HITBSecConf

Amsterdam – April 12th, 2018

BRIDA

WHEN BURP SUITE MEETS FRIDA

THE SPEAKERS

mediaservice.net
CORPORATE SECURITY & IMAGE

Piergiovanni Cipolloni

Security Advisor

@Mediaservice.net S.r.l.

(piergiovanni.cipolloni@mediaservice.net)

- ~ 20 years in Penetration Testing
- Security researcher

THE SPEAKERS

mediaservice.net
CORPORATE SECURITY & IMAGE

Federico Dotta

Security Advisor

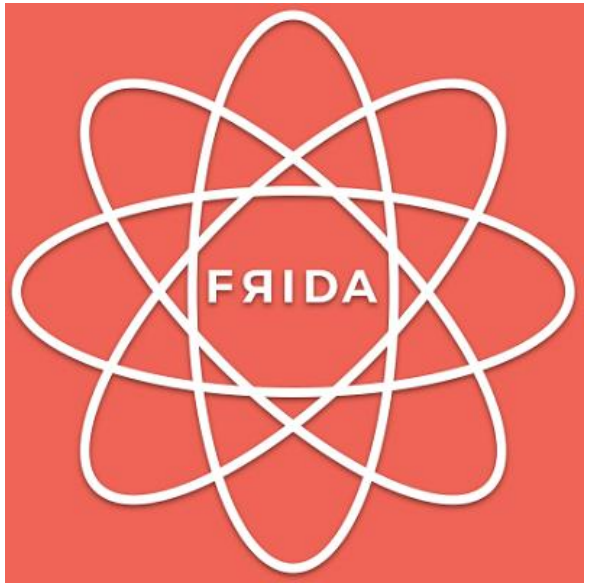
@Mediaservice.net S.r.l.

(federico.dotta@mediaservice.net)

- OSCP, CREST PEN, CSSLP
- 8+ years in Penetration Testing
- Focused on application security
- Developer of sec tools: https://github.com/federicodotta
- Trainer

TOPICS

BRIDA



Web Application

- Fixed client (web browser)
- Logic is usually mainly on the backend components
- Client-side application code is usually coded with interpreted languages
- Provisioned directly from the application server

Mobile Application

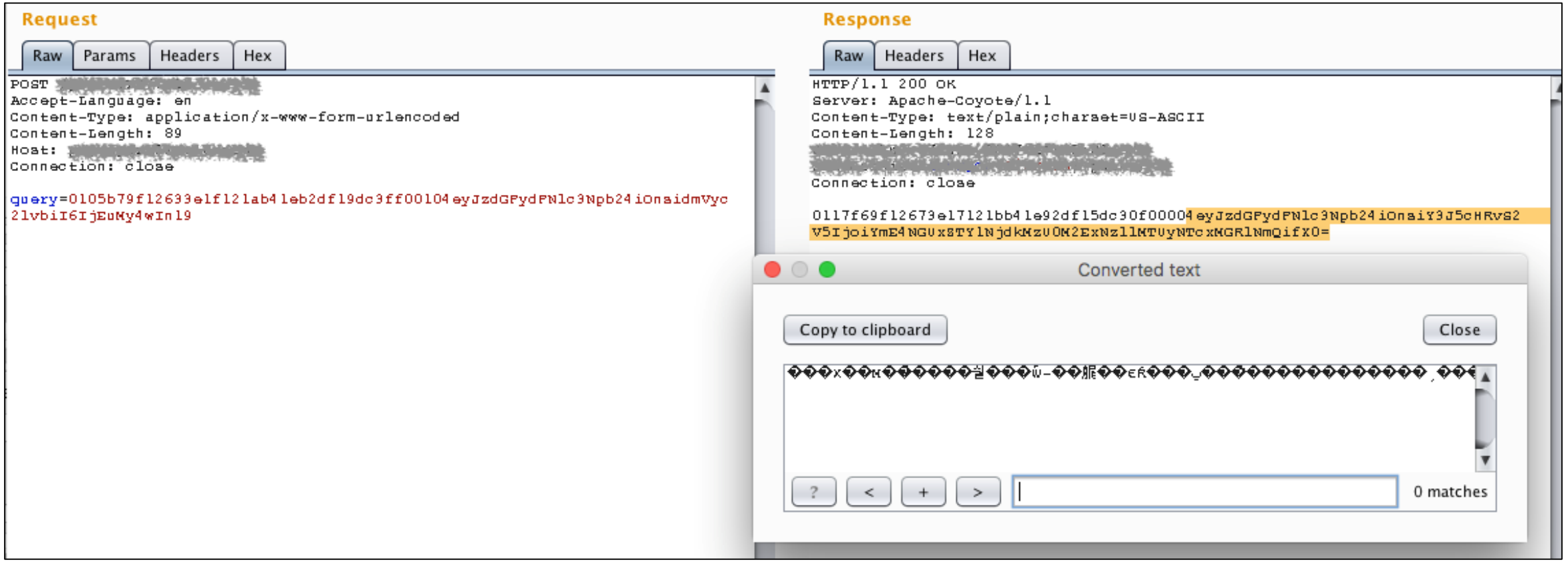
- Custom compiled client
- Logic is usually divided between client and backend
- Client-side application code can be interpreted or compiled
- Provisioned from a trusted third party

Mobile Applications

It's **almost impossible** to properly test a complex mobile application without skills in:

- Reversing (compiled Java/C code for Android,Objective-C/Swift code for iOS applications)
- Instrumentation and debugging
- Development of custom plugins for your favorite HTTP Proxy (Burp Suite, OWASP ZAP)

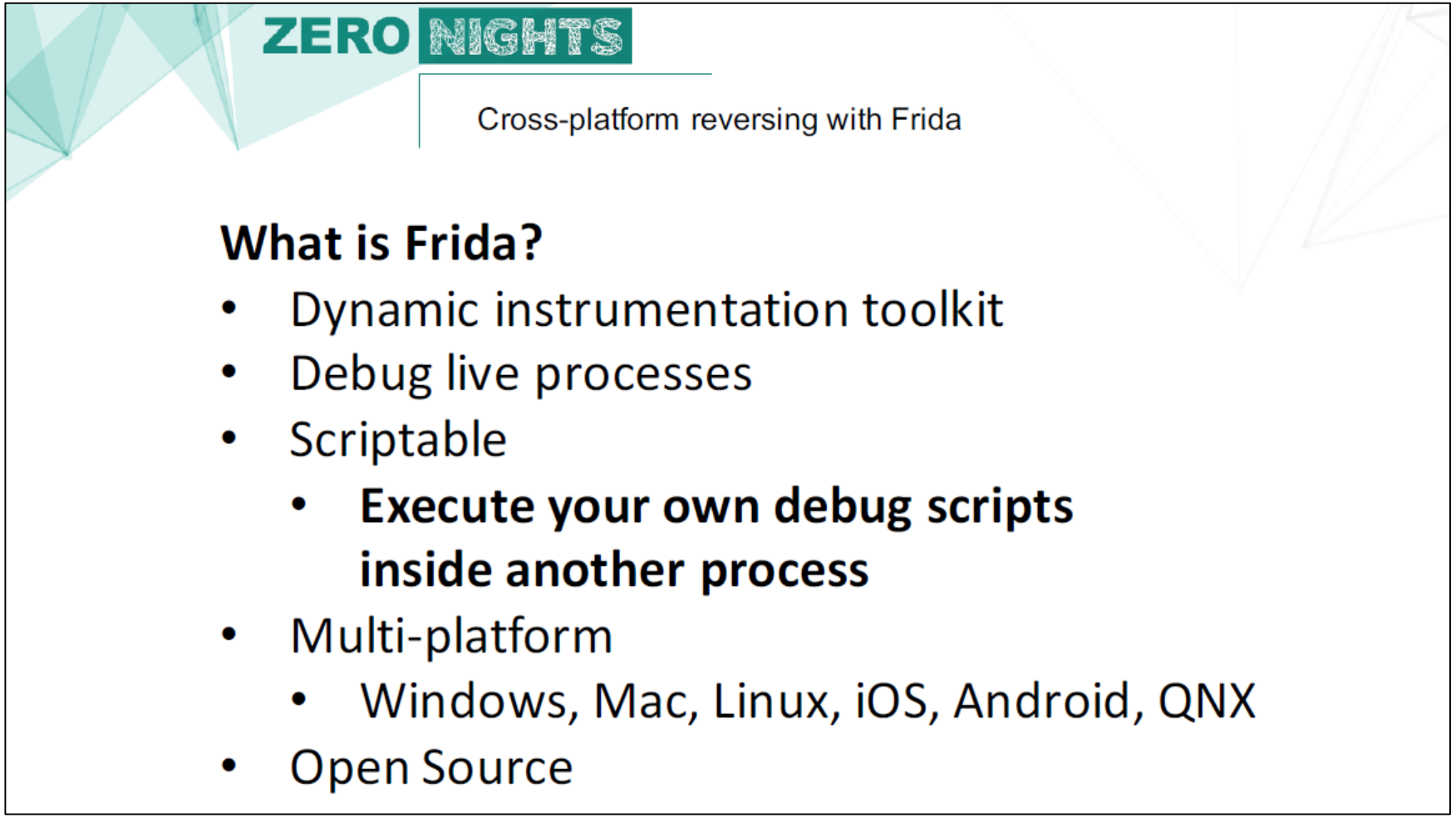
COMMON SCENARIO



PortSwigger Burp Suite

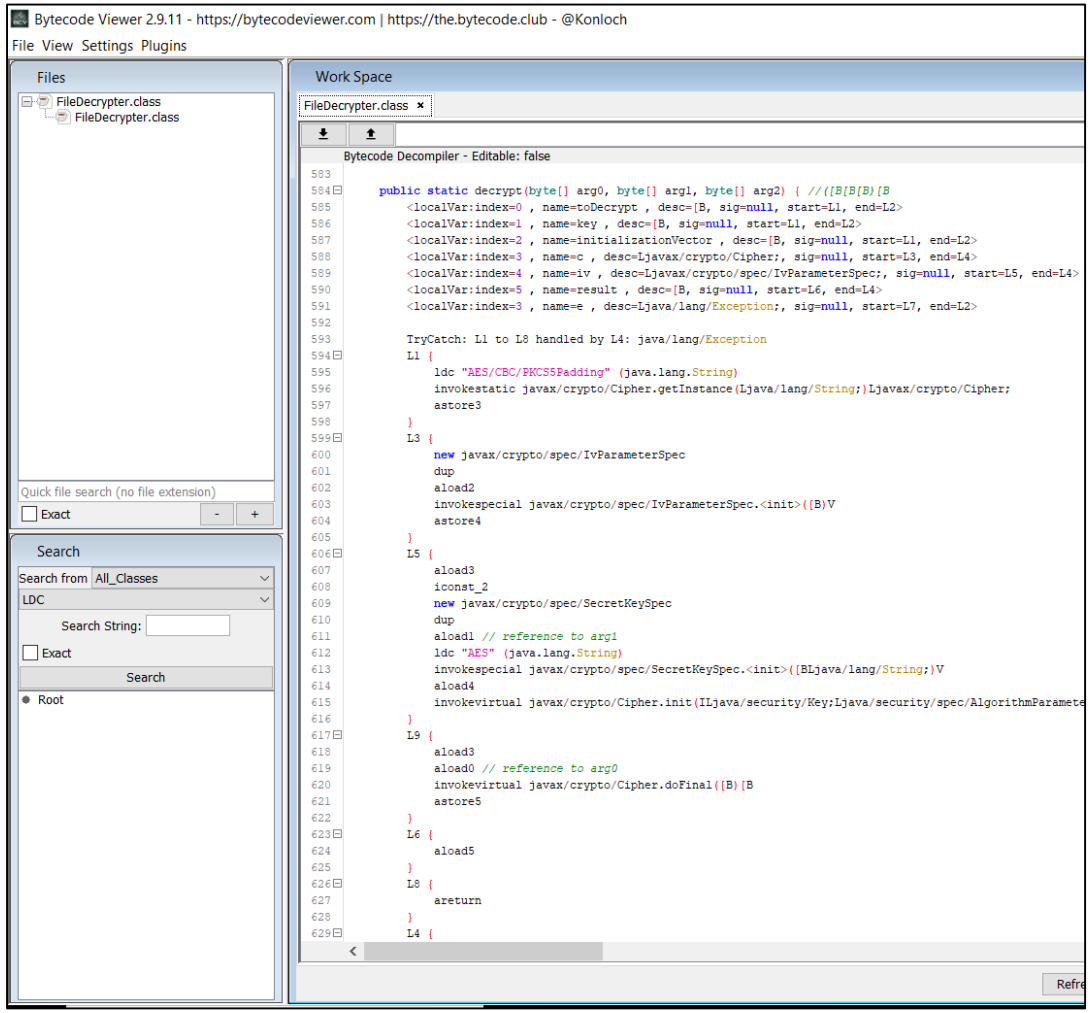
- Suite of tools that helps penetration testers during assessments
- It contains a lot of powerful tools: HTTP Proxy, Intruder (fuzzer), a great automatic Scanner and a Repeater tool
- Furthermore, it offers a server very useful to test external service interactions (Collaborator) and a excellent session manager
- It exports APIs to extend its functionalities, and consequently a huge number of plugins have been released by various developers to aid pentesters in almost any situation
- **It is de-facto standard for web application security testing.**

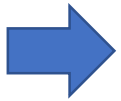
FRIDA



From ZERO NIGHTS - http://2015.zeronights.org/assets/files/23-Ravnas.pdf

BEFORE FRIDA





```java
import java.lang.Object;
import java.util.*;
import java.util.Base64;
import javax.crypto.SecretKey;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.Cipher;
import java.security.Key;
import java.security.*;
import java.security.spec.InvalidKeySpecException;


public class jaDec {

    public static void main(String[] args) {


        String plainText = "****************";

        byte[] bKey = new byte[] {(byte)0

        SecretKeySpec originalKey;

        Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());

        //originalKey = new SecretKeySpec(byteKey, 0, byteKey.length, "AES/CBC/PKCS7Padding");
        originalKey = new SecretKeySpec(bKey, "AES/CBC/PKCS7Padding");

        System.out.println("bKey len: " + bKey.length);

        byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
            IvParameterSpec ivspec = new IvParameterSpec(iv);

        try {

            //Cipher myCypherIn = Cipher.getInstance("AES/CTR/NoPadding", "BC");
            Cipher myCypherIn = Cipher.getInstance("AES/CTR/NoPadding", "BC");
            myCypherIn.init(Cipher.DECRYPT_MODE, originalKey, ivspec);
            System.out.println("Cipher Init");
            //String decryptText = new String(myCypherIn.doFinal(plainTextDe.getBytes("UTF-8")));
            //String decryptText = new String(myCypherIn.doFinal(Base64.getDecoder().decode(plainText)));
            String decryptText = new String(myCypherIn.doFinal(Base64.getDecoder().decode(plainText.getBytes("U
            System.out.println("Len: " + decryptText.length());
            System.out.println("Decrypted Text: " + decryptText);

        } catch (Exception ex1)
            {
                System.out.print(ex1.toString());
            }
    }
}
```

DECRYPTION - FRIDA EXAMPLE

```
4   var encryptionClass = Java.use("a.b.b");
5
6   encryptionClass.decryptMethod.overload("java.lang.String").implementation = function(cypherText) {
7
8       var plainText = this.decryptMethod.overload("java.lang.String").call(this,cypherText);
9
10      console.log(plainText);
11
12      return plainText;
13
14  };
```

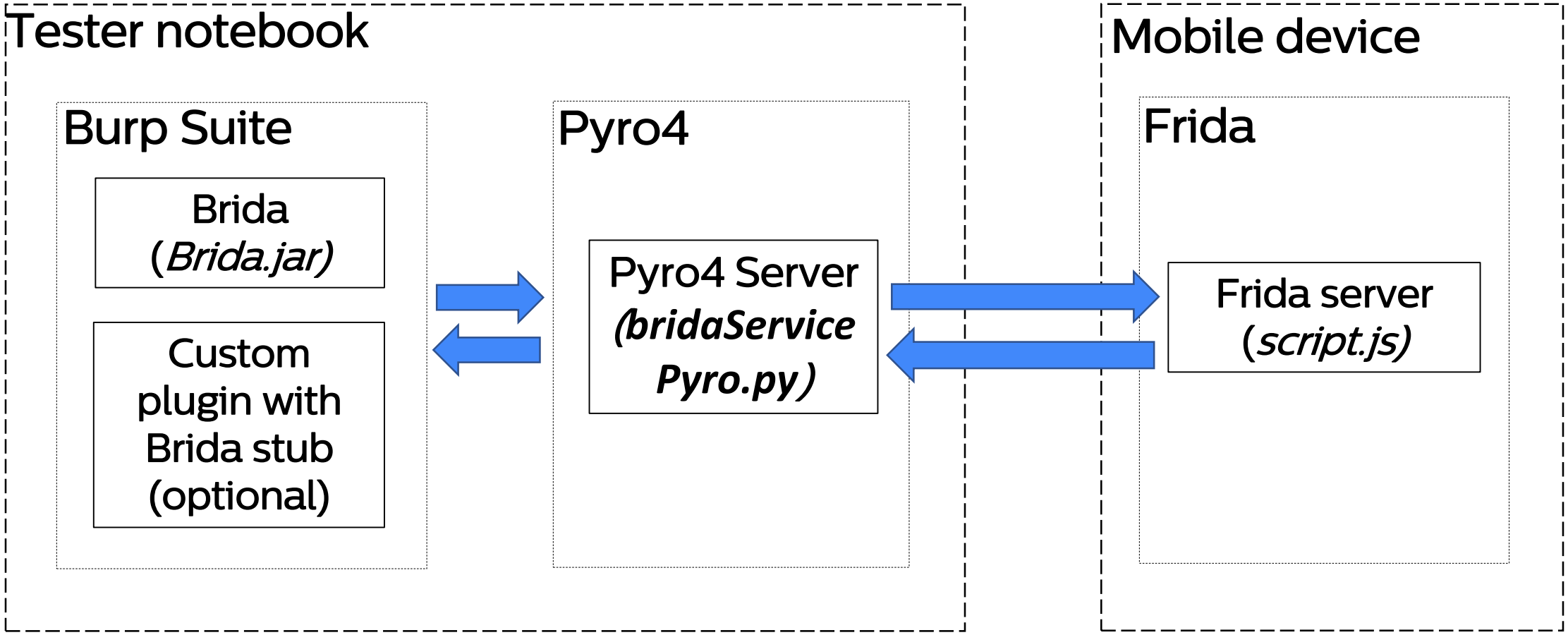
BUT SOMETIMES IS NOT ENOUGH...

- Let's take as an example a mobile application that uses symmetric crypto with random keys **in addition to TLS** to encrypt the POST bodies of all requests
- These random keys could be generated from a secret stored inside mobile device's protected areas (Secure Enclave)
- Also, supposing we know the secrets and all the details regarding the employed encryption algorithm, a complex Burp Suite plugin would be necessary to decrypt incoming requests and encrypt outgoing ones

WHY DON'T WE LET THE MOBILE APP DO THE DIRTY WORK FOR US?

- It acts as a **bridge** between **Burp Suite** and **Frida**
- Allows to call mobile application's functions directly from Burp Suite using Frida
- It is possible to code simple Burp Suite plugins that call mobile application's functions in order to execute complex tasks (for example encryption, hashing, signing, encoding) without having to fully understand how these complex tasks are accomplished and without having to reimplement them in our plugin

How does Brida work?



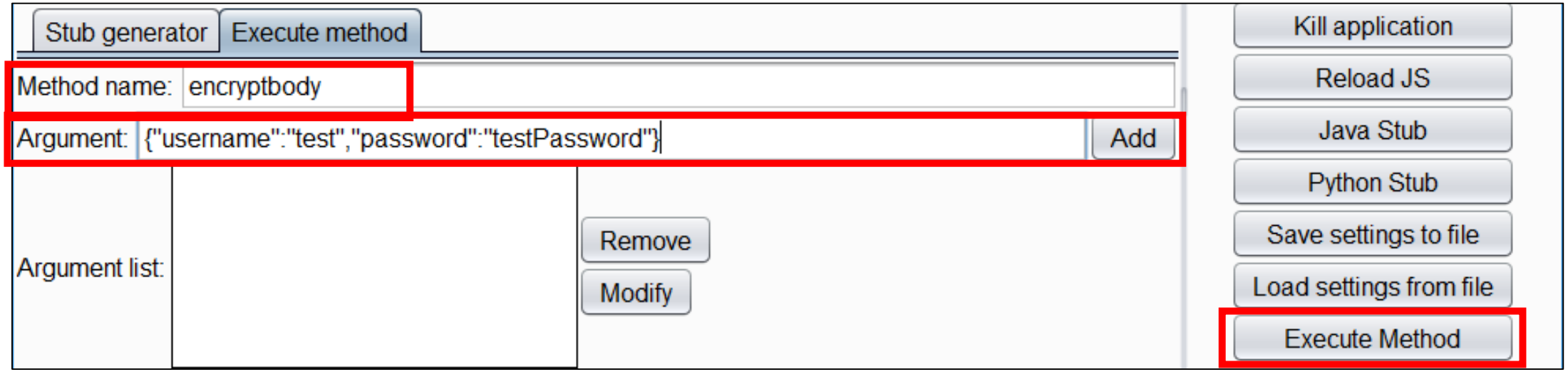
How does Brida work?

- Thanks to the «rpc» object of Frida it is possible to expose RPC-style functions
- From Burp Suite we call a Pyro function that acts as a bridge
- Pyro calls the selected Frida exported function and returns the result back to Burp Suite

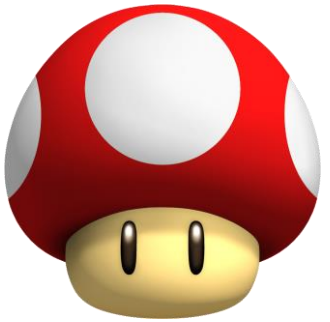
BRIDA 0.1

- Dedicated tab to call Frida exported functions and methods
- Context menu entries that call Frida exported functions
- Dedicated tab that generates code stubs for custom plugins

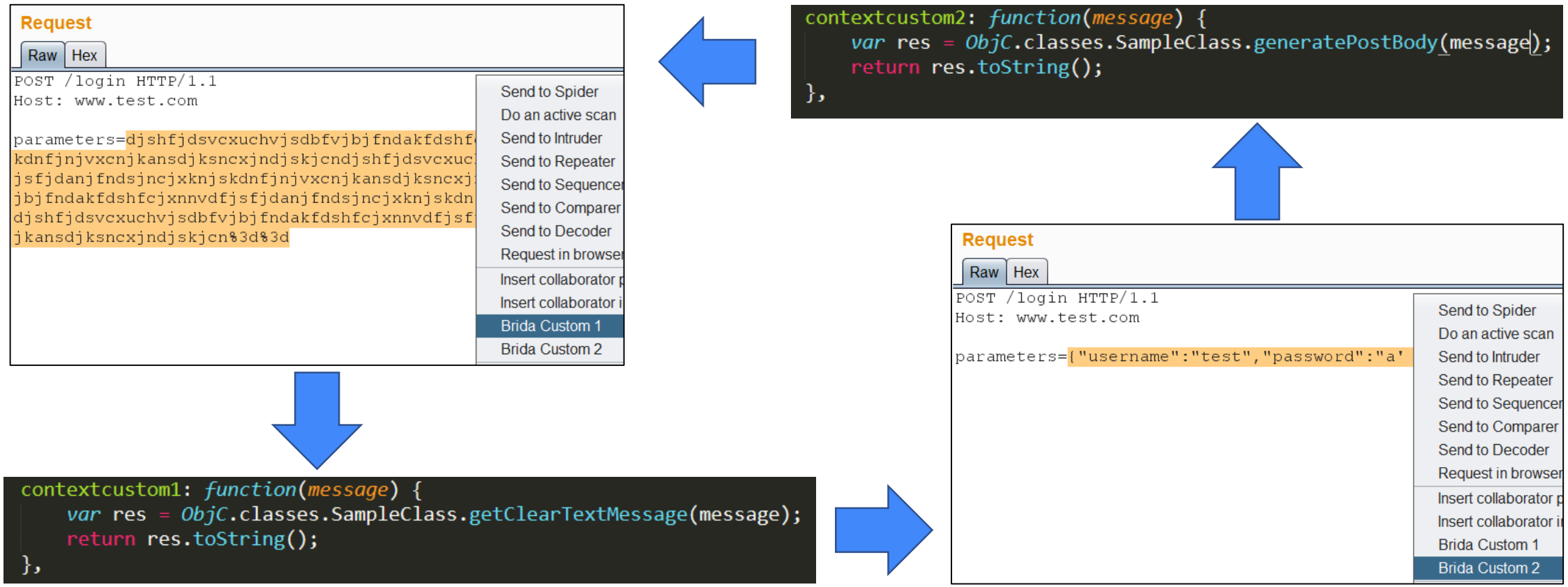
BRIDA 0.1 - EXECUTE METHOD



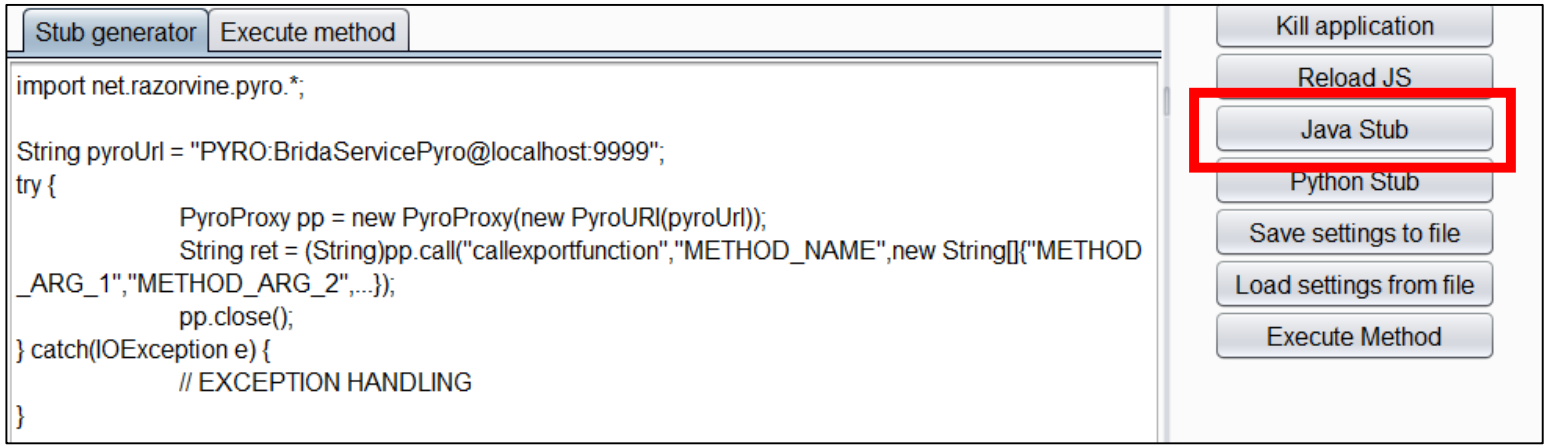
```
rpc.exports = {

    encryptbody: function(body) {
        var res = ObjC.classes.SampleClass.generatePostBody(body);
        return res.toString();
    }

}
```



BRIDA 0.1 - CONTEXT MENU



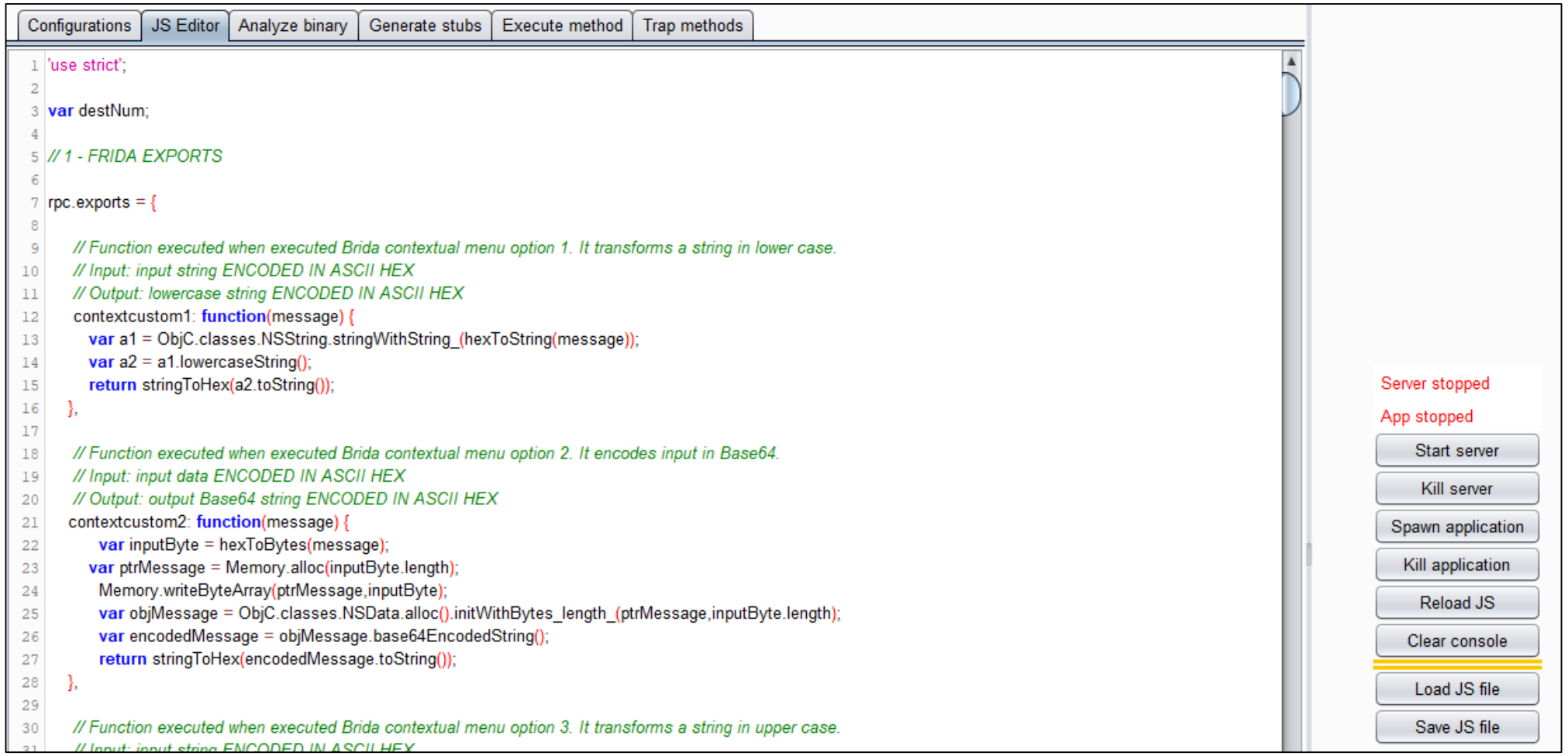
BRIDA 0.1 - STUB GENERATOR



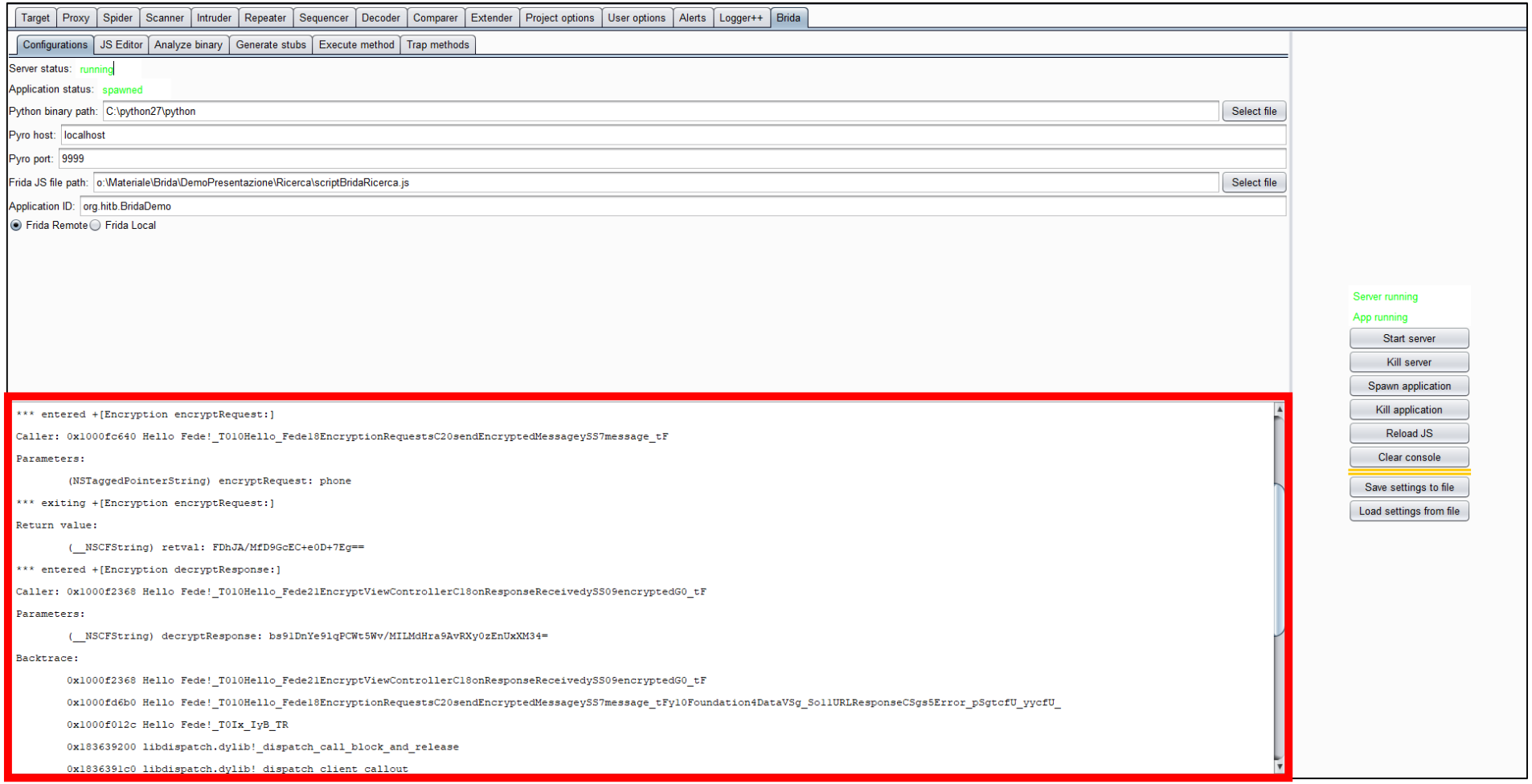
BRIDA 0.2

- Integrated JS editor
- Integrated Frida console
- Dedicated tab to analyze target binary
- Graphical hooking of functions for inspection
- Graphical hooking of functions for replacement

BRIDA 0.2 - JS EDITOR



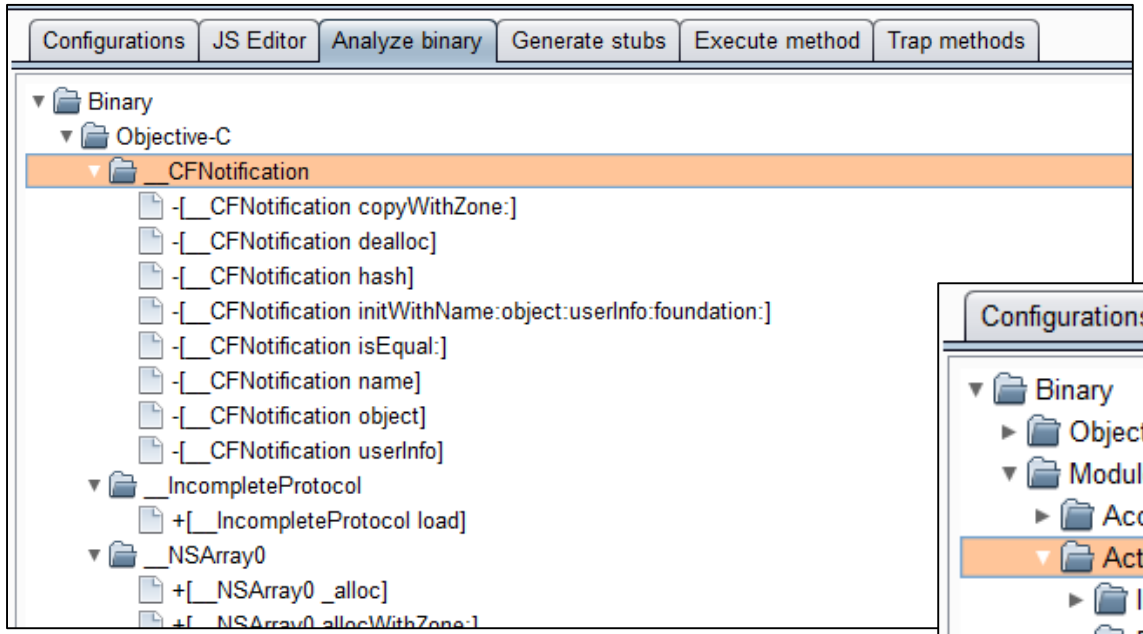
BRIDA 0.2 - FRIDA CONSOLE

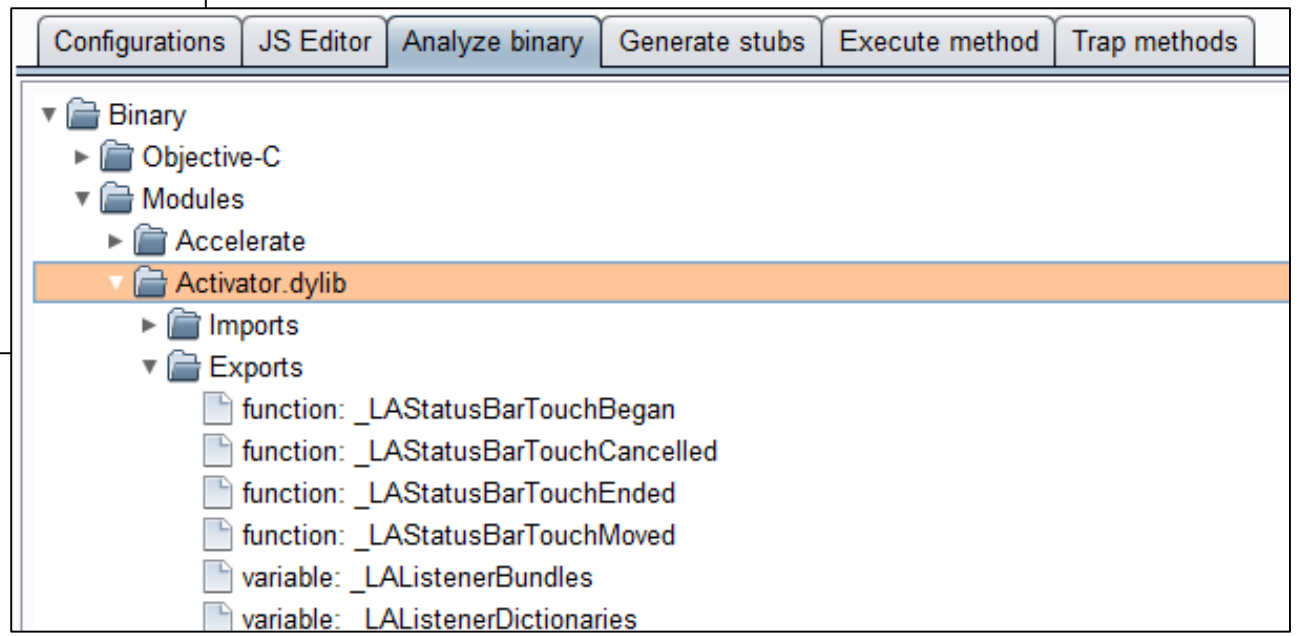


BRIDA 0.2 - ANALYSIS TAB

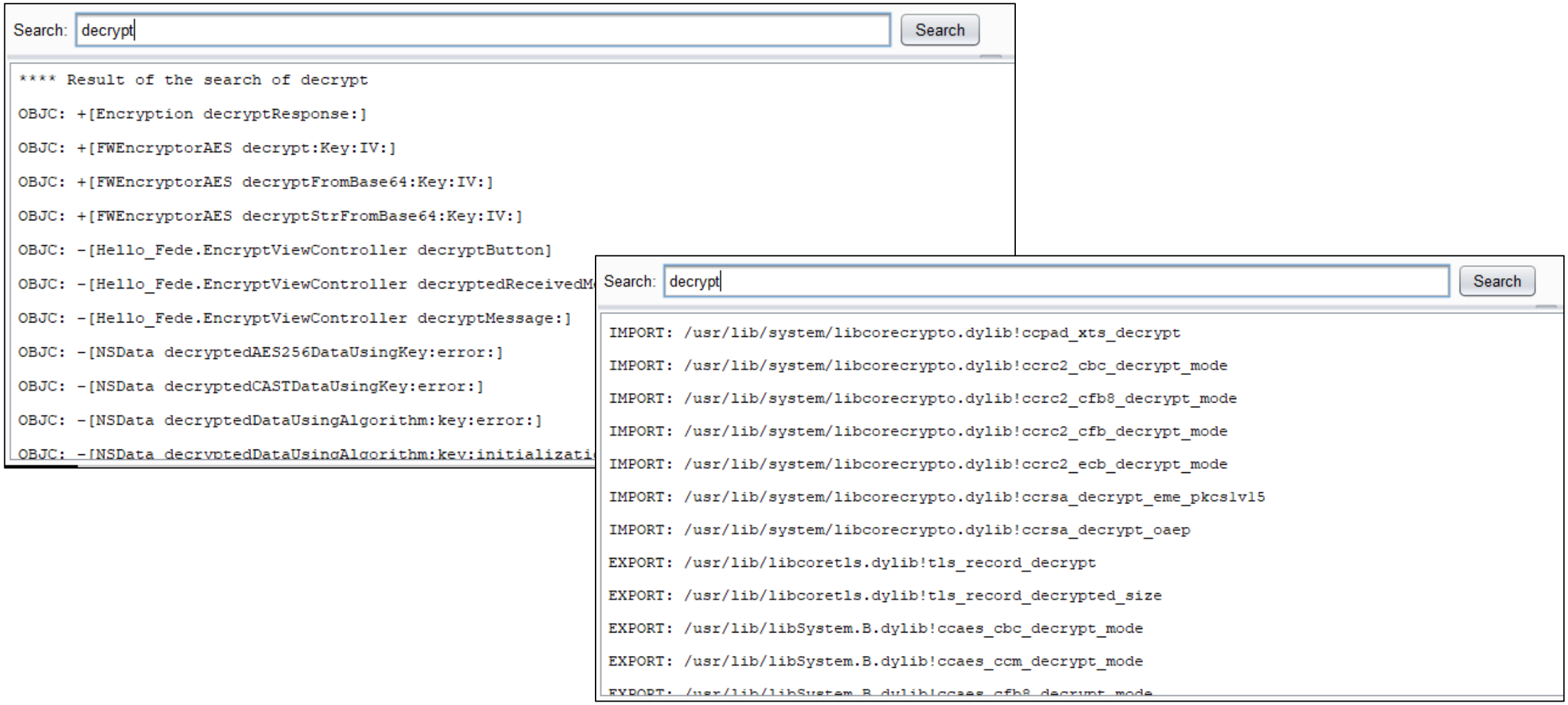
- Objective-C classes and methods graphical tree (iOS only)
- Java classes and methods graphical tree (Android only)
- Library imports and exports on all Frida supported platforms!
- «Search» functionality on Objective-C and library imports and exports (Java not supported due to Frida's current limitation on the «API Resolver» component)

BRIDA 0.2 - ANALYSIS TAB





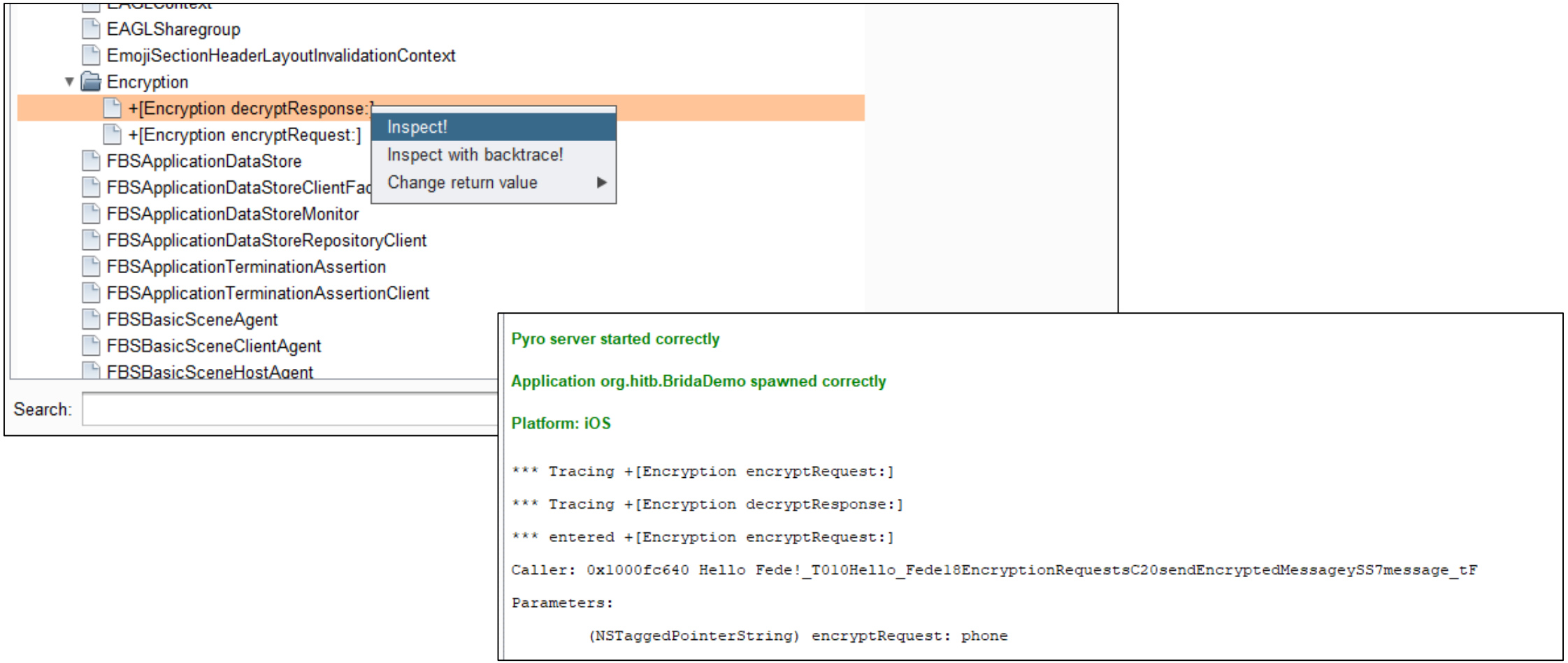
BRIDA 0.2 - ANALYSIS TAB - SEARCH



BRIDA 0.2 - GRAPHICAL INSPECTION

- By right-clicking on a method (Objective-C or Java) or an exported function it is possible to «inspect» that method/function
- From the click onwards, every time that the inspected function is executed in the binary, input parameters and return value will be printed out in the integrated output console
- It is also possible to inspect an entire Objective-C or Java class (all the contained methods will be hooked)
- «Print Backtrace» option is also available

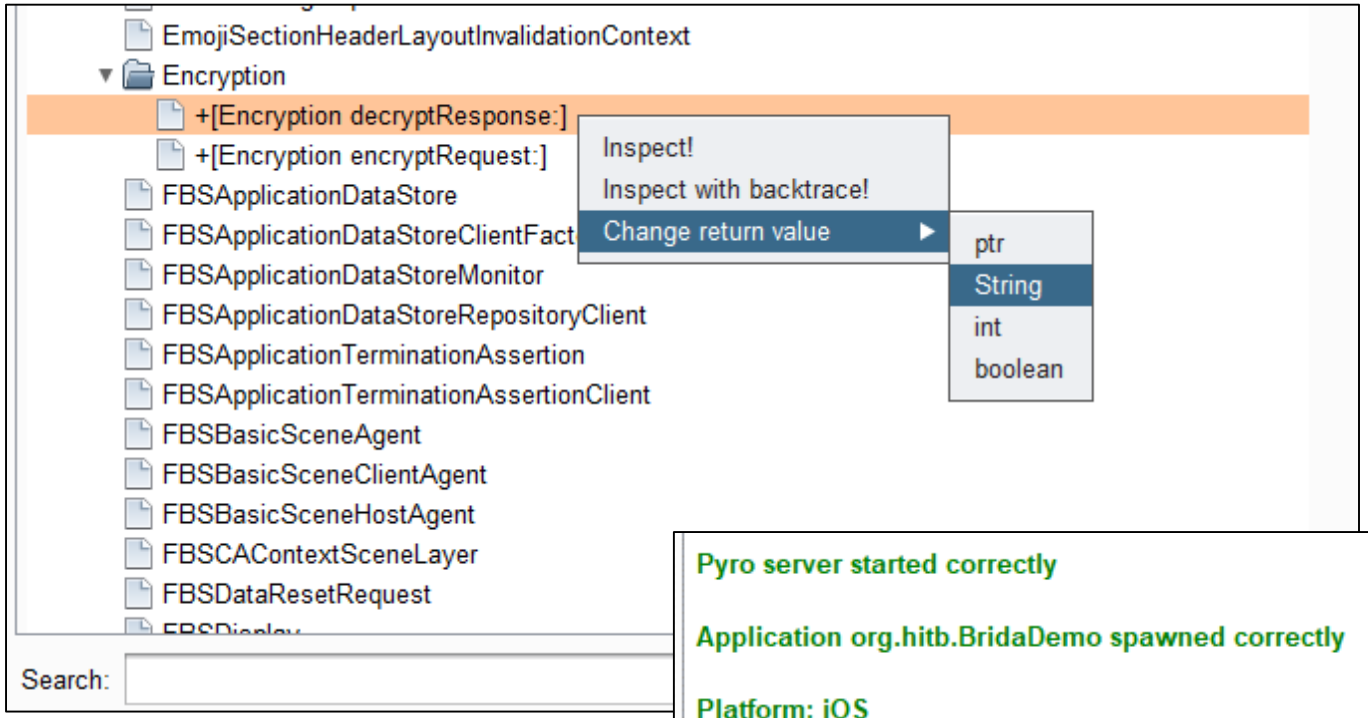
BRIDA 0.2 - GRAPHICAL INSPECTION



BRIDA 0.2 - CHANGE RETURN VALUE

- By right-clicking on a method Objective-C or Java) or an exported function it is also possible to change the return value of that method/ function
- Integer, String, Boolean and pointer are the supported return types, at the moment
- This functionality can be very useful to quickly bypass some security features (like «SSL pinning» or «Jailbreak/Root check»)

BRIDA 0.2 - CHANGE RETURN VALUE




```
Pyro server started correctly

Application org.hitb.BridaDemo spawned correctly

Platform: iOS

*** Replacing return value of +[Encryption decryptResponse:] with New return value

*** +[Encryption decryptResponse:] Replacing phone (price: 123)

 with New return value
```

BUT BRIDA WAS BORN TO...

...HANDLE COMPLEX SITUATIONS!

HANDLING COMPLEX SITUATIONS

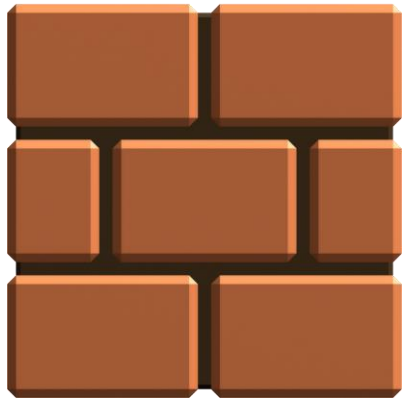
- An application that encrypts the body of all requests and responses with a custom and heavily-obfuscated algorithm
- An application that signs the body of all requests
- An application that periodically executes a challenge-response routine with the backend, computing the response based on complex and heavily-obfuscated logic

COMPLEXITIES WITHOUT BRIDA

- Testing applications that employ complex security features as the ones described in the previous slide is a mess!
- The job usually requires:
  - A lot of reversing to understand encryption and other security features (often heavily obfuscated!)
  - A lot of coding, in order to re-implement those features in a Burp Suite plugin
  - … because if we don't implement a plugin for our favorite HTTP proxy we are not able to thoroughly pentest the backend!

COMPLEXITIES WITH BRIDA

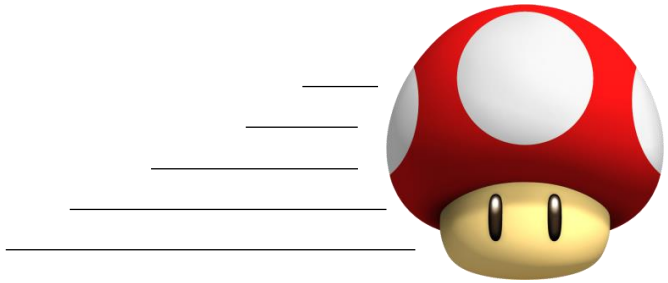
- Handling these situations with Brida is simpler and faster:
  - The reversing job is aimed only at finding functions used by the application to implement security features **without the need to understand how these features are implemented nor how they work!**
  - We will still need to code a Burp Suite plugin, but a very simple one with few lines of code **which only calls the mobile functions instead of having to re-implement them**, thanks to Brida and Frida!
  - We add an exported function to Frida JS that calls the mobile functions we need, and we call that exported function from our plugin



LET'S SEE HOW TO HANDLE THESE SITUATIONS IN THE DEMO!

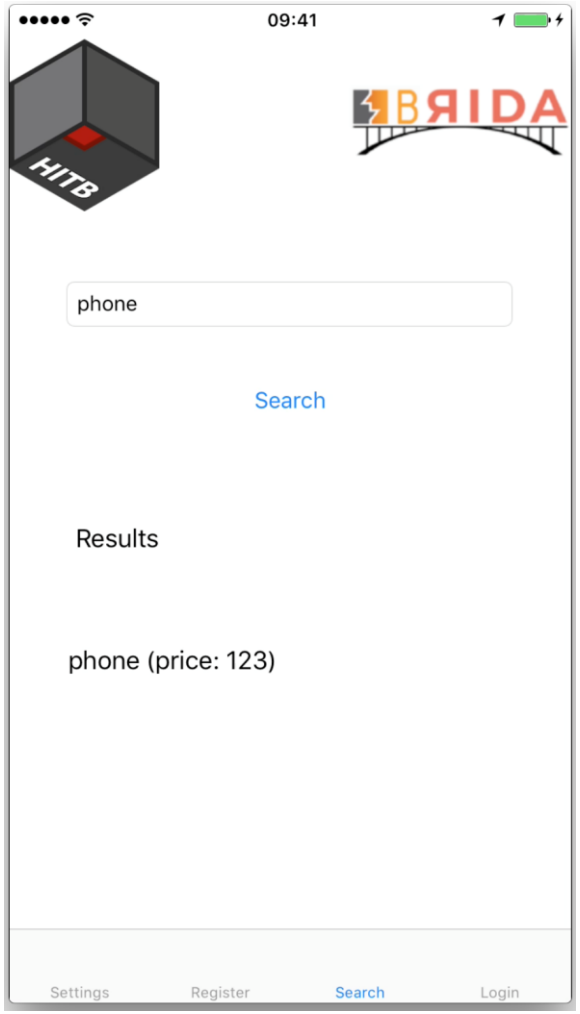
DEMO APPLICATION

- Three different use cases
- Each use case is a simplification of a real situation we faced during penetration tests conducted on mobile applications
- In all those situations Brida was almost essential



USE CASE 1

USE CASE 1



- We have a simple iOS app that provides a search functionality
- If we click on the «Search» button, the results are printed below the search form

USE CASE 1

Eκ(«apple»)

Search

Eκ(«Red apple»)

Response

Request: Eκ(«apple»)
Response: Eκ(«Red apple»)

Eκ(«apple»)

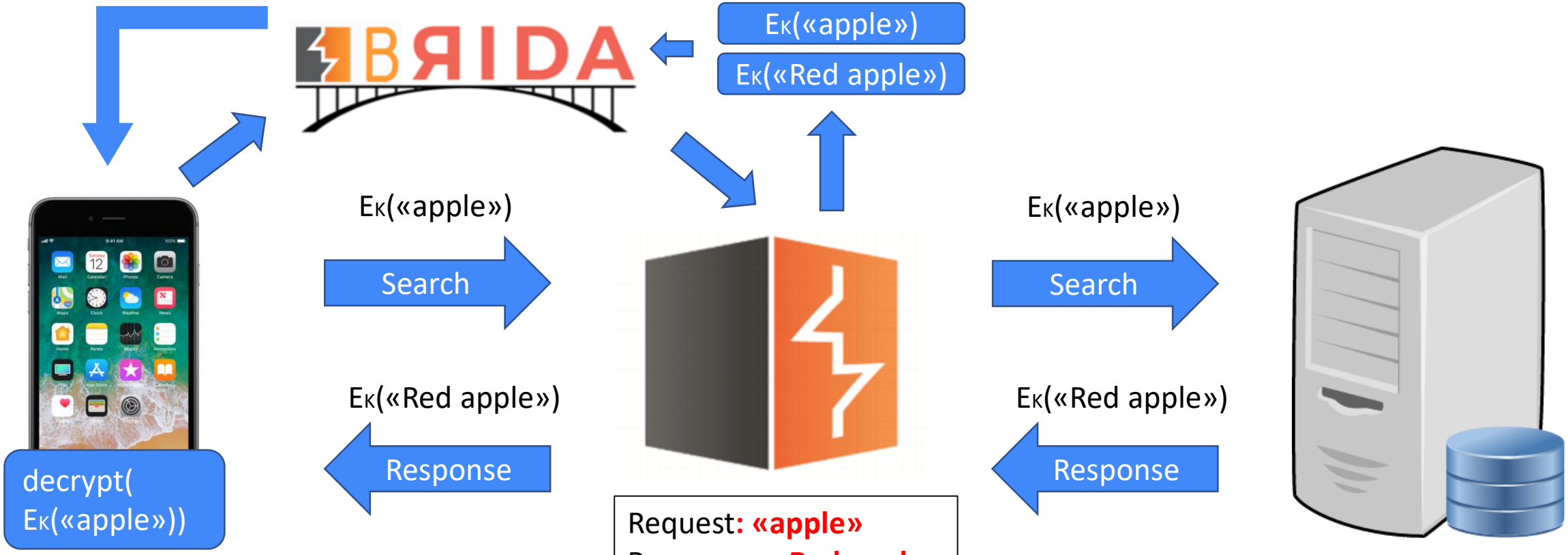
Search

Eκ(«Red apple»)

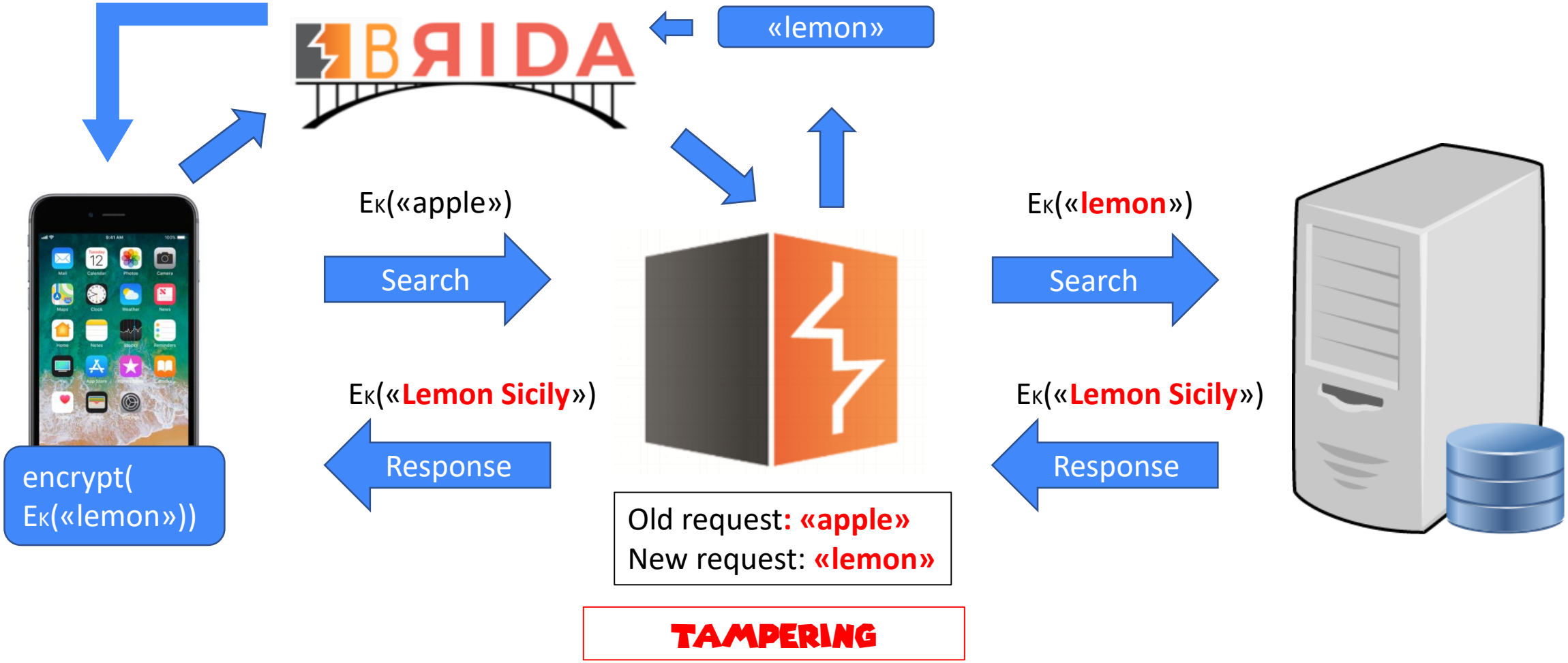
Response

NO INSPECTION - NO TAMPERING

USE CASE 1 - INSPECTION

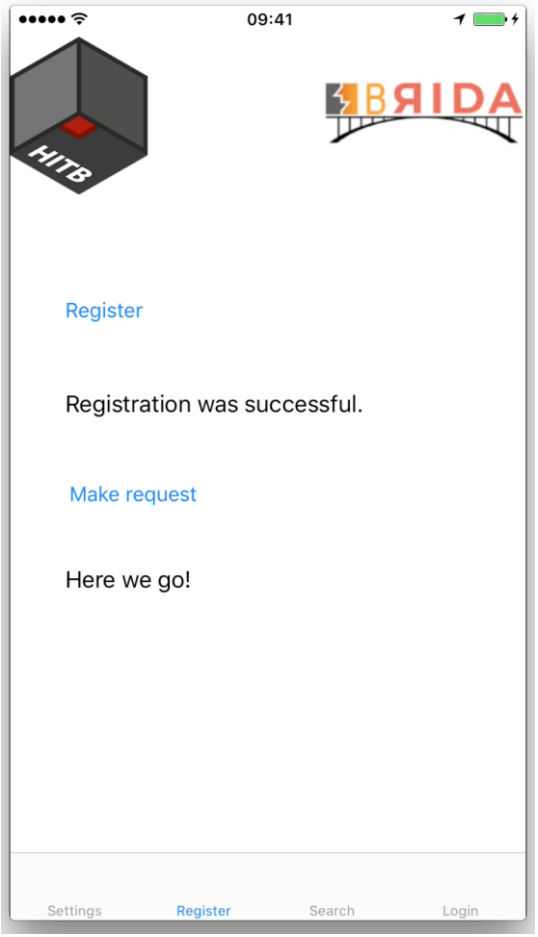


USE CASE 1 - TAMPERING



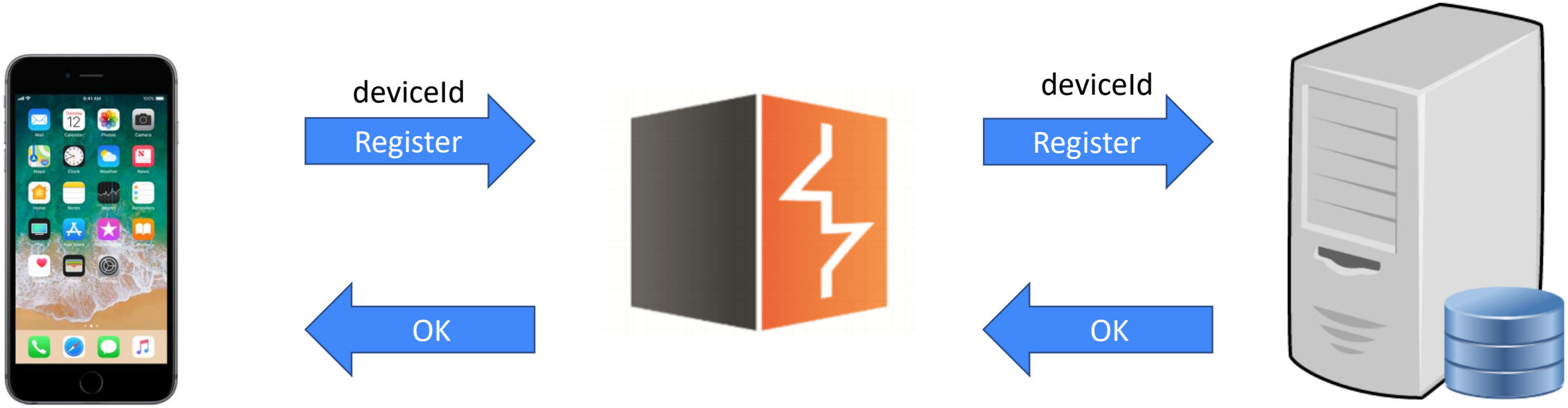
USE CASE 2

USE CASE 2



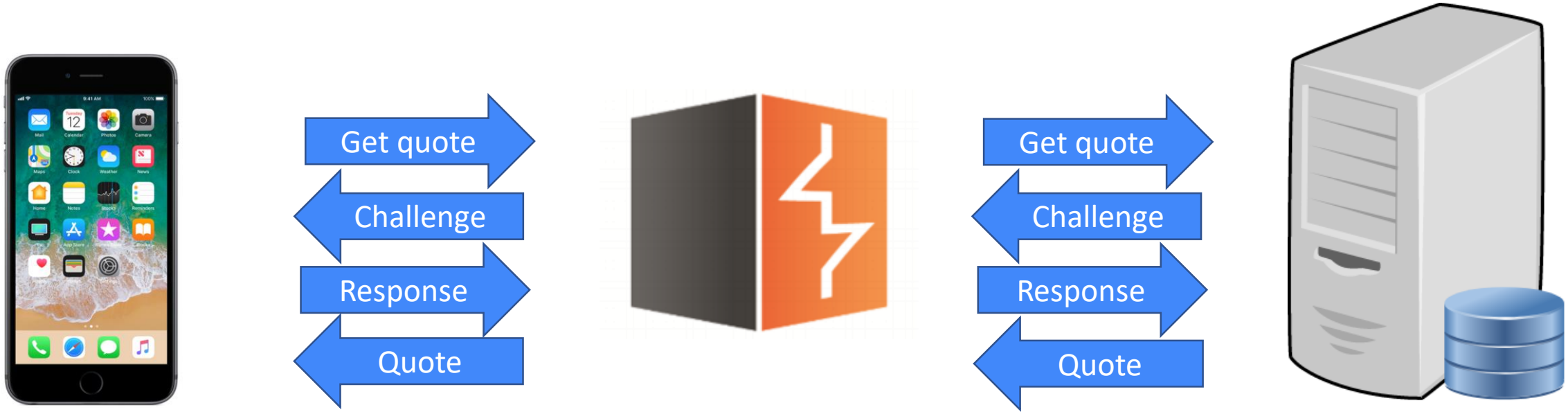
- We have a simple iOS app with two buttons: «Register» and «Make request»
- Once registered, by clicking on the «Make request» button it is possible to get a Super Mario quote!

USE CASE 2 - REGISTRATION



USE CASE 2 - MAKE REQUEST FIRST 10 REQUESTS...

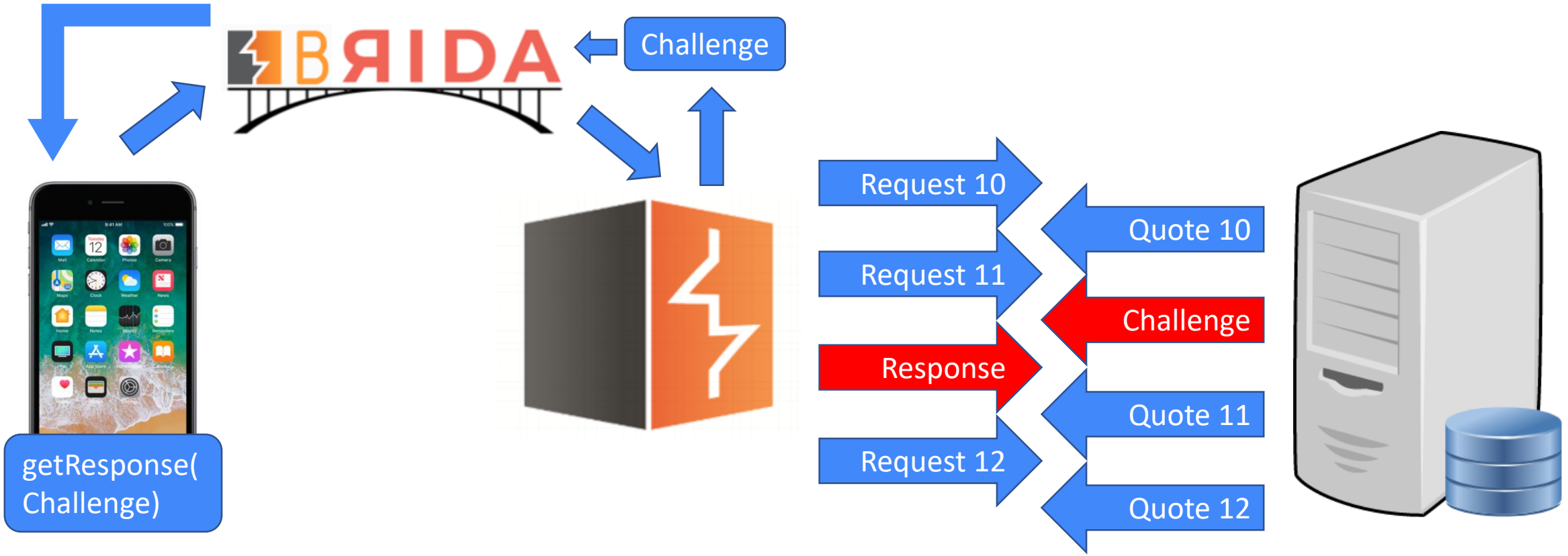
USE CASE 2 - MAKE REQUEST ...REQUEST 11



USE CASE 2 - SCANNER

NO SCANNER - NO INTRUDER

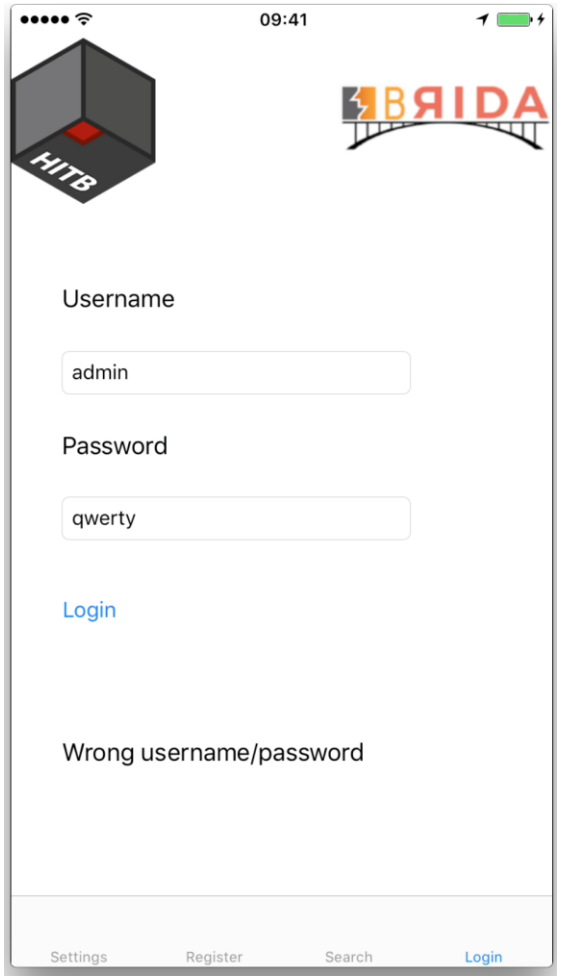
USE CASE 2 - SCANNER



OK SCANNER – OK INTRUDER

USE CASE 3

USE CASE 3



- We have a simple iOS app with a login form
- The application returns «User logged in» if the correct username and password are inserted, «Wrong username/password» otherwise

USE CASE 3

admin/Eκ(«password»)

Login

«User logged in»

Response

admin/Eκ(«password»)

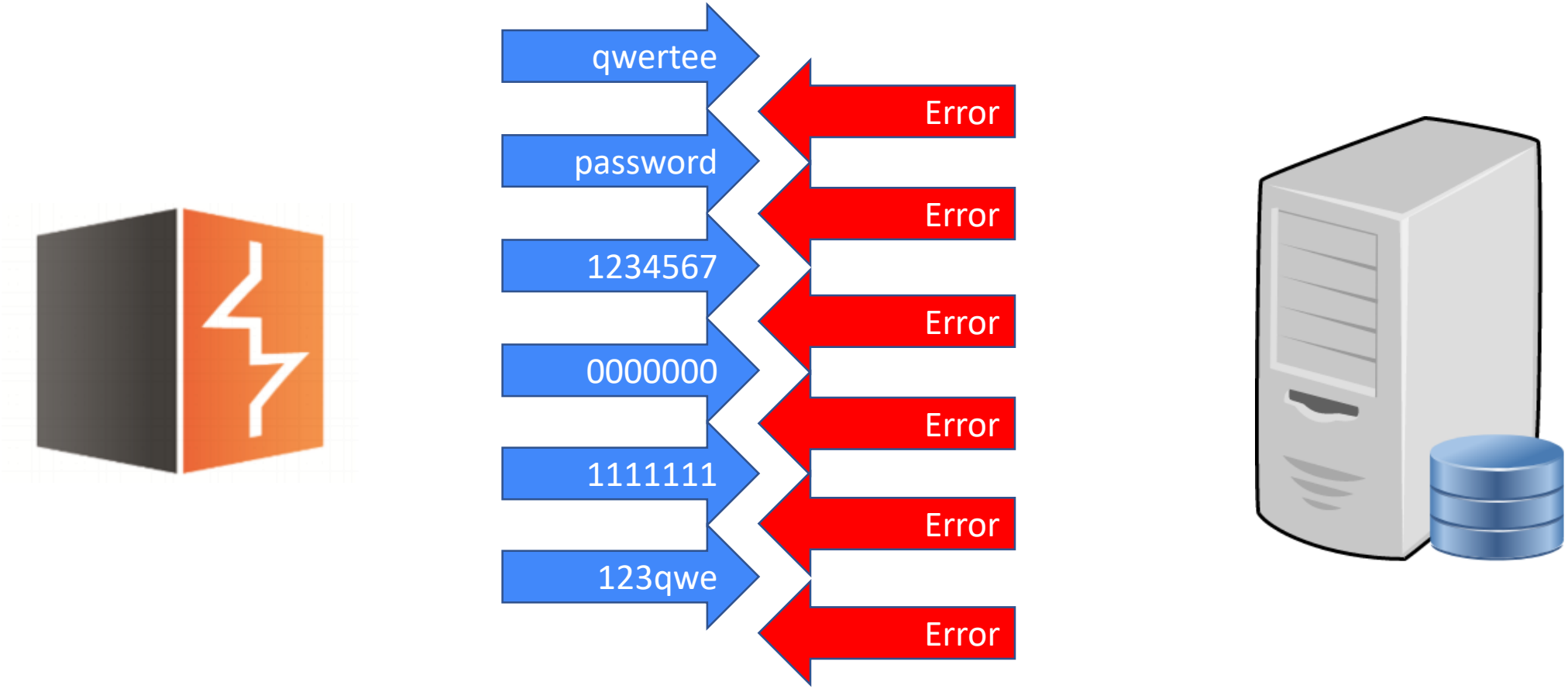
Login

«User logged in»

Response

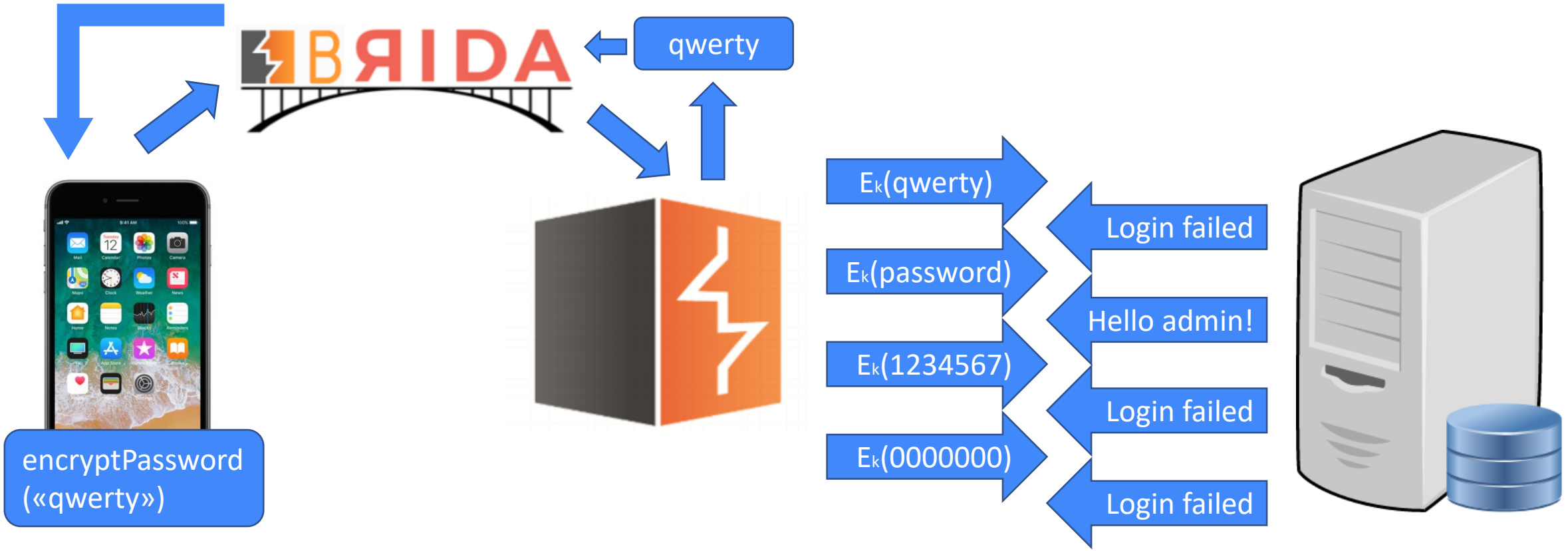
Request: admin/Eκ(«password»)

USE CASE 3 - INTRUDER



NO SCANNER - NO INTRUDER

USE CASE 3 - INTRUDER



OK SCANNER - OK INTRUDER

LINKS!

- Brida repo: https://github.com/federicodotta/Brida
- Brida releases: https://github.com/federicodotta/Brida/releases
- Burp Suite: https://portswigger.net/burp
- Frida: https://www.frida.re/
- Article that describes Brida (0.1): https://techblog.mediaservice.net/2017/07/brida-advanced-mobile-application-penetration-testing-with-frida/

THANKS!

ANY QUESTION?

FEEL FREE TO CONTACT US AT:
FEDERICO.DOTTA@MEDIASERVICE.NET
PIERGIOVANNI.CIPOLLONI@MEDIASERVICE.NET

CONGRATULATIONS MARIO!

AUTHORS

FEDERICO DOTTA - PIERGIOVANNI CIPOLLONI

REVIEW

MAURIZIO AGAZZINI - MARCO IVALDI

THANKS

MATTIA VINCI

LICENSE

CREATIVE COMMONS

Image list (all licensed under CC 4.0)

http://pngimg.com/download/30587

http://pngimg.com/download/30532

http://pngimg.com/download/30516

http://pngimg.com/download/30540

http://pngimg.com/download/30595

http://pngimg.com/download/30521

http://pngimg.com/download/30534

http://pngimg.com/download/30545

http://pngimg.com/download/30551

http://pngimg.com/download/30555

http://pngimg.com/download/30548

http://pngimg.com/download/30539

http://pngimg.com/download/30480

http://pngimg.com/download/30588

http://pngimg.com/download/30584

Disclaimer

This presentation is neither owned by nor affiliated with Nintendo or the creators of the Mario franchise in any way. Stuff related to Super Mario is copyrighted by Nintendo and other parties that have a direct proximity of relationship with the Mario franchise. The images used in this presentation are all released under Creative Commons 4.0 license.