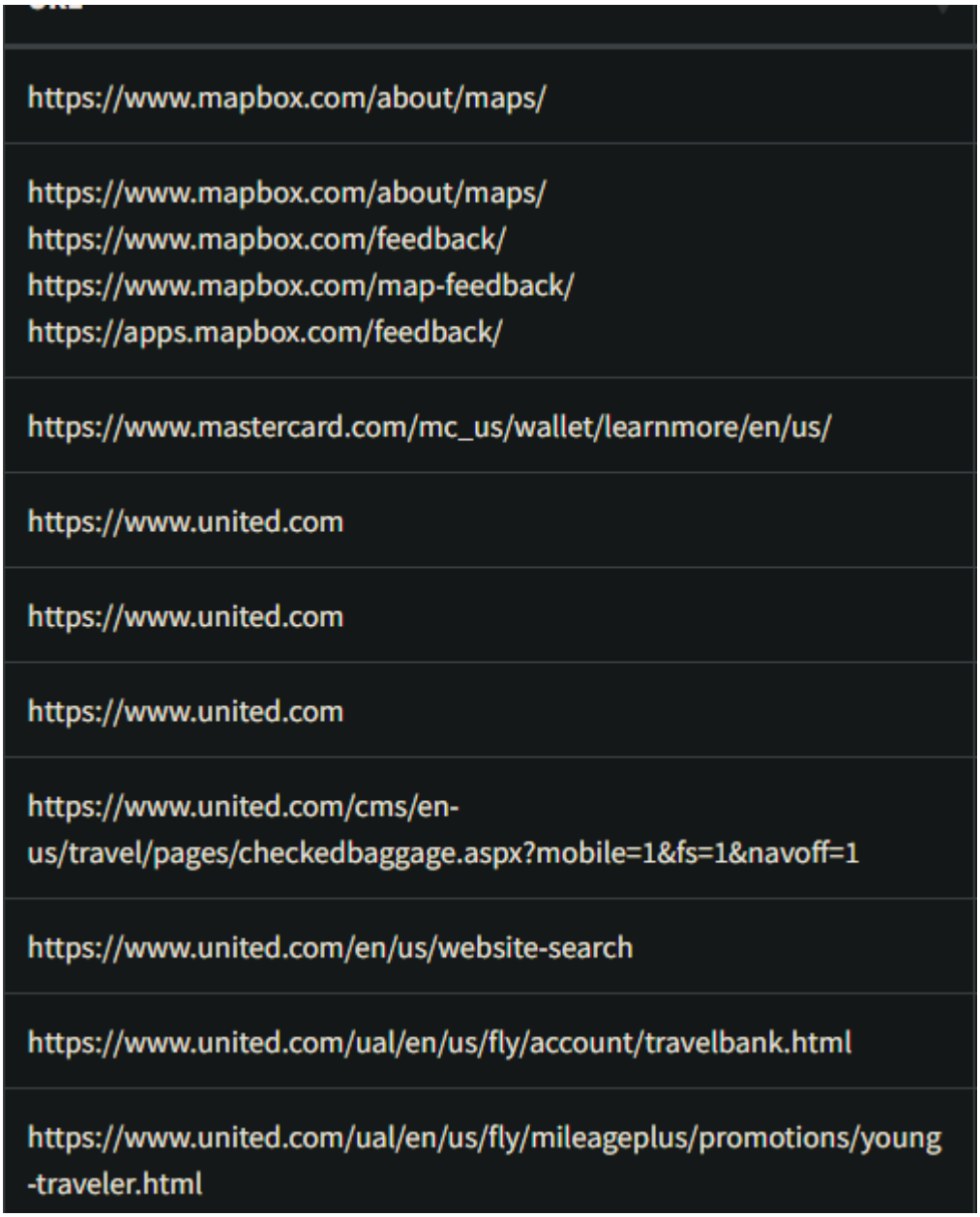
MASVS-AUTH-1:

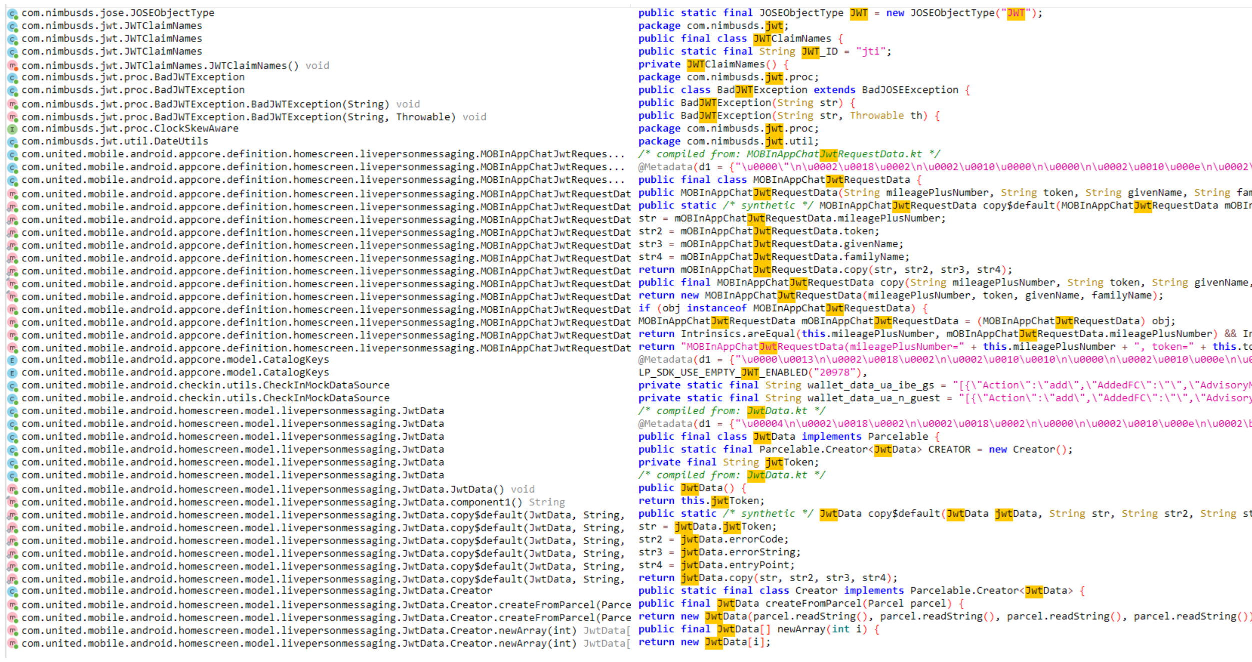
- There is over thousands of url's being used in the app. The vast majority of which are using https. Many of the URL's appear that do us https are at least appropriately used:



This photo shows urls relating to maps/location, mastercard, and young travelers.

- The app does not block additional login attempts past 10.

- Searching token in jadx in mobsf resulted little to nothing of interest, however JWT resulted in finding many relevant mentions of JWT and its role in receiving data about the user especially with the milesplus number one of the credentials used to login:



```
com.nimbusds.jose.JOSEObjectType    public static final JOSEObjectType JWT = new JOSEObjectType("JWT");
com.nimbusds.jwt.JWTClaimNames    package com.nimbusds.jwt;
com.nimbusds.jwt.JWTClaimNames    public final class JWTClaimNames {
com.nimbusds.jwt.JWTClaimNames    public static final String JWT_ID = "jti";
com.nimbusds.jwt.JWTClaimNames.JWTClaimNames() void    private JWTClaimNames() {
com.nimbusds.jwt.proc.BadJWTException    package com.nimbusds.jwt.proc;
com.nimbusds.jwt.proc.BadJWTException    public class BadJWTException extends BadJOSEException {
com.nimbusds.jwt.proc.BadJWTException.BadJWTException(String) void    public BadJWTException(String str) {
com.nimbusds.jwt.proc.BadJWTException.BadJWTException(String, Throwable) void    public BadJWTException(String str, Throwable th) {
com.nimbusds.jwt.proc.ClockSkewAware    package com.nimbusds.jwt.proc;
com.nimbusds.jwt.util.DateUtils    package com.nimbusds.jwt.util;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtReques...    /* compiled from: MOBInAppChatJwtRequestData.kt */
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtReques...    @Metadata(d1 = {"\u0000\"\n\u0002\u0018\u0002\n\u0002\u0010\u0000\n\u0000\n\u0002\u0010\u000e\n\u0002\
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtReques...    public final class MOBInAppChatJwtRequestData {
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    public MOBInAppChatJwtRequestData(String mileagePlusNumber, String token, String givenName, String fam
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    public static /* synthetic */ MOBInAppChatJwtRequestData copy$default(MOBInAppChatJwtRequestData mOBIn
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    str = mOBInAppChatJwtRequestData.mileagePlusNumber;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    str2 = mOBInAppChatJwtRequestData.token;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    str3 = mOBInAppChatJwtRequestData.givenName;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    str4 = mOBInAppChatJwtRequestData.familyName;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    return mOBInAppChatJwtRequestData.copy(str, str2, str3, str4);
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    public final MOBInAppChatJwtRequestData copy(String mileagePlusNumber, String token, String givenName,
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    return new MOBInAppChatJwtRequestData(mileagePlusNumber, token, givenName, familyName);
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    if (obj instanceof MOBInAppChatJwtRequestData) {
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    MOBInAppChatJwtRequestData mOBInAppChatJwtRequestData = (MOBInAppChatJwtRequestData) obj;
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    return Intrinsics.areEqual(this.mileagePlusNumber, mOBInAppChatJwtRequestData.mileagePlusNumber) && In
com.united.mobile.android.appcore.definition.homescreen.livepersonmessaging.MOBInAppChatJwtRequestDat    return "MOBInAppChatJwtRequestData(mileagePlusNumber=" + this.mileagePlusNumber + ", token=" + this.to
com.united.mobile.android.appcore.model.CatalogKeys    @Metadata(d1 = {"\u0000\u0013\n\u0002\u0018\u0002\n\u0002\u0010\u0010\n\u0000\n\u0002\u0010\u000e\n\u0
com.united.mobile.android.appcore.model.CatalogKeys    LP_SDK_USE_EMPTY_JWT_ENABLED("20978"),
com.united.mobile.android.checkin.utils.CheckInMockDataSource    private static final String wallet_data_ua_ibe_gs = "[{\"Action\":\"add\",\"AddedFC\":\"\",\"AdvisoryM
com.united.mobile.android.checkin.utils.CheckInMockDataSource    private static final String wallet_data_ua_n_guest = "[{\"Action\":\"add\",\"AddedFC\":\"\",\"Advisory
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    /* compiled from: JwtData.kt */
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    @Metadata(d1 = {"\u00004\n\u0002\u0018\u0002\n\u0002\u0018\u0002\n\u0000\n\u0002\u0010\u000e\n\u0002\b
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    public final class JwtData implements Parcelable {
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    public static final Parcelable.Creator<JwtData> CREATOR = new Creator();
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    private final String jwtToken;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData    /* compiled from: JwtData.kt */
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.JwtData() void    public JwtData() {
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.component1() String    return this.jwtToken;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    public static /* synthetic */ JwtData copy$default(JwtData jwtData, String str, String str2, String st
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    str = jwtData.jwtToken;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    str2 = jwtData.errorCode;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    str3 = jwtData.errorString;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    str4 = jwtData.entryPoint;
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.copy$default(JwtData, String,    return jwtData.copy(str, str2, str3, str4);
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.Creator    public static final class Creator implements Parcelable.Creator<JwtData> {
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.Creator.createFromParcel(Parce    public final JwtData createFromParcel(Parcel parcel) {
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.Creator.createFromParcel(Parce    return new JwtData(parcel.readString(), parcel.readString(), parcel.readString(), parcel.readString())
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.Creator.newArray(int) JwtData[    public final JwtData[] newArray(int i) {
com.united.mobile.android.homescreen.model.livepersonmessaging.JwtData.Creator.newArray(int) JwtData[    return new JwtData[i];
```

- Strong Password Policy Exists:

Password

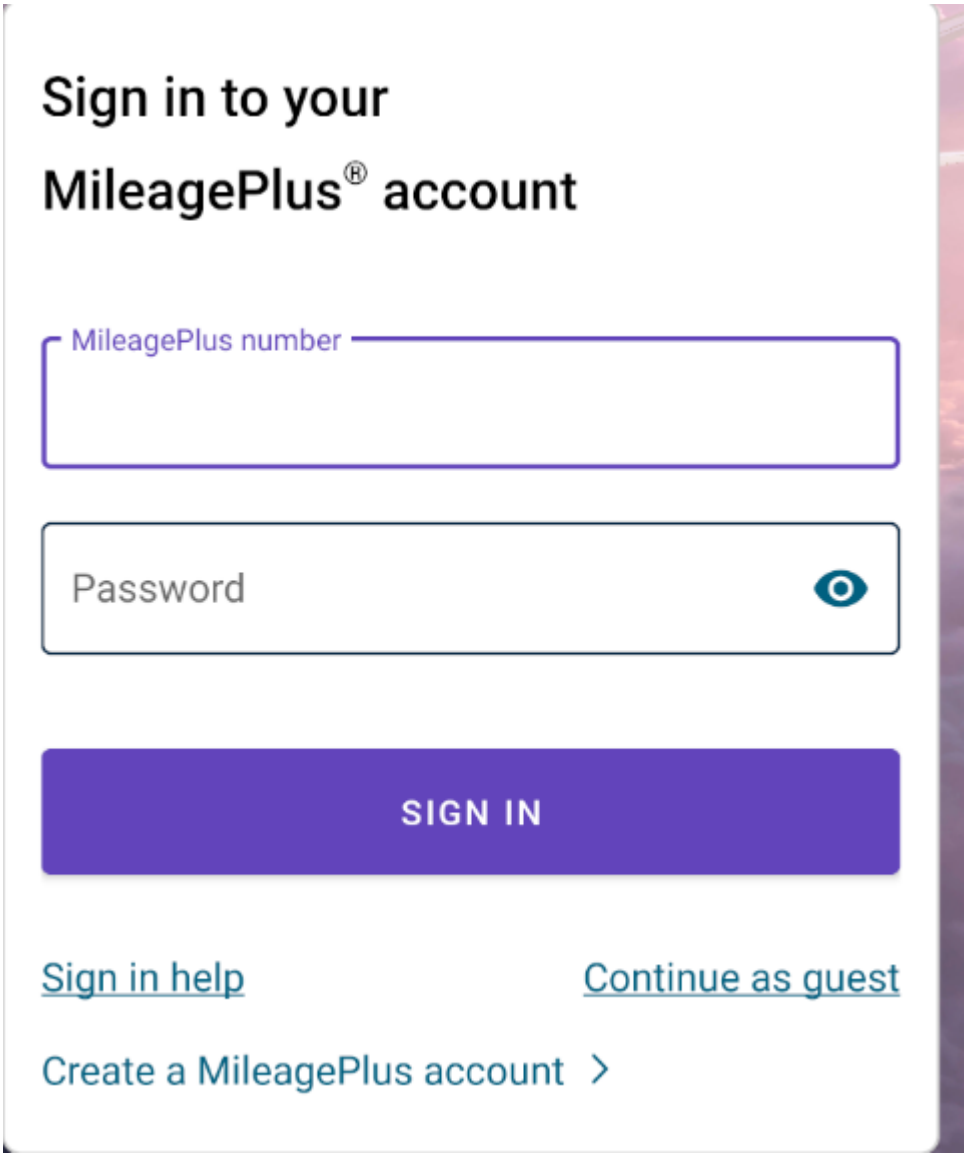
To protect the security of your MileagePlus account, please create a unique password that meets the requirements listed below.

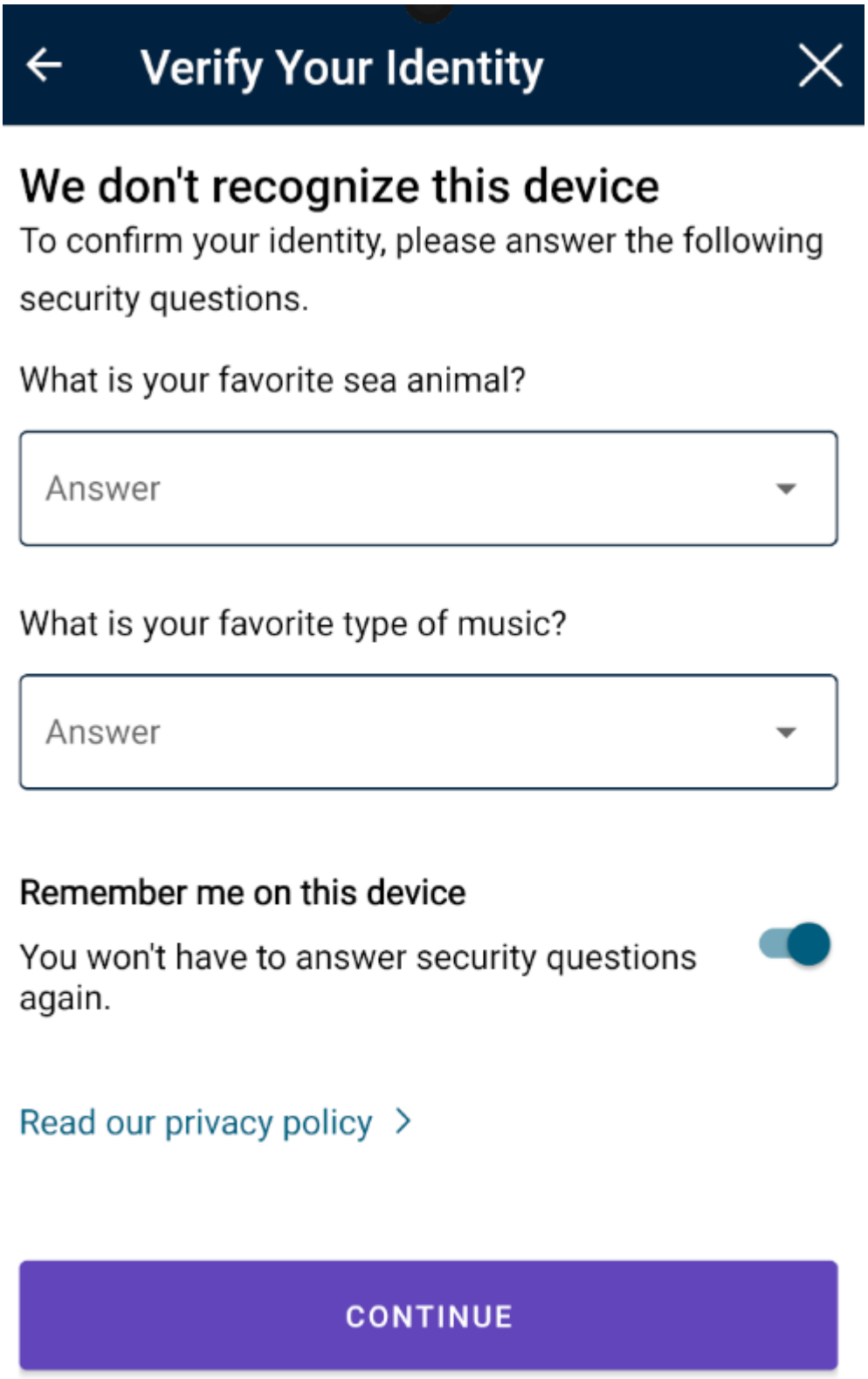
Password requirements

- Must be at least 8 characters in length, with a maximum of 32 characters.
- Must include at least one letter and one number.
- Standard special characters (such as "!" "&" and "+") are allowed.
- Password is case sensitive.
- Cannot include your email address, MileagePlus number or MileagePlus username.

In addition 5 Security questions must be created.

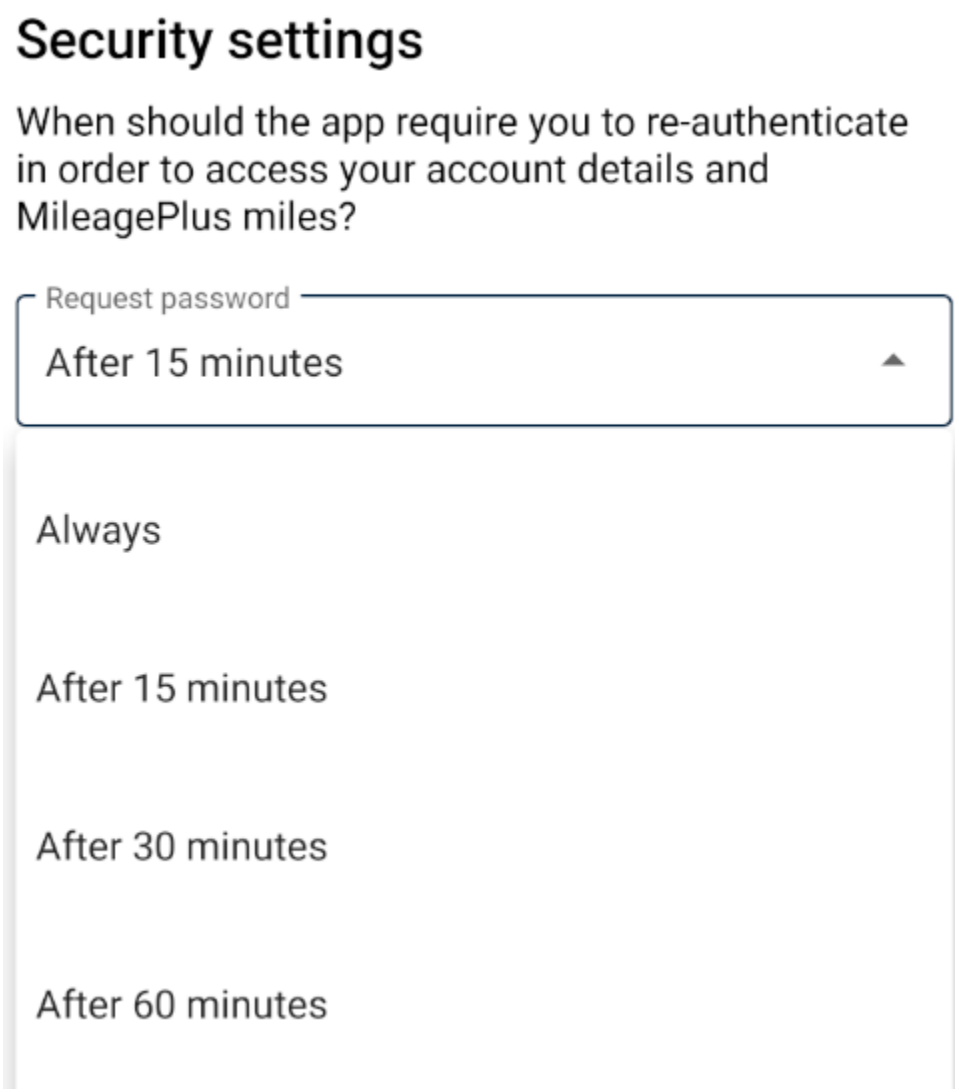
Also to sign in, a mile plus number is used:




Verify Your Identity
We don't recognize this device
To confirm your identity, please answer the following security questions.
What is your favorite sea animal?
Answer
What is your favorite type of music?
Answer
Remember me on this device
You won't have to answer security questions again.
Read our privacy policy
CONTINUE

- After signing on to app for the first time on a new device you can disable security questions.

- The app does have feature to logout user instead to access or to use any action related to sensitive information or the account the user must reauthenticate:



- App logs out safely, and back button does not return to previous session.

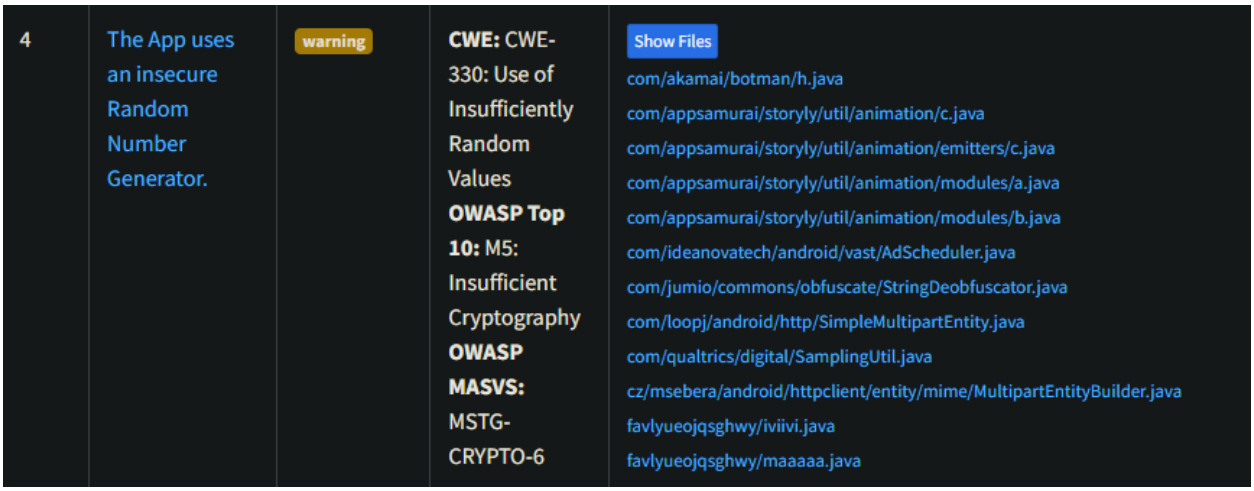
MASVS-AUTH-2:

MASVS-AUTH-3:

There is no 2FA feature for the app, there may be a 2FA feature with SMS. But cannot confirm due to using a burner account.

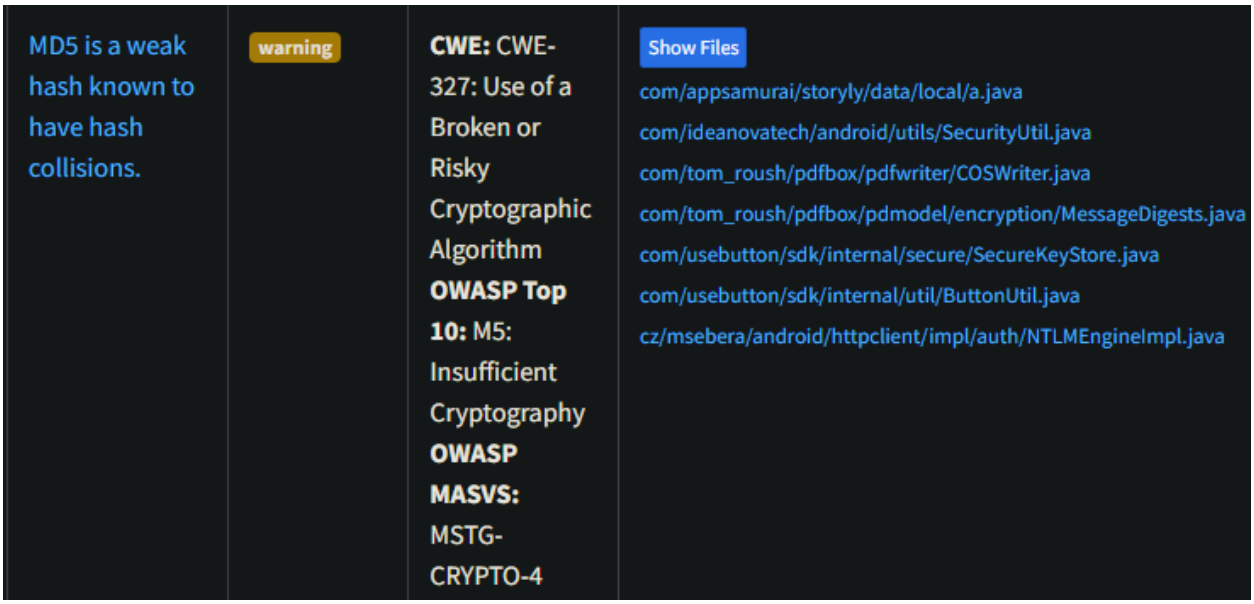
MASVS-CRYPTO-1:

- Weak random number Generator:

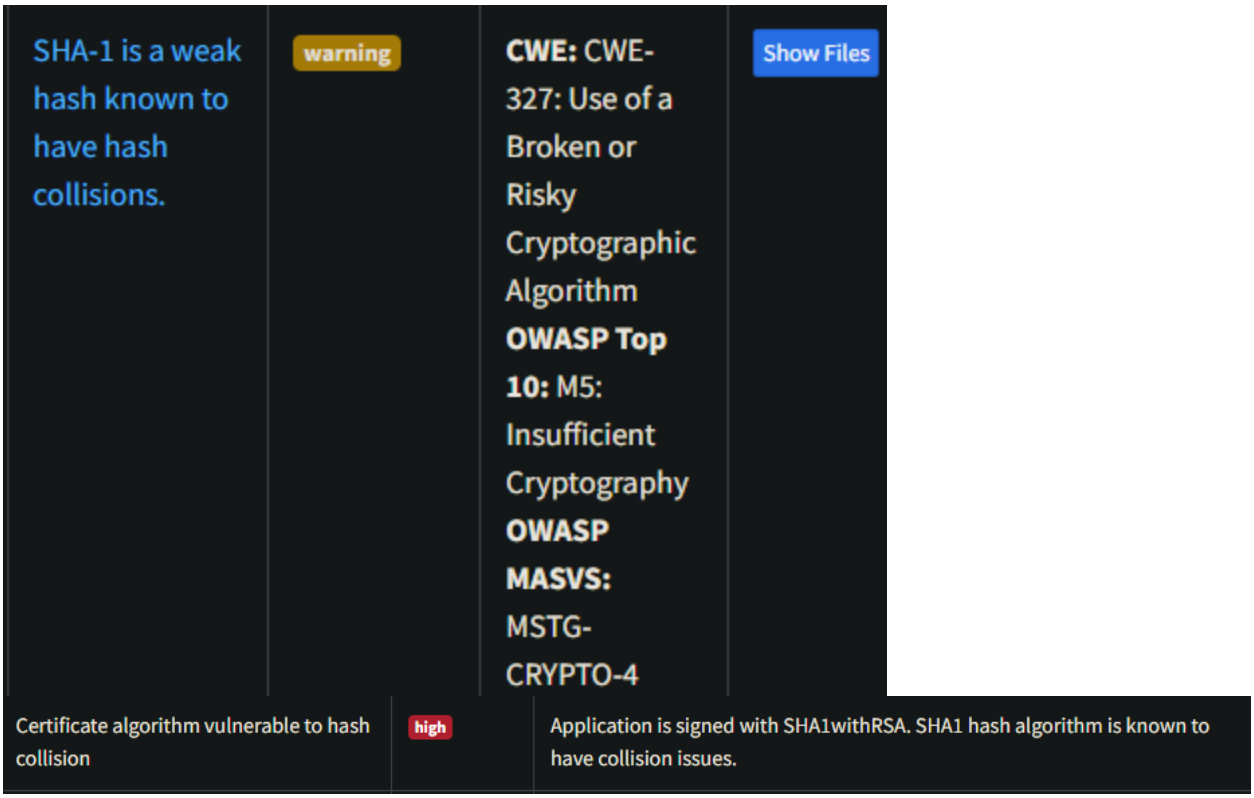


4	The App uses an insecure Random Number Generator.	warning	**CWE:** CWE-330: Use of Insufficiently Random Values **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-6	Show Files com/akamai/botman/h.java com/appsamurai/storyly/util/animation/c.java com/appsamurai/storyly/util/animation/emitters/c.java com/appsamurai/storyly/util/animation/modules/a.java com/appsamurai/storyly/util/animation/modules/b.java com/ideanovatech/android/vast/AdScheduler.java com/jumio/commons/obfuscate/StringDeobfuscator.java com/loopj/android/http/SimpleMultipartEntity.java com/qualtrics/digital/SamplingUtil.java cz/msebera/android/httpclient/entity/mime/MultipartEntityBuilder.java favlyueojqsghwy/iviivi.java favlyueojqsghwy/maaaaa.java

- Insecure Algorithms:



MD5 is a weak hash known to have hash collisions.	warning	**CWE:** CWE-327: Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	Show Files com/appsamurai/storyly/data/local/a.java com/ideanovatech/android/utils/SecurityUtil.java com/tom_roush/pdfbox/pdfwriter/COSWriter.java com/tom_roush/pdfbox/pdmodel/encryption/MessageDigests.java com/usebutton/sdk/internal/secure/SecureKeyStore.java com/usebutton/sdk/internal/util/ButtonUtil.java cz/msebera/android/httpclient/impl/auth/NTLMEngineImpl.java



There is no reports from the NIAP Analysis

There are tens of thousands of hard coded secrets in the MOBSF report, making it difficult to investigate and find legitimate secrets. This will be further investigated with for signs of api keys or secrets with other tools at a later date.

Using JADX there does not appear to be any secrets in the open, with the only thing of any interest is the secret key being used in the AES encryption process:

The app uses AES:

But not DES

MASVS-CRYPTO-2: