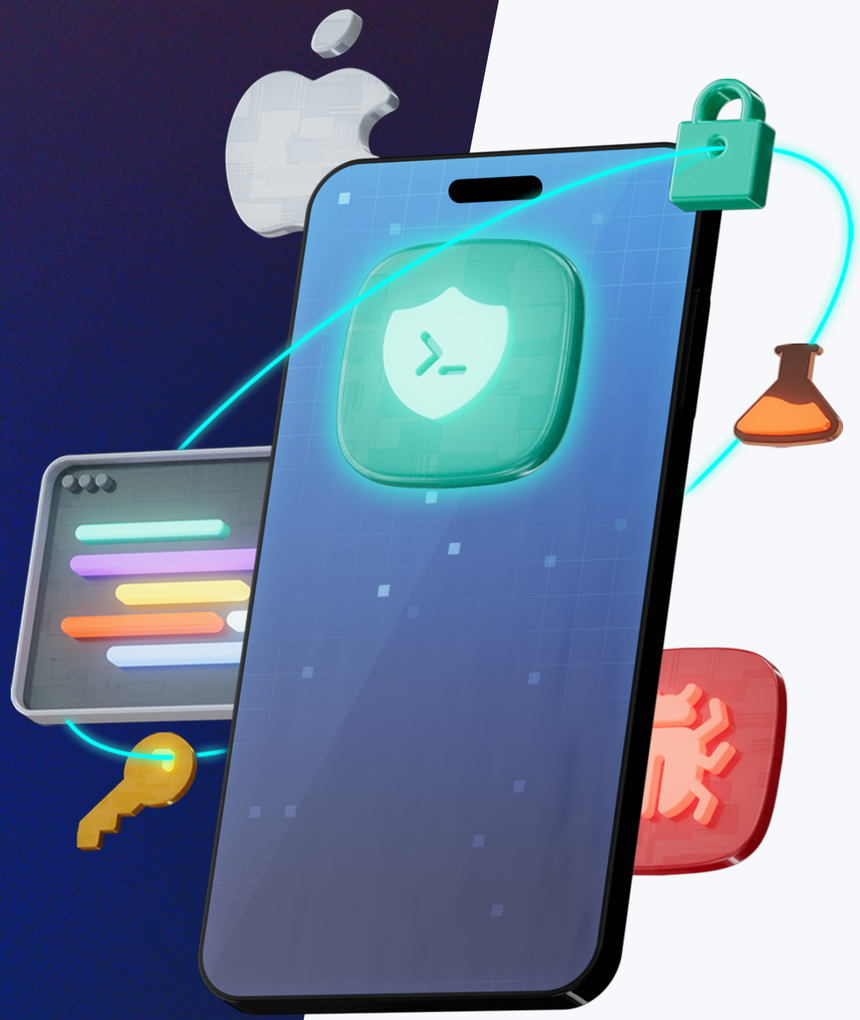
Translating Mobile App Security Lessons to the Flutter Stack



GUARDSQUARE
Mobile application protection

Who am I?

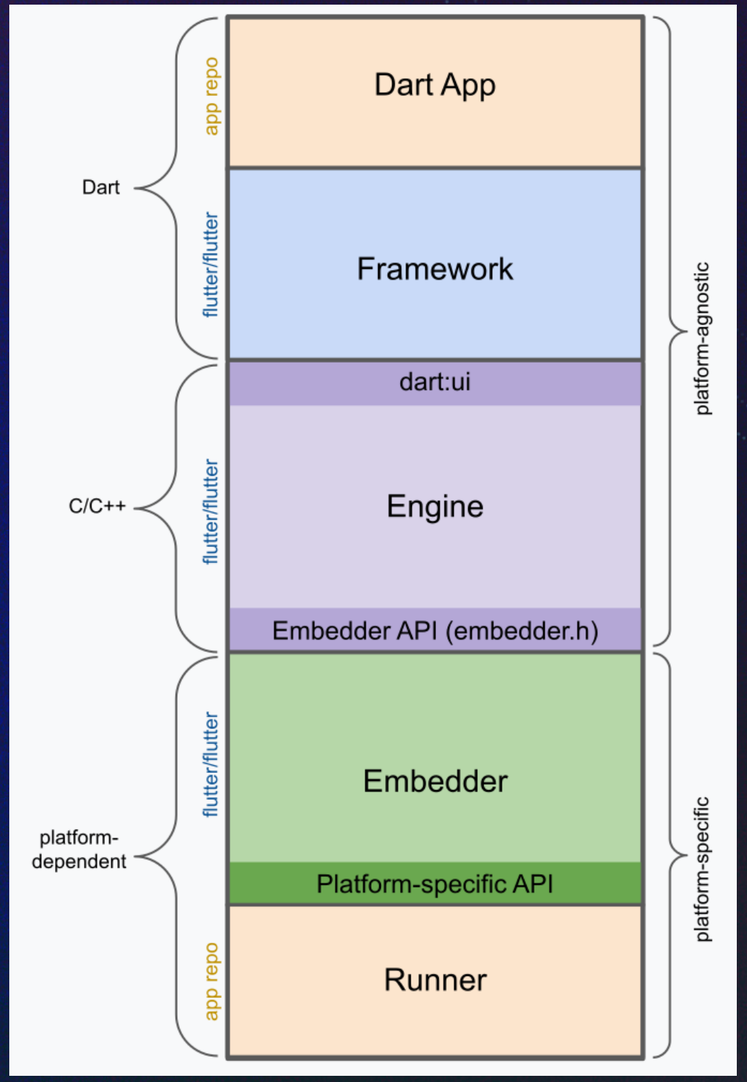
Samuel Hopstock

Software Engineer at Guardsquare, Munich

- **Working on:**
  - App protection: iXGuard
  - App analysis: AppSweep (iOS and Flutter team)
- **Previously worked on:**
  - AppSweep Android

Flutter app structure

```dart
import 'package:flutter/material.dart';

void main() => runApp(const MyApp());

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Hello Flutter')),
        body: const Center(
          child: Text(
            '🦋 Beautiful UIs in one codebase!',
            style: TextStyle(fontSize: 18),
          ),
        ),
      ),
    );
  }
}
```



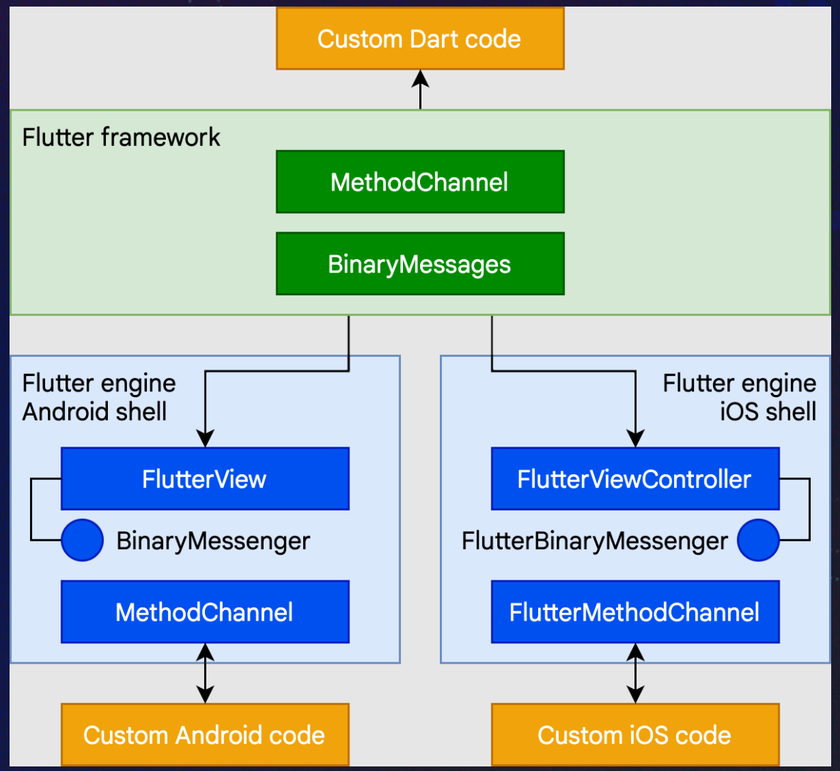
https://docs.flutter.dev/resources/architectural-overview#anatomy-of-an-app

Interaction with native APIs

```dart
// Dart side
const channel = MethodChannel('foo');
final greeting = await channel.invokeMethod('bar', 'world') as String;
print(greeting);
```

```kotlin
// Android (Kotlin)
val channel = MethodChannel(flutterView, "foo")
channel.setMethodCallHandler { call, result ->
  when (call.method) {
    "bar" -> result.success("Hello, ${call.arguments}")
    else -> result.notImplemented()
  }
}
```

```swift
// iOS (Swift)
let channel = FlutterMethodChannel(name: "foo", binaryMessenger: flutterView)
channel.setMethodCallHandler {
  (call: FlutterMethodCall, result: FlutterResult) -> Void in
  switch (call.method) {
    case "bar": result("Hello, \(call.arguments as! String)")
    default: result(FlutterMethodNotImplemented)
  }
}
```



https://docs.flutter.dev/resources/architectural-overview#platform-channels

Security misconceptions

Security misconceptions

- Native binaries = safe binaries

Security misconceptions

- Native binaries = safe binaries
- No Dart decompiler = no threat

Security misconceptions

- Native binaries = safe binaries
- No Dart decompiler = no threat
- --obfuscate = real protection

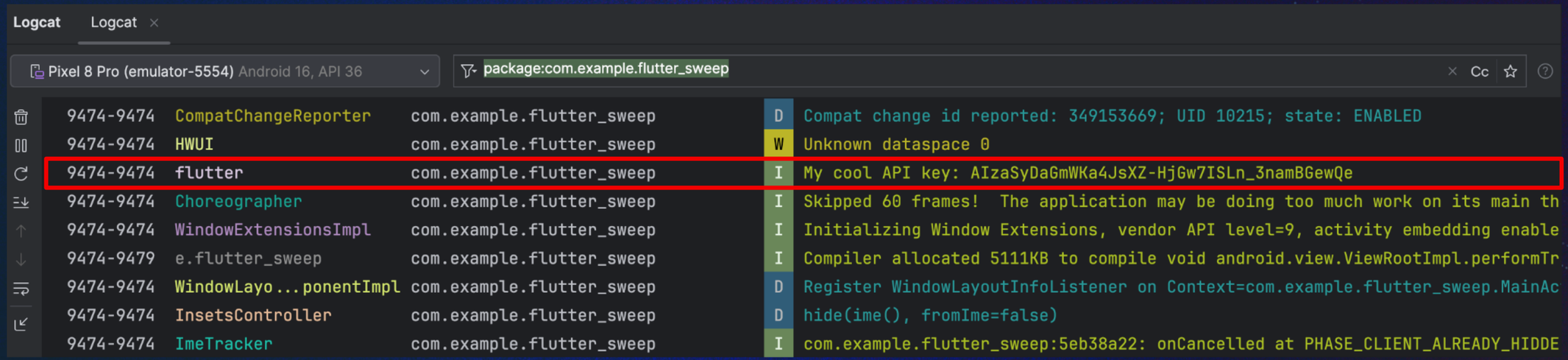
Security misconceptions

- Native binaries = safe binaries
- No Dart decompiler = no threat
- --obfuscate = real protection
- Cross-platform = more abstractions = less vulnerabilities

Surface-level findings

```dart
// sample.dart
void verySecret() {
    print("My cool API key: AIzaSyDaGmWKa4JsXZ-HjGw7ISLn_3namBGewQe");
}

void main() {
  verySecret();
}
```

Verbose logging



Hardcoded secrets

```
$> strings libapp.so

_kDartVmSnapshotInstructions
_kDartIsolateSnapshotInstructions
_kDartVmSnapshotData
_kDartIsolateSnapshotData
_kDartSnapshotBuildId
[...]
My cool API key: AIzaSyDaGmWKa4JsXZ-HjGw7ISLn_3namBGewQe
[...]
.eh_frame
.dynstr
.dynsym
.hash
.dynamic
```

Symbol leakage

Symbol leakage

```
$> nm libapp.so
```

Symbol leakage

```
$> nm libapp.so
libapp.so: no symbols
```

Symbol leakage

```
$> nm libapp.so
libapp.so: no symbols
```

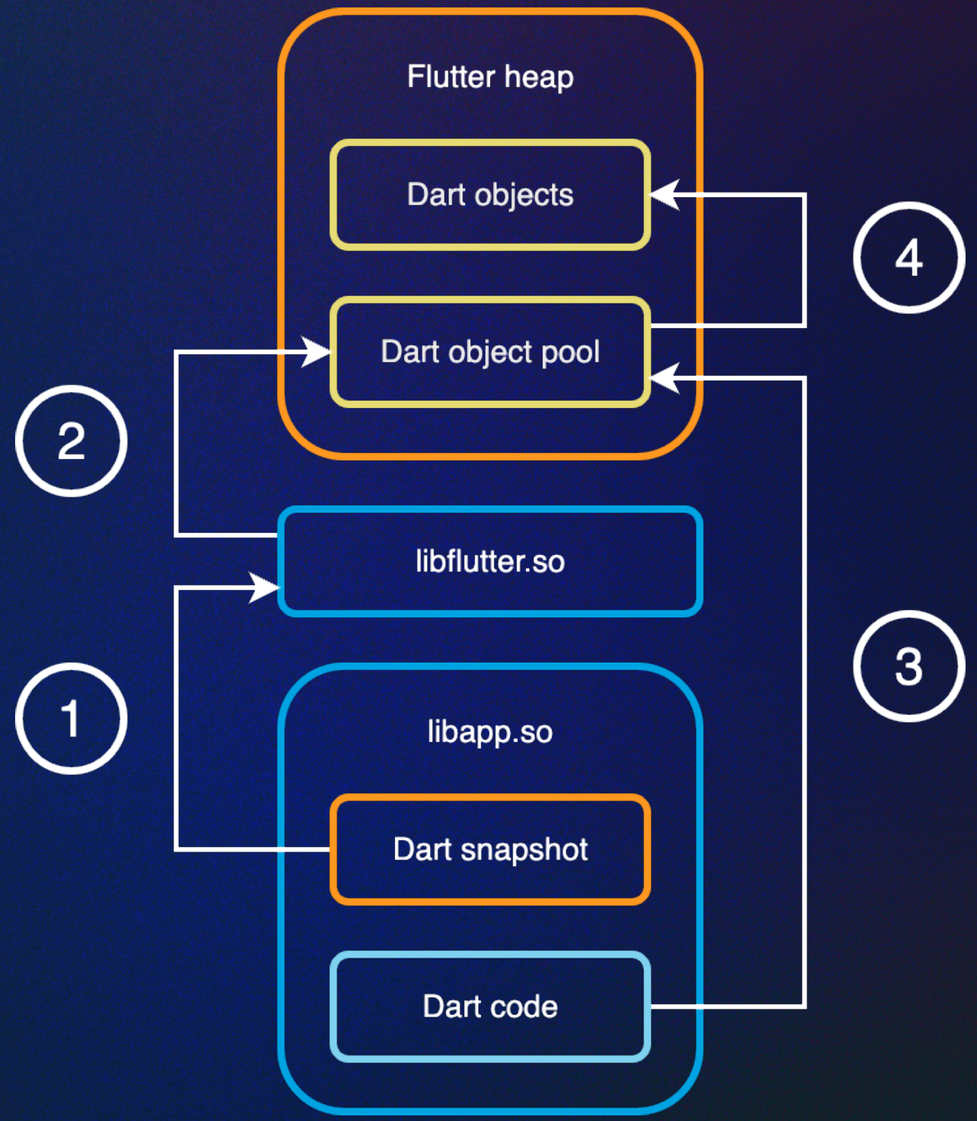
🤔

Symbol leakage

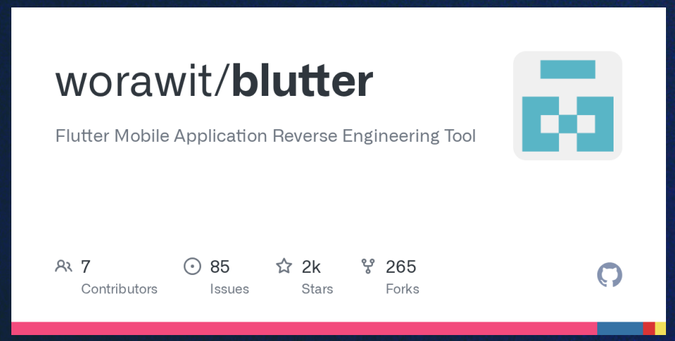
```
$> strings libapp.so
```

```
[...]
<optimized out>
Handle
MonomorphicSmiableCall
ClosureData
dyn:implicit:call
_NativeFinalizer
_FunctionType
Uint64List
[...]
verySecret
main
print
[...]
```

The Dart **object pool**



The Dart object pool




```
$> python blutter.py --dart-version “3.9.2_android_arm64” libapp.so out

libapp is loaded at 0x1044f4000
Dart heap at 0x300000000
Analyzing the application
Dumping Object Pool
Generating application assemblies
Generating Frida script
```

The Dart **object pool**

```
out
├── asm
│   └── file:
│       └── Users
│           └── samuel
│               └── flutter
│                   └── bsides_talk
│                       └── low-effort
│                           └── low-effort.dart
├── blutter_frida.js
├── ida_script/
├── objs.txt
└── pp.txt
```

```
// low-effort.dart
// class id: 1048595, size: 0x8
class :: {

  [closure] static void main(dynamic) {
    // ** addr: 0x9aa24, size: 0x30
    // 0x9aa24: EnterFrame
    //     0x9aa24: stp             fp, lr, [SP, #-0x10]!
    //     0x9aa28: mov             fp, SP
    // [...]
  }
  static void verySecret() {
    // ** addr: 0x9aa54, size: 0x30
    // 0x9aa54: EnterFrame
    //     0x9aa54: stp             fp, lr, [SP, #-0x10]!
    //     0x9aa58: mov             fp, SP
    // [...]
  }
}
```

The Dart **object pool**

```
out
├── asm
│   └── file:
│       └── Users
│           └── samuel
│               └── flutter
│                   └── bsides_talk
│                       └── low-effort
│                           └── low-effort.dart
├── blutter_frida.js
├── ida_script/
├── objs.txt
└── pp.txt
```

```
// low-effort.dart
// class id: 1048595, size: 0x8
class :: {

  [closure] static void main(dynamic) {
    // ** addr: 0x9aa24, size: 0x30
    // 0x9aa24: EnterFrame
    //     0x9aa24: stp             fp, lr, [SP, #-0x10]!
    //     0x9aa28: mov             fp, SP
    // [...]
  }
  static void verySecret() {
    // ** addr: 0x9aa54, size: 0x30
    // 0x9aa54: EnterFrame
    //     0x9aa54: stp             fp, lr, [SP, #-0x10]!
    //     0x9aa58: mov             fp, SP
    // [...]
  }
}
```

The Dart **object pool**

```
0049aa24    int64_t main()

0049aa24  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa28  fd030faa   mov     fp, x15
0049aa2c  501f40f9   ldr     x16, [x26, #0x38]
0049aa30  ff0110eb   cmp     x15, x16
0049aa34  c9000054   b.ls    0x49aa4c

0049aa38  07000094   bl      verySecret
0049aa3c  e00316aa   mov     x0, x22
0049aa40  ef031daa   mov     x15, fp
0049aa44  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aa48  c0035fd6   ret

0049aa4c  04f8ff97   bl      sub_498a5c
0049aa50  fafffff17  b       0x49aa38
```

The Dart object pool

```
0049aa54    int64_t verySecret()

0049aa54  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa58  fd030faa   mov     fp, x15
0049aa5c  501f40f9   ldr     x16, [x26, #0x38]
0049aa60  ff0110eb   cmp     x15, x16
0049aa64  c9000054   b.ls    0x49aa7c

0049aa68  07000094   bl      print
0049aa6c  e00316aa   mov     x0, x22
0049aa70  ef031daa   mov     x15, fp
0049aa74  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aa78  c0035fd6   ret

0049aa7c  f8f7ff97   bl      sub_498a5c
0049aa80  fafffff17  b       0x49aa68
```

The Dart object pool

```
0049aa84    int64_t print()

0049aa84  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa88  fd030faa   mov     fp, x15
0049aa8c  501f40f9   ldr     x16, [x26, #0x38]
0049aa90  ff0110eb   cmp     x15, x16
0049aa94  e9000054   b.ls    0x49aab0

0049aa98  615f4ef9   ldr     x1, [x27, #0x1cb8]
0049aa9c  07000094   bl      printToConsole
0049aaa0  e00316aa   mov     x0, x22
0049aaa4  ef031daa   mov     x15, fp
0049aaa8  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aaac  c0035fd6   ret

0049aab0  ebf7ff97   bl      sub_498a5c
0049aab4  f9ffff17   b       0x49aa98
```

The Dart object pool

```
0049aa84    int64_t print()

0049aa84  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa88  fd030faa   mov     fp, x15
0049aa8c  501f40f9   ldr     x16, [x26, #0x38]
0049aa90  ff0110eb   cmp     x15, x16
0049aa94  e9000054   b.ls    0x49aab0

0049aa98  615f4ef9   ldr     x1, [x27, #0x1cb8]
0049aa9c  07000094   bl      printToConsole
0049aaa0  e00316aa   mov     x0, x22
0049aaa4  ef031daa   mov     x15, fp
0049aaa8  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aaac  c0035fd6   ret

0049aab0  ebf7ff97   bl      sub_498a5c
0049aab4  f9ffff17   b       0x49aa98
```

The Dart **object pool**

```
0049aa84    int64_t print()

0049aa84  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa88  fd030faa   mov     fp, x15
0049aa8c  501f40f9   ldr     x16, [x26, #0x38]
0049aa90  ff0110eb   cmp     x15, x16
0049aa94  e9000054   b.ls    0x49aab0

0049aa98  615f4ef9   ldr     x1, [x27, #0x1cb8]
0049aa9c  07000094   bl      printToConsole
0049aaa0  e00316aa   mov     x0, x22
0049aaa4  ef031daa   mov     x15, fp
0049aaa8  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aaac  c0035fd6   ret

0049aab0  ebf7ff97   bl      sub_498a5c
0049aab4  f9ffff17   b       0x49aa98
```

```
$> grep “0x1cb8” out/pp.txt
```

The Dart object pool

```
0049aa84    int64_t print()

0049aa84  fd79bfa9   stp     fp, lr, [x15, #-0x10]!
0049aa88  fd030faa   mov     fp, x15
0049aa8c  501f40f9   ldr     x16, [x26, #0x38]
0049aa90  ff0110eb   cmp     x15, x16
0049aa94  e9000054   b.ls    0x49aab0

0049aa98  615f4ef9   ldr     x1, [x27, #0x1cb8]
0049aa9c  07000094   bl      printToConsole
0049aaa0  e00316aa   mov     x0, x22
0049aaa4  ef031daa   mov     x15, fp
0049aaa8  fd79c1a8   ldp     fp, lr, [x15], #0x10
0049aaac  c0035fd6   ret

0049aab0  ebf7ff97   bl      sub_498a5c
0049aab4  f9ffff17   b       0x49aa98
```

$> grep “0x1cb8” out/pp.txt

[pp+0x1cb8] String: "My cool API key: AIzaSyDaGmWKa4JsXZ-HjGw7ISLn_3namBGewQe"

Insecure TLS validation

```dart
import 'dart:io';
import 'package:http/io_client.dart';

class InsecureHttpOverrides extends HttpOverrides {
  @override
  HttpClient createHttpClient(final SecurityContext? context) {
    return super.createHttpClient(context)
      ..badCertificateCallback = (X509Certificate cert, String host, int port) => true;
  }
}

void main() async {
  HttpOverrides.global = new InsecureHttpOverrides();
  var response = await IOClient(HttpClient()).get(Uri.parse('https://self-signed.badssl.com/'));
  print(response.body);
}
```

Insecure TLS validation

Insecure TLS validation

- **Why** do you need non-standard behavior?

Insecure TLS validation

- **Why** do you need non-standard behavior?
- Self-signed certificates, custom CAs etc?

  → Really necessary?

Insecure TLS validation

- **Why** do you need non-standard behavior?
- Self-signed certificates, custom CAs etc?

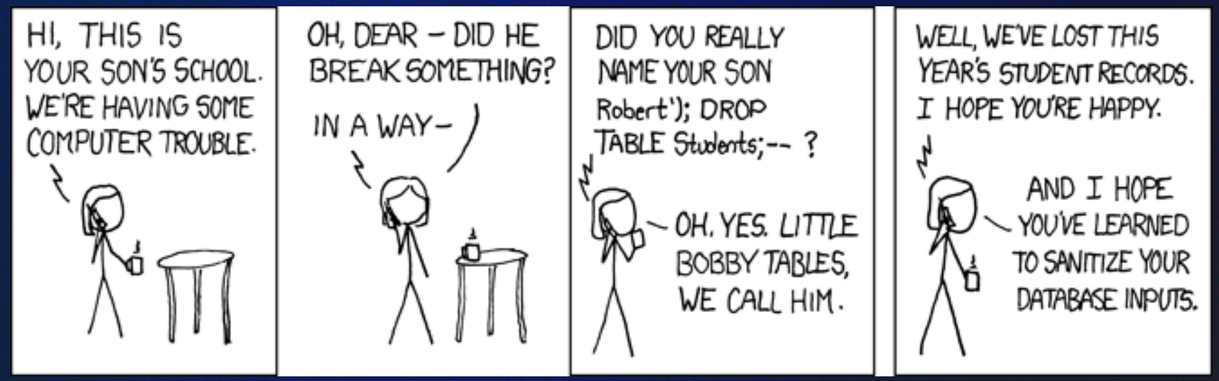
  → Really necessary?
- Use well-tested libraries instead of custom code

Injection vulnerabilities

```dart
import 'dart:io';
import 'package:sqflite_common/sqlite_api.dart';

void updateTask(Database db, Int taskId, String fileName) async {
  await db.execute(
    'UPDATE task SET file_name="$filename" WHERE task_id="$taskId";'
  );
}
```

Injection vulnerabilities

```dart
import 'dart:io';
import 'package:sqflite_common/sqlite_api.dart';

void updateTask(Database db, Int taskId, String fileName) async {
  await db.execute(
    'UPDATE task SET file_name="$filename" WHERE task_id="$taskId";'
  );
}

// 💀
updateTask(db, 'whatever", url="https://evil.com" where 1 = 1 ;--');
```

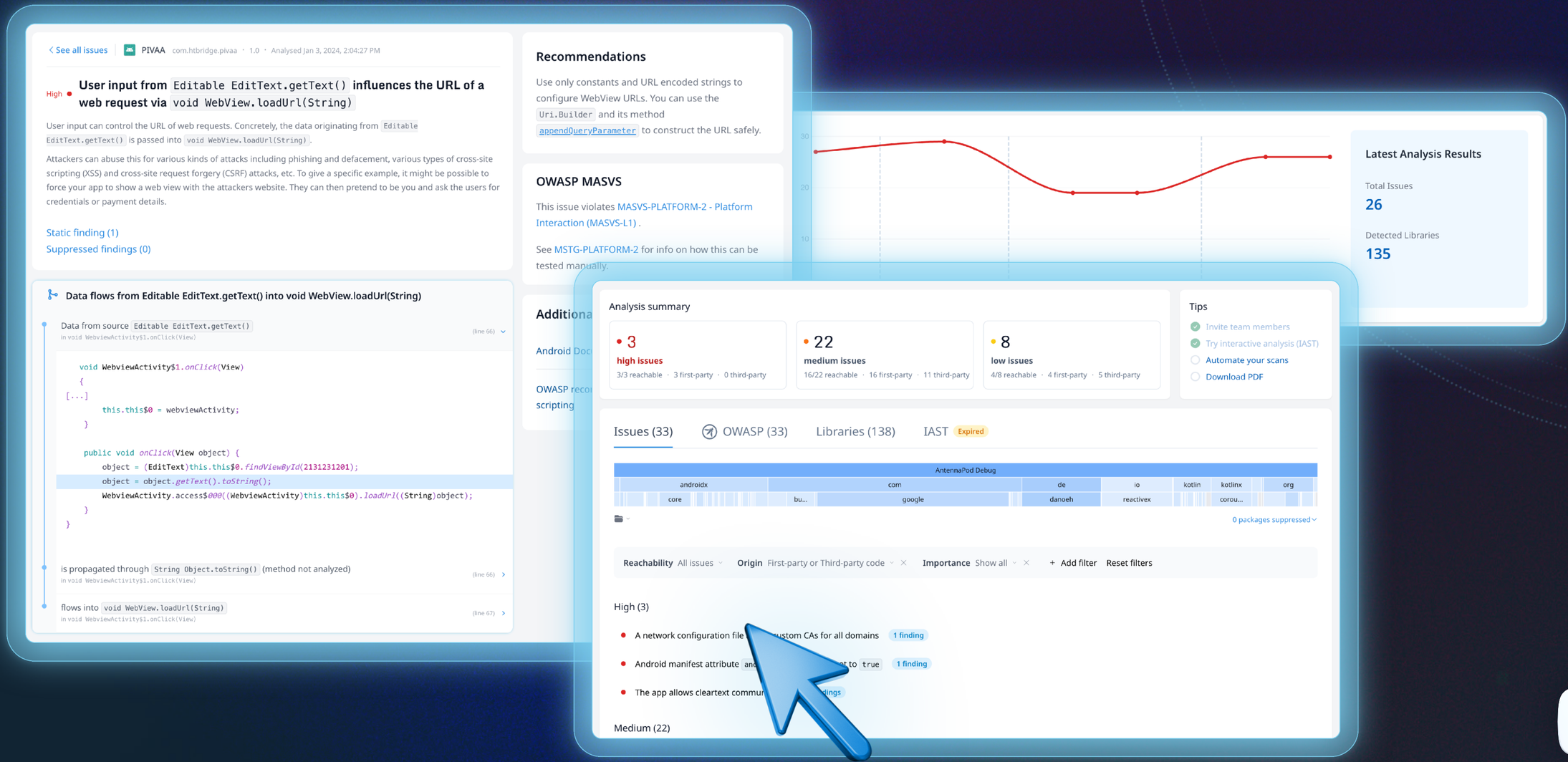


Injection vulnerabilities

- Identify sources of external inputs
- Reject/verify/sanitize input data
- Special care necessary for APIs related to
  - Serialization
  - Dynamic code loading/execution
  - Databases
  - Filesystem interaction
  - LLM prompts

Tamper resistant ≠ vulnerability-free

Automated security testing

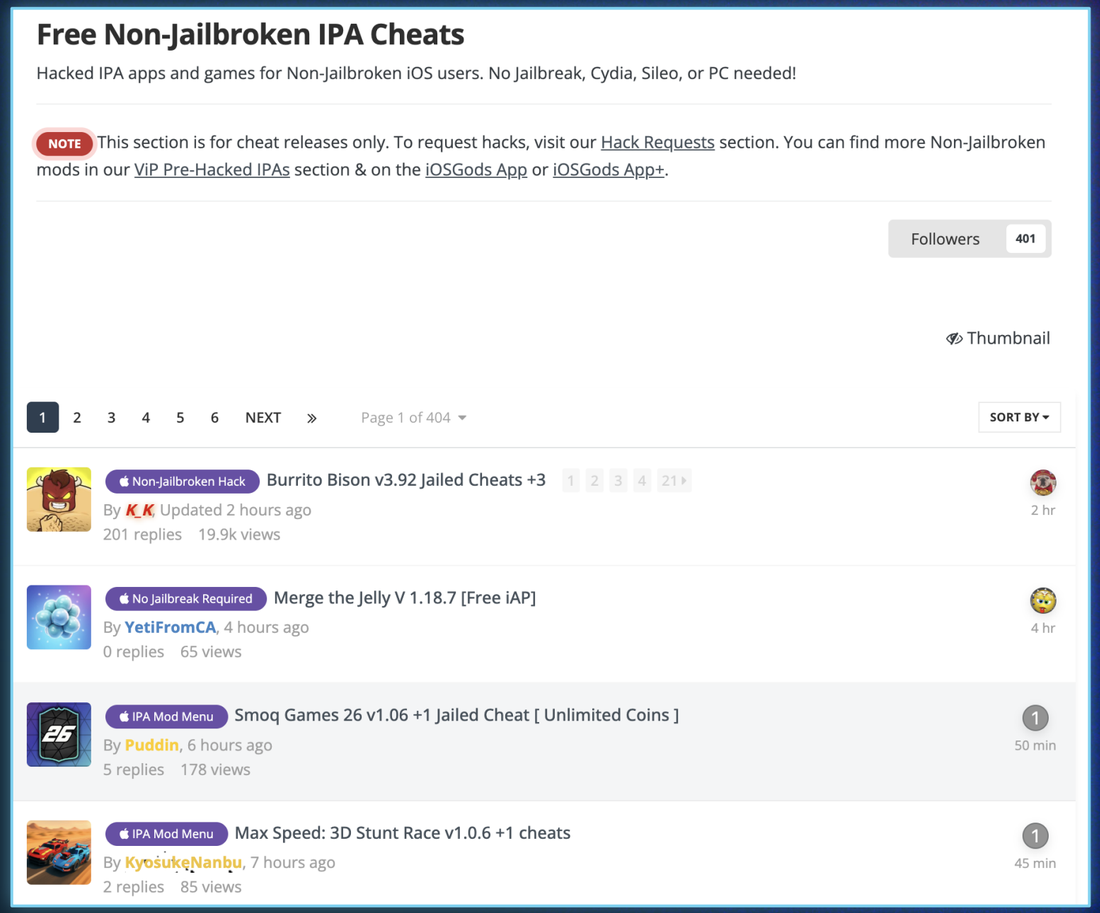


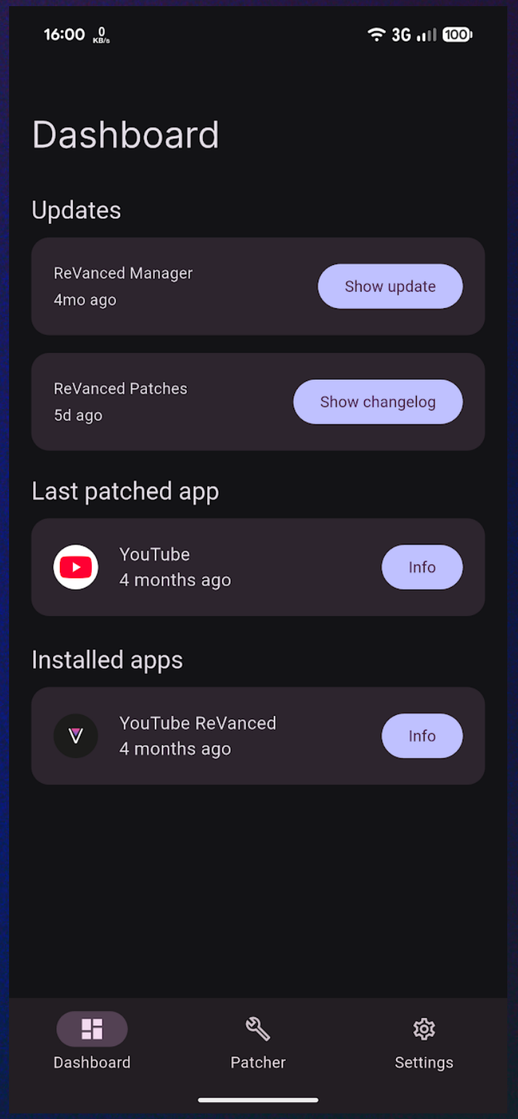
Runtime tampering

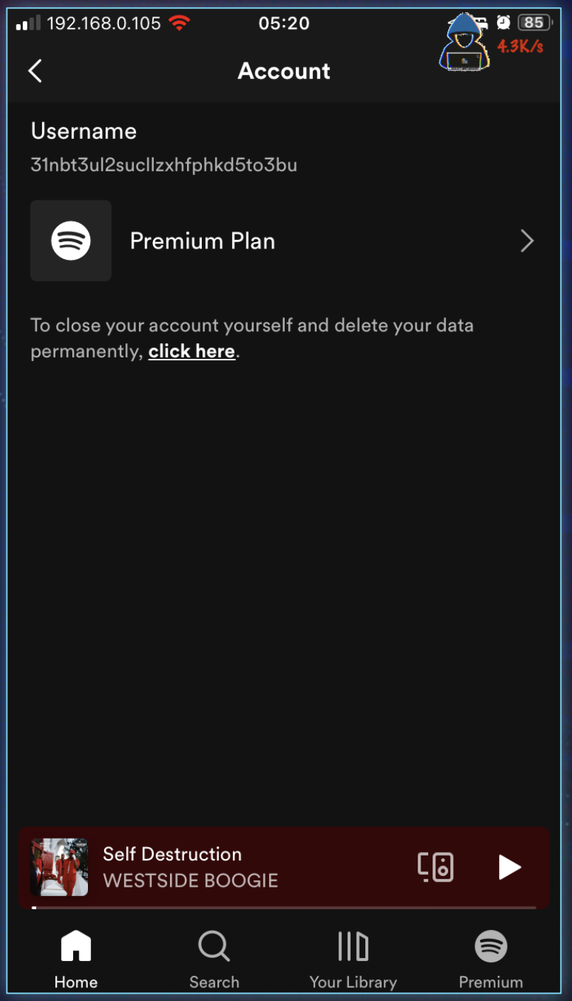
- Inline hooking (patching function prologues)
- Direct patching of function bodies
- Hook Flutter VM
- Tamper with Java/Swift parts of the app
- …



Repackaging attacks



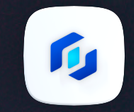




Typical repackaging approach

Typical repackaging approach

1. **Unpack bundle**



Typical repackaging approach

1. **Unpack bundle**
2. **Modify binaries and/or resources**
    - Patch out security checks
    - Remove paywall restrictions
    - Add custom features
    - Remove ads
    - Alter AndroidManifest.xml/Info.plist
    - ...

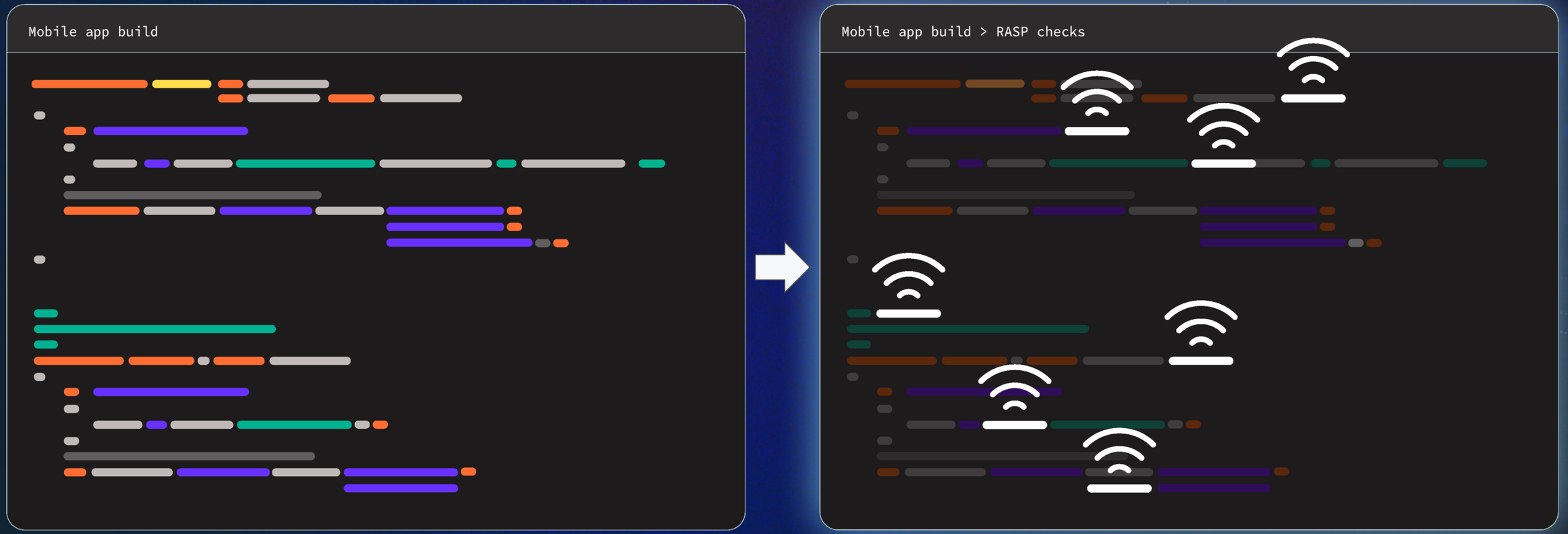
Typical repackaging approach

1. **Unpack bundle**
2. **Modify binaries and/or resources**
    - Patch out security checks
    - Remove paywall restrictions
    - Add custom features
    - Remove ads
    - Alter AndroidManifest.xml/Info.plist
    - ...
3. **Re-zip and re-sign bundle**

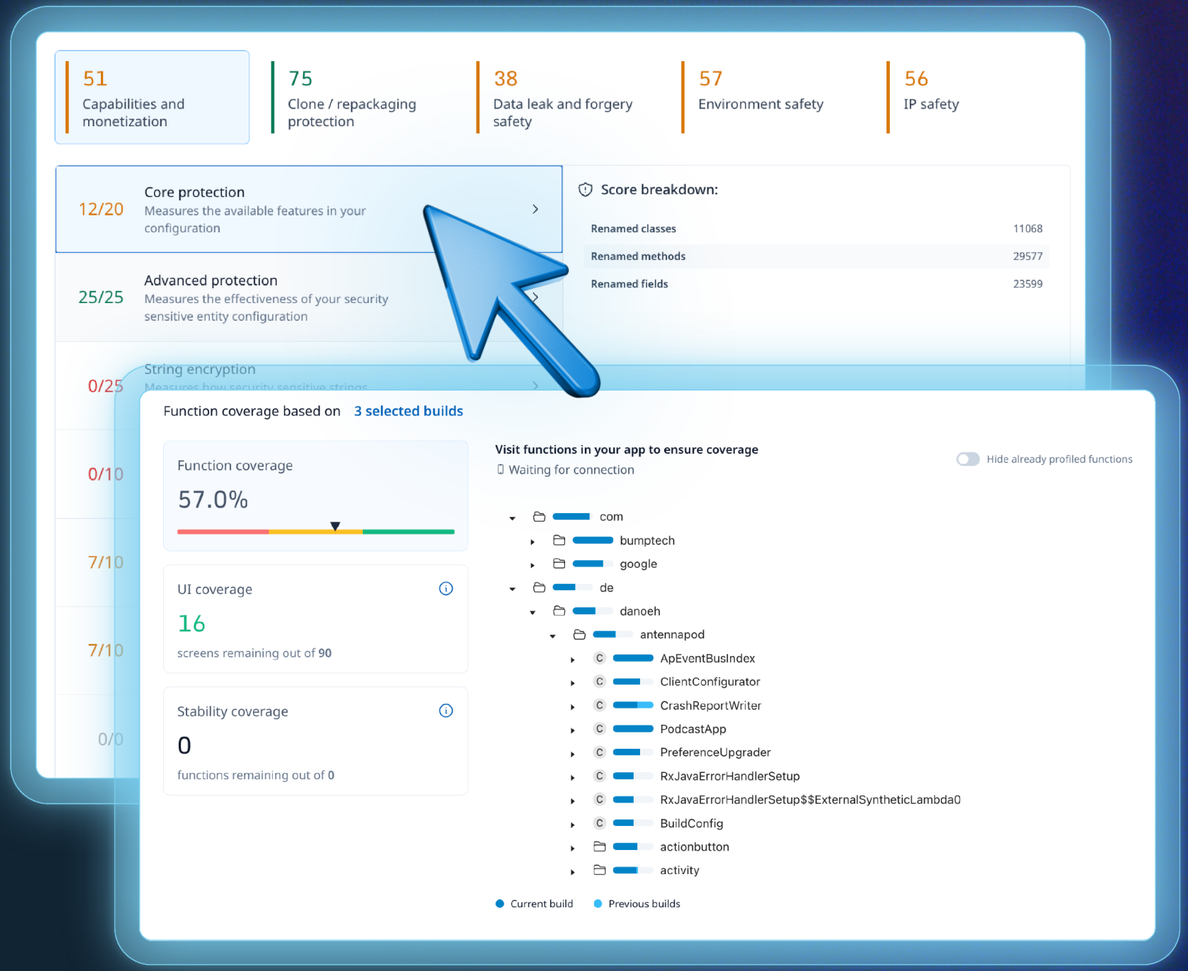
Typical repackaging approach

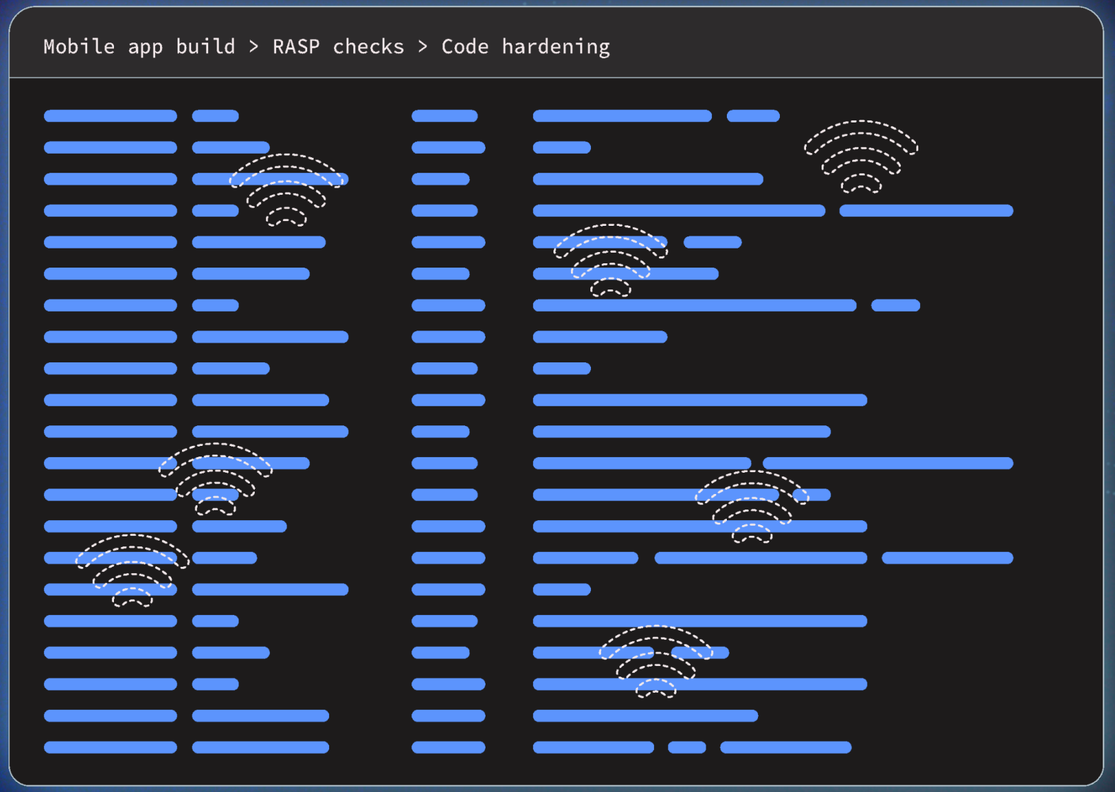
1. **Unpack bundle**
2. **Modify binaries and/or resources**
   - Patch out security checks
   - Remove paywall restrictions
   - Add custom features
   - Remove ads
   - Alter AndroidManifest.xml/Info.plist
   - ...
3. **Re-zip and re-sign bundle**
4. **Distribute and install app**

Runtime app self-protection

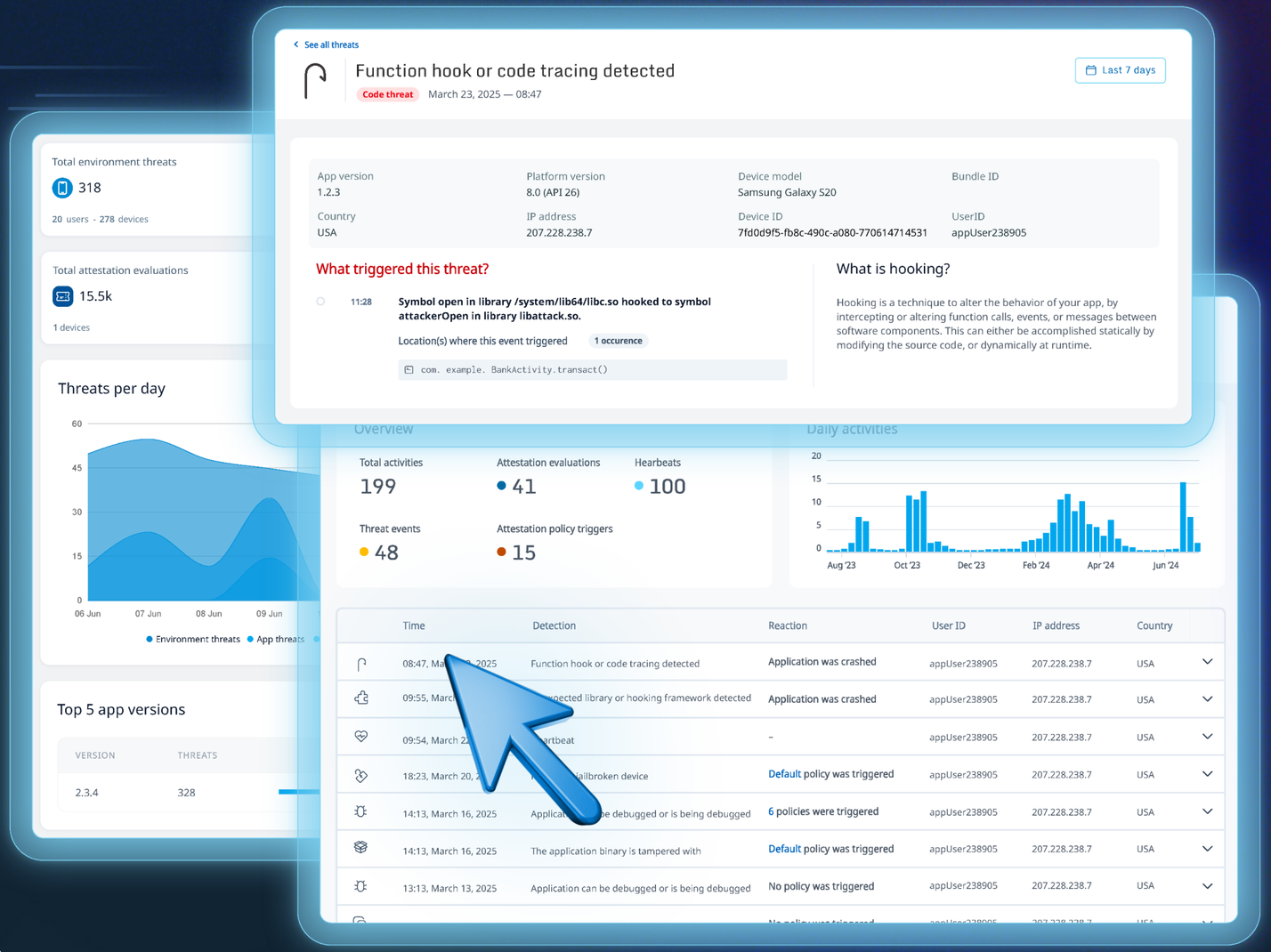


Multi-layered protection strategy





Monitoring threats



Conclusion

Conclusion

- Be aware of what you ship

Conclusion

- Be aware of what you ship
- What is your attacker model?

Conclusion

- Be aware of what you ship
- What is your attacker model?
- Take a holistic approach to security

Conclusion

- Be aware of what you ship
- What is your attacker model?
- Take a holistic approach to security
- Integrate security aspects into your CI setup

Conclusion

- Be aware of what you ship
- What is your attacker model?
- Take a holistic approach to security
- Integrate security aspects into your CI setup
- Flutter apps are as (in)secure as regular apps

Thank you!

GUARDSQUARE
Mobile application protection

www.guardsquare.com