14CPVT_WordPlace
4928. 00000000000bd464 T _ZNK17CPDF_VariableText17GetLineBeginPlaceERK14CPVT_WordPlace
4929. 00000000000beea4 T _ZNK17CPDF_VariableText18GetSectionEndPlaceERK14CPVT_WordPlace
4930. 00000000000bd48c T _ZNK17CPDF_VariableText20GetSectionBeginPlaceERK14CPVT_WordPlace
4931. 00000000000bf860 T _ZNK17CPDF_VariableText20WordIndexToWordPlaceEi
4932. 00000000000bdd00 T _ZNK17CPDF_VariableText20WordPlaceToWordIndexERK14CPVT_WordPlace
4933. 000000000010f80c T _ZNK18CPDF_RenderOptions14TranslateColorEj
4934. 000000000015dea0 T _ZNK20CFX_CMapDWordToDWord12GetNextAssocERPvRjS2_
4935. 000000000015de8c T _ZNK20CFX_CMapDWordToDWord16GetStartPositionEv
4936. 000000000015de38 T _ZNK20CFX_CMapDWordToDWord6LookupEjRj
4937. 00000000000eb7c8 T _ZNK20CPDF_ContentMarkData7GetMCIDEv
4938. 0000000000145d4 T _ZNK20CPDF_DIBTransferFunc17TranslateScanlineEPhPKh
4939. 0000000000114984 T _ZNK20CPDF_DIBTransferFunc20TranslateDownSamplesEPhPKhii
4940. 00000000001019d0 T _ZNK20CPDF_IndirectObjects13GetLastObjNumEv
4941. 000000000011389c T _ZNK20CPDF_PageRenderCache13GetCachedSizeEP11CPDF_Stream
4942. 0000000000155d30 T _ZNK22CFX_BaseSegmentedArray12IterateIndexEiRiPPvPFiS1_S1_ES1_
4943. 0000000000155c9c T _ZNK22CFX_BaseSegmentedArray14IterateSegmentEPKhiPFiPvS2_ES2_
4944. 0000000000155e44 T _ZNK22CFX_BaseSegmentedArray5GetAtEi
4945. 0000000000155e0c T _ZNK22CFX_BaseSegmentedArray7IterateEPFiPvS0_ES0_
4946. 0000000000155c20 T _ZNK22CFX_BaseSegmentedArray8GetIndexEi
4947. 000000000015d574 T _ZNK22CFX_MapByteStringToPtr10GetAssocAtERK15CFX_ByteStringCRj

Severity
Low

VULNERABILITY

App allowed to run in a rooted device

OWASP MASVS	Common Weakness Enumeration
8.1 R	-
Known Exploits	**Common Vulnerability Scoring System**
CVE-2017-4896	CVE-2017-4896-CVSS 3.0 Score 3.8

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#testing-root-detection-mstg-resilience-1

Reference URL[s]:

https://www.howtogeek.com/115297/how-to-root-your-android-why-you-might-want-to/
https://insights.samsung.com/2015/10/12/is-rooting-your-phone-safe-the-security-risks-of-rooting-devices/
https://en.wikipedia.org/wiki/Superuser
https://www.educba.com/rooting-android/

CVSS BaseScore and Vector

CVE-2017-4896
Version & Base Score : CVSS 3.0 Score 3.8
CVSS Scoring Vector : CVSS:3.0/AV:L/AC:L/PR:L/UI:N/S:C/C:L/I:N/A:N

THREAT

Android is a multi-user Linux-based operating system in which each app is a different user. By default, the system assigns to each app a unique Linux user ID which remains unknown to the app itself. The system sets permissions for all the files in an app so that only the user ID assigned to that app can access them

Each process has its own virtual machine [VM] and every app runs in its own Linux process, so each Android app lives in its own security sandbox. The Android system starts the process when any of the app's components need to be executed and then shuts down the process when it is no longer needed or when the system must recover memory for other apps

Root is also a user. The difference is that the root user [or superuser] has the permission to do anything to any file in any place in the system

RISK

Running apps on a rooted device means that they have full privileges and - thus - are no longer confined in their sandboxes. Hence, a malicious app can be given the possibility to gain access to personal information such as contact lists, emails and other data or to collect sensitive data like credentials and passwords

This can become even more serious with corporate resources. In this case a hacker can gain entry to corporate resources when the phone connects into the secure network or through corporate applications on the device

FIX

Quixxi App Shield can automatically forbid the app to be executed in a rooted device by enabling the "Terminate the app when running in rooted device" Shield entry

If you want/need to add an exception for those customers who are running the official app [i.e. downloaded from Google Play, Samsung or Amazon stores] on a rooted device please be sure to select the "Allow apps installed from Google Play, Samsung and Amazon stores to bypass the root protection" entry. This last point is definitely to be analyzed for paid apps

In scenarios involving sensitive data please recommend the customers to use the stock Android versions or invite them to unroot their phones

Severity
Low

VULNERABILITY

App allowed to run in an emulator

OWASP MASVS	Common Weakness Enumeration
8.5 R	-
Known Exploits	**Common Vulnerability Scoring System**
-	-

Best Practices:

https://github.com/OWASP/owasp-mstg/blob/1.1.3/Document/0x05j-Testing-Resiliency-Against-Reverse-Engineering.md#testing-emulator-detection-mstg-resilience-5
https://github.com/OWASP/owasp-mstg/blob/master/Document/0x05c-Reverse-Engineering-and-Tampering.md

Reference URL[s]:

https://developer.android.com/tools/help/emulator.html

CVSS BaseScore and Vector

-

THREAT

An emulator activity outside the development process is a good indication that someone other than you is trying to analyse the app

RISK

When an app runs in an emulator an attacker can dynamically analyse it, access its sensitive data and steal its intellectual property

FIX

Quixxi App Shield can automatically forbid the app to run into an emulator by enabling the "Terminate the app when connected to the emulator" Shield option

Severity
Low

VULNERABILITY

Missing check for the download source

OWASP MASVS	Common Weakness Enumeration
7.1	CWE-610
Known Exploits	**Common Vulnerability Scoring System**
CVE-2018-9582	CVE-2018-9582-CVSS 3.0 Score 7.8

Best Practices:

http://www.securityfocus.com/bid/1064
https://source.android.com/security/bulletin/2019-01-01.html