Android, iOS and Hybrid Applications

Mobile-Development

Overview

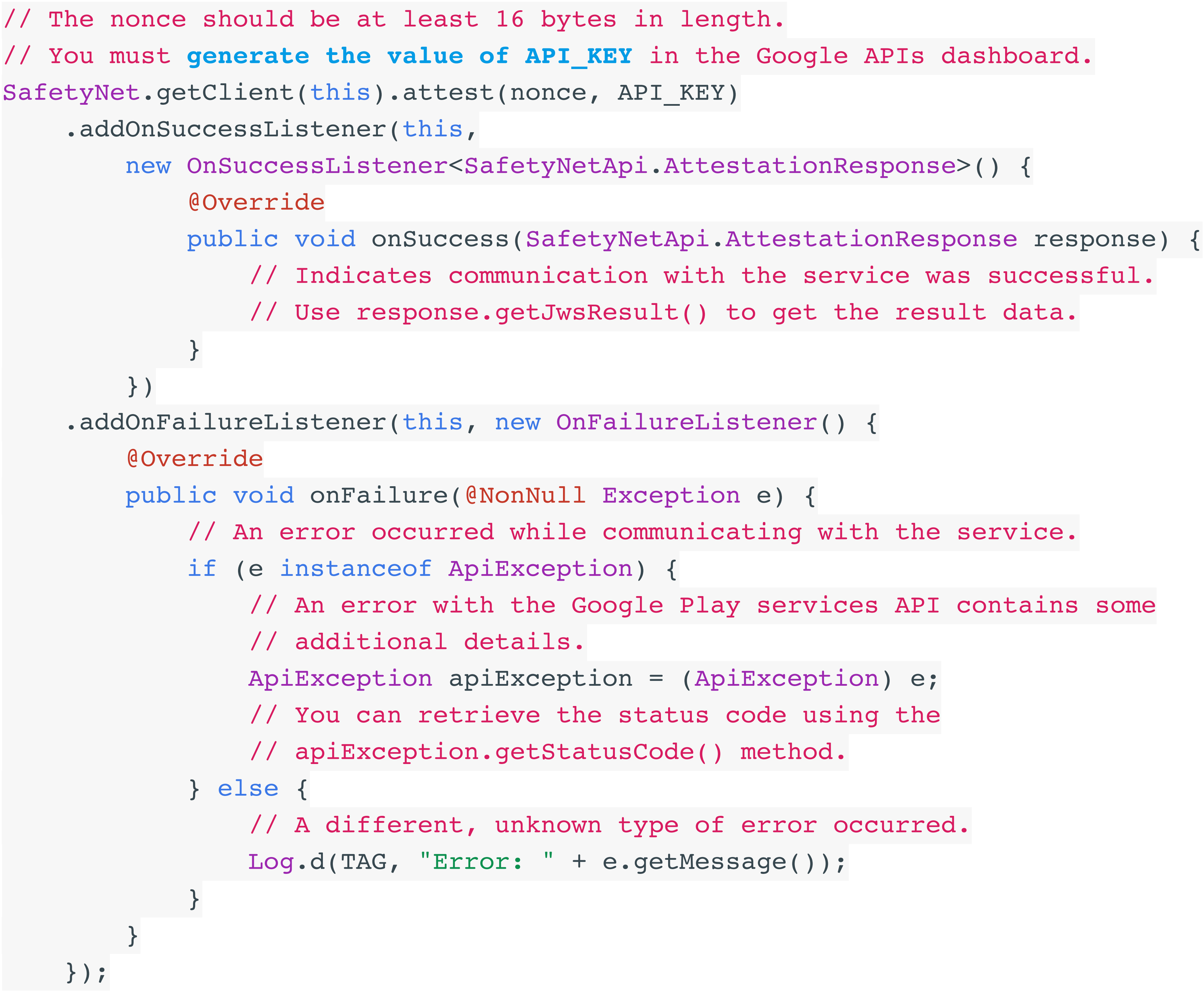
- Root Detection
- (Jailbreak Detection)
- Use a password to encrypt/decrypt user data
- Biometrics

ROOT DETECTION

- Use the official API
- SafetyNet
  - Obtain nonce
  - Send the check request
  - Validate the response

https://developer.android.com/training/safetynet/attestation

ROOT DETECTION



```
// The nonce should be at least 16 bytes in length.
// You must generate the value of API_KEY in the Google APIs dashboard.
SafetyNet.getClient(this).attest(nonce, API_KEY)
    .addOnSuccessListener(this,
        new OnSuccessListener<SafetyNetApi.AttestationResponse>() {
            @Override
            public void onSuccess(SafetyNetApi.AttestationResponse response) {
                // Indicates communication with the service was successful.
                // Use response.getJwsResult() to get the result data.
            }
        })
    .addOnFailureListener(this, new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            // An error occurred while communicating with the service.
            if (e instanceof ApiException) {
                // An error with the Google Play services API contains some
                // additional details.
                ApiException apiException = (ApiException) e;
                // You can retrieve the status code using the
                // apiException.getStatusCode() method.
            } else {
                // A different, unknown type of error occurred.
                Log.d(TAG, "Error: " + e.getMessage());
            }
        }
    });
```

ROOT DETECTION

- Use the nuget package for Xamarin Forms
- Implement it in the device specific project
- Think about what to do as a reaction if the device is rooted

https://www.nuget.org/packages/Xamarin.GooglePlayServices.SafetyNet/

Jailbreak

- You can try or use a library…
- Cat & Mouse game
- This needs to be supported by the device manufacturer to be efficient

NOTES

- We are not going to focus on it
- We don’t write any code for/about it
- But you should remember it for the written exam

Questions?

LOCAL PASSWORD AUTHENTICATION

- Use a Key-Derivation-Function to convert the password

```
public byte[] GenerateKey(string passphrase)
{
  // Number of PBKDF2 hardening rounds to use. Larger values increase
  // computation time. You should select a value that causes computation
  // to take >100ms.
  int iterations = 5000;

  // Generate a 256-bit key
  int outputKeyLength = 256;

  SecretKeyFactory secretKeyFactory = SecretKeyFactory.GetInstance("PBKDF2WithHmacSHA1");
  IKeySpec keySpec = new PBEKeySpec(passphrase.ToCharArray(), _salt.Take(32).ToArray(),
iterations, outputKeyLength);
  ISecretKey secretKey = secretKeyFactory.GenerateSecret(keySpec);

  return secretKey.GetEncoded();
}

public static byte[] _salt = Encoding.UTF8.GetBytes("SuperSalt1234");
```

LOCAL PASSWORD AUTHENTICATION

- Use the Cipher (Android) to en/decrypt

```
public byte[] Encrypt(byte[] input, byte[] key)
{
  var secretKey = new SecretKeySpec(key, "AES");
  var cipher = Cipher.GetInstance("AES");
  cipher.Init(CipherMode.EncryptMode, secretKey);

  // TODO: In production you should generate a random IV and store it somewhere.

  return cipher.DoFinal(input);
}
```

DEPENDENCY SERVICE

- Use the Dependency Service to access it from Forms

```
// Android code

[assembly: Dependency(typeof(PasswordEncryptionService))]


// Shared code

var service = DependencyService.Get<IPasswordEncryptionService>();
var key = service.GenerateKey(Password);
```

QUESTIONS?

WALKTHROUGH

- Android Sample Password
- Implement your authentication screen
- We are going to store the username & password with biometrics afterwards

BIOMETRIC

- Advantages?
  - User tend to use weak PIN/Patterns for device locks
  - Super convenient
  - Very hard to crack if you don’t know the owner

Biometric Setup - Android

- Make sure you've an Emulator/Device with API 28+
- Set the Target Version of your **Android** project to API 28+

BIOMETRIC SETUP - ANDROID

- Update the permissions
  - Open the AndroidManifest.xml
  - Add the “USE_BIOMETRIC” permission
  - It’s a non sensitive permission (not like GPS for example)

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1"
android:versionName="1.0" package="ch.hfu.todo">
  <!—use-sdk etc. -->
  <uses-permission android:name="android.permission.USE_BIOMETRIC" />
  <!-- application etc. -->
</manifest>
```

https://docs.microsoft.com/en-us/xamarin/android/platform/fingerprint-authentication/

BIOMETRIC SETUP - ANDROID

- Create the interface in the **shared** project

```
public interface IBiometricAuthenticationService
{
  void Authenticate(Action success, Action<string> onError);
}
```

BIOMETRIC SETUP - ANDROID

- Add a static reference to your Activity in the **Android** project **MainActivity** file

```
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
  public MainActivity()
  {
    Activity = this;
  }

  public static Activity Activity { get; private set; }

  // EXCLUDED THE REST OF THE CODE FOR CLARITY
}
```

BIOMETRIC SETUP - ANDROID

- Add the implementation in the **Android** project

```
[assembly: Dependency(typeof(BiometricAuthenticationService))]
namespace Todo.Droid.Security
{
  public class BiometricAuthenticationService : IBiometricAuthenticationService
  {
    public void Authenticate(Action success, Action<string> onError)
    {
      var builder = new BiometricPrompt.Builder(MainActivity.Activity)
        .SetDescription("Authenticate with Biometrics!")
        .SetTitle("TODO BIO")
        .SetNegativeButton("Cancel", MainActivity.Activity.MainExecutor, new CancelClickListener())
        .Build();

      builder.Authenticate(new CancellationSignal(),
                           MainActivity.Activity.MainExecutor,
                           new BiometricAuthenticationCallback(success, onError));
    }
  }
}
```

BIOMETRIC SETUP - ANDROID

- Add the CancelClickListener inside the class we just created

```
public class BiometricAuthenticationService : IBiometricAuthenticationService
{
  // REMOVED CODE FOR CLARITY

  private class CancelClickListener : Java.Lang.Object, IDialogInterfaceOnClickListener
  {
    public void OnClick(IDialogInterface dialog, int which)
    {
      // NOP
    }
  }
}
```

BIOMETRIC SETUP - ANDROID

- Add the BiometricAuthenticationCallback inside the same class

```
public class BiometricAuthenticationService : IBiometricAuthenticationService
{
  // REMOVED CODE FOR CLARITY

  private class BiometricAuthenticationCallback : BiometricPrompt.AuthenticationCallback
  {
    public BiometricAuthenticationCallback(Action callback, Action<string> onError)
    {
      _callback = callback;
      _onError = onError;
    }

    public override void OnAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result)
    {
      _callback();
    }

    public override void OnAuthenticationError([GeneratedEnum] BiometricErrorCode errorCode,
                                                               ICharSequence errString)

    {
      _onError(errString.ToString());
    }

    private readonly Action _callback;
    private readonly Action<string> _onError;
  }
}
```

BIOMETRIC SETUP - ANDROID

- Query for biometric authentications somewhere:

```
private void AuthenticateWithBiometrics()
{
  var biometricService = DependencyService.Get<IBiometricAuthenticationService>();

  biometricService.Authenticate(() =>
  {
    // We are authenticated. Do something.
  },
  (error) =>
  {
    // Failed to authenticate
  });
}
```

BIOMETRIC TESTING – ANDROID

- On the Emulator click the three dots on the grey panel
- Open the “Fingerprint” menu on the left in the popup that opened
- On the Emulator go to Settings -> Security -> Fingerprint and setup a fingerprint
- To simulate a “finger touch” click “Touch the Screen” in the grey popup

Questions?

BIOMETRIC – ANDROID

- Walkthrough
- Implement a basic biometric authentication
- In the next step we’re going to extend that example

Biometric Encryption - Android

- The previous method is not ideal
- It’s not really secure that way
- Normally you want to protect sensitive data (username, password, key etc.)

BIOMETRIC ENCRYPTION – ANDROID

- Extend the interface

```
public interface IBiometricAuthenticationService
{
  void Encrypt(byte[] input, Action<byte[]> success, Action<string> error);

  void Decrypt(byte[] input, Action<byte[]> success, Action<string> error);
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Add the BiometricCryptoHelper in the **Android** project

```
public class BiometricCryptoHelper
{
  public BiometricCryptoHelper()
  {
    _keystore = KeyStore.GetInstance(KeyStoreName);
    _keystore.Load(null);

    // TODO For testing we delete the key on every restart.
    if (_keystore.ContainsAlias(KeyAlias))
    {
      _keystore.DeleteEntry(KeyAlias);
    }

    CreateKey();
  }

  public static byte[] IV { get; set; }

  private readonly KeyStore _keystore;
  private const string KeyStoreName = "AndroidKeyStore";
  private const string KeyAlias = "_todoKey";
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Add the key creation logic

```
public class BiometricCryptoHelper
{
  // REMOVED CODE FOR CLARITY

  private void CreateKey()
  {
    KeyGenerator keyGen = KeyGenerator.GetInstance(KeyProperties.KeyAlgorithmAes, KeyStoreName);
    KeyGenParameterSpec keyGenSpec =
    new KeyGenParameterSpec.Builder(KeyAlias, KeyStorePurpose.Encrypt | KeyStorePurpose.Decrypt)
            .SetKeySize(256)
            .SetBlockModes(KeyProperties.BlockModeCbc)
            .SetEncryptionPaddings(KeyProperties.EncryptionPaddingPkcs7)
            .SetUserAuthenticationRequired(true)
            .Build();
    keyGen.Init(keyGenSpec);
    keyGen.GenerateKey();
  }
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Initialise the Cipher

```
public class BiometricCryptoHelper
{
  // REMOVED CODE FOR CLARITY

  private Cipher CreateCipher(CipherMode mode)
  {
    var key = _keystore.GetKey(KeyAlias, null);
    var cipher = Cipher.GetInstance($”{KeyProperties.KeyAlgorithmAes}/{KeyProperties.BlockModeCbc}/
{KeyProperties.EncryptionPaddingPkcs7}");

    try
    {
      if (mode == CipherMode.DecryptMode)
      {
        cipher.Init(mode, key, new IvParameterSpec(IV));
      }
      else
      {
        cipher.Init(mode, key);
      }
    }
    catch (KeyPermanentlyInvalidatedException ex)
    {
      // TODO: The key was invalidated because the Biometric setup changed or a permanent lock out happened.
    }

    return cipher;
  }
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Add the public method to create the CryptoObject

```
public class BiometricCryptoHelper
{
  // REMOVED CODE FOR CLARITY

  public BiometricPrompt.CryptoObject CreateCryptoObject(CipherMode mode)
  {
    var cipher = CreateCipher(mode);

    return new BiometricPrompt.CryptoObject(cipher);
  }
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Adapt the implementation of the service

```
public class BiometricAuthenticationService : IBiometricAuthenticationService
{
  public void Encrypt(byte[] input, Action<byte[]> success, Action<string> error)
  {
    var prompt = BuildPrompt();
    prompt.Authenticate(
      _cryptoHelper.CreateCryptoObject(CipherMode.EncryptMode),
      new CancellationSignal(), MainActivity.Activity.MainExecutor,
      new BiometricEncryptionCallback(input, success, error));
  }

  public void Decrypt(byte[] input, Action<byte[]> success, Action<string> error)
  {
    var prompt = BuildPrompt();
    prompt.Authenticate(
      _cryptoHelper.CreateCryptoObject(CipherMode.DecryptMode),
      new CancellationSignal(), MainActivity.Activity.MainExecutor,
      new BiometricEncryptionCallback(input, success, error));
  }

  private readonly BiometricCryptoHelper _cryptoHelper = new BiometricCryptoHelper();
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Extract the prompt

```
public class BiometricAuthenticationService : IBiometricAuthenticationService
{
   // REMOVED CODE FOR CLARITY

  private BiometricPrompt BuildPrompt()
  {
    return new BiometricPrompt.Builder(MainActivity.Activity)
                      .SetDescription("Authenticate with Biometrics!")
                      .SetTitle("TODO BIO")
                      .SetNegativeButton("Cancel",
                                         MainActivity.Activity.MainExecutor,
                                         new CancelClickListener())
                      .Build();
    }
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Update the Callback

```
private class BiometricEncryptionCallback : BiometricPrompt.AuthenticationCallback
{
  public BiometricEncryptionCallback(byte[] input, Action<byte[]> success, Action<string> error)
  {
    _input = input;
    _success = success;
    _error = error;
  }

  public override void OnAuthenticationSucceeded(BiometricPrompt.AuthenticationResult result)
  {
    if (BiometricCryptoHelper.IV == null)
    {
      BiometricCryptoHelper.IV = result.CryptoObject.Cipher.GetIV();
    }

    _success(result.CryptoObject.Cipher.DoFinal(_input));
  }

  public override void OnAuthenticationError([GeneratedEnum] BiometricErrorCode errorCode, ICharSequence errString)
  {
    _error(errString.ToString());
  }

  private readonly byte[] _input;
  private readonly Action<byte[]> _success;
  private readonly Action<string> _error;
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Encrypt and Decrypt values

```
private void DecryptWithBiometric()
{
  var service = DependencyService.Get<IBiometricAuthenticationService>();
  service.Decrypt(
                  Convert.FromBase64String(TextToEncrypt),
                  (obj) => Output = Encoding.UTF8.GetString(obj),
                  (obj) => Output = "An error occurred");
}

private void EncryptWithBiometric()
{
  var service = DependencyService.Get<IBiometricAuthenticationService>();
  service.Encrypt(
                  Encoding.UTF8.GetBytes(TextToEncrypt),
                  (obj) => Output = Convert.ToBase64String(obj),
                  (obj) => Output = "An error occurred");
}
```

BIOMETRIC ENCRYPTION – ANDROID

- Typical workflow (Low/Medium Security):
  - Authentication with username & password
  - If valid encrypt them with the Biometric Service
  - Store the encrypted values
  - Check if they are there on the next startup
  - Load and decrypt them with the Biometric Service
  - Login with the values

BIOMETRIC ENCRYPTION – ANDROID

- There's an example for simple file handling
  - IFileService and the SimpleFileService
  - Extend them if you need to persist data
  - You can use Newtonsoft to convert from and to JSON

Questions?

BIOMETRIC ENCRYPTION – ANDROID

- Walkthrough
- Extend your app with biometric login

ADDITIONAL TASKS

- Implement SafetyNet for you application
- Add an indirection to your secure storage
    - Think about how to gracefully fallback from Biometrics to Username/Password
    - Encrypt all the user-data that is generated/read with your app
    - Handle edge cases as Key-Invalidation - e.g. fingerprint added