

HITB SecConf

Amsterdam – April 12th, 2018

BRIDA

**WHEN BURP SUITE
MEETS FRIDA**



Piergiovanni Cipolloni

Security Advisor

@Mediaservice.net S.r.l.

(piergiovanni.cipolloni@mediaservice.net)

- ~ 20 years in Penetration Testing
- Security researcher



THE SPEAKERS



Federico Dotta

Security Advisor

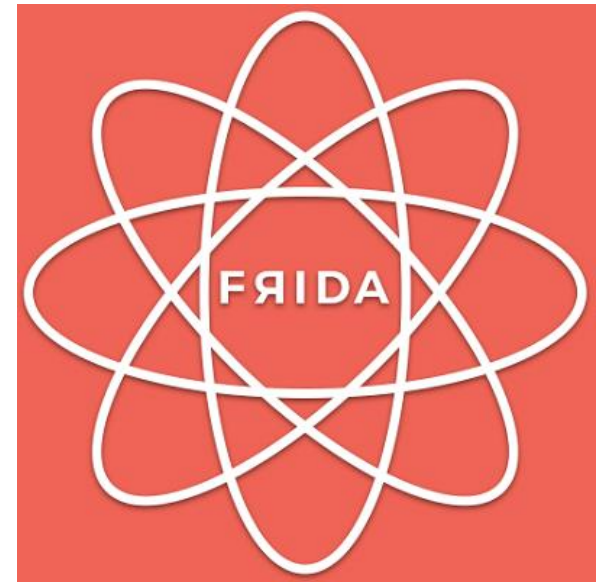
@Mediaservice.net S.r.l.

(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



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

Request

Raw Params Headers Hex

```
POST [REDACTED]
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
Content-Length: 89
Host: [REDACTED]
Connection: close

query=0105b79f12639e1f121ab41eb2df19dc3ff00104eyJzdGFpdFNlc3Npb24iOnsiaidmVyc2lvbiI6IjEuMy4wIn19
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain;charset=US-ASCII
Content-Length: 128
[REDACTED]
Connection: close

0117f69f12673e17121bb41e92df15dc30f00004eyJzdGFpdFNlc3Npb24iOnsiaiY3J5cHRvS2V5IjoieYmE4NGUxSTYlbnJkdkZlU0M2ExNzllMTUyNTc3MGRlNmQifX0=
```

Converted text

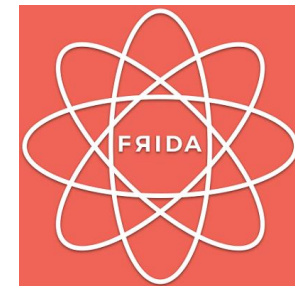
Copy to clipboard Close

0 matches

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.



ZERO NIGHTS

Cross-platform reversing with Frida

What is Frida?

- Dynamic instrumentation toolkit
- Debug live processes
- Scriptable
 - **Execute your own debug scripts inside another process**
- Multi-platform
 - Windows, Mac, Linux, iOS, Android, QNX
- Open Source

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

BEFORE FRIDA

Bytecode Viewer 2.9.11 - https://bytecodeviewer.com | https://thebytecode.club - @Konloch

File View Settings Plugins

Files

- FileDecrypter.class
- FileDecrypter.class

Work Space

FileDecrypter.class x

Bytecode Decompiler - Editable: false

```
583
584 public static decrypt(byte[] arg0, byte[] arg1, byte[] arg2) { //([B[B[B[B
585   <localVar:index=0, name=toDecrypt, desc=[B, sig=null, start=L1, end=L2>
586   <localVar:index=1, name=key, desc=[B, sig=null, start=L1, end=L2>
587   <localVar:index=2, name=initializationVector, desc=[B, sig=null, start=L1, end=L2>
588   <localVar:index=3, name=c, desc=Ljavax/crypto/Cipher;, sig=null, start=L3, end=L4>
589   <localVar:index=4, name=iv, desc=Ljavax/crypto/spec/IvParameterSpec;, sig=null, start=L5, end=L4>
590   <localVar:index=5, name=result, desc=[B, sig=null, start=L6, end=L4>
591   <localVar:index=3, name=e, desc=Ljava/lang/Exception;, sig=null, start=L7, end=L2>
592
593   TryCatch: L1 to L8 handled by L4: java/lang/Exception
594   L1 {
595     ldc "AES/CBC/PKCS5Padding" (java.lang.String)
596     invokestatic javax/crypto/Cipher.getInstance(Ljava/lang/String;)Ljavax/crypto/Cipher;
597     astore3
598   }
599   L3 {
600     new javax/crypto/spec/IvParameterSpec
601     dup
602     aload2
603     invokespecial javax/crypto/spec/IvParameterSpec.<init>([B)V
604     astore4
605   }
606   L5 {
607     aload3
608     iconst_2
609     new javax/crypto/spec/SecretKeySpec
610     dup
611     aload1 // reference to arg1
612     ldc "AES" (java.lang.String)
613     invokespecial javax/crypto/spec/SecretKeySpec.<init>([BLjava/lang/String;)V
614     aload4
615     invokevirtual javax/crypto/Cipher.init(ILjava/security/Key;Ljava/security/spec/AlgorithmParameterSpec;)V
616   }
617   L9 {
618     aload3
619     aload0 // reference to arg0
620     invokevirtual javax/crypto/Cipher.doFinal([B[B
621     astore5
622   }
623   L6 {
624     aload5
625   }
626   L8 {
627     areturn
628   }
629   L4 {
```



```
3 import java.lang.Object;
4 import java.util.*;
5 import java.util.Base64;
6 import javax.crypto.SecretKey;
7 import javax.crypto.*;
8 import javax.crypto.spec.*;
9 import javax.crypto.Cipher;
10 import java.security.Key;
11 import java.security.*;
12 import java.security.spec.InvalidKeySpecException;
13
14
15 public class jaDec {
16
17   public static void main(String[] args) {
18
19
20     String plainText = "*****";
21
22     byte[] bKey = new byte[] {(byte)0x01,(byte)0x02,(byte)0x03,(byte)0x04,(byte)0x05,(byte)0x06,(byte)0x07,(byte)0x08,(byte)0x09,(byte)0x0a,(byte)0x0b,(byte)0x0c,(byte)0x0d,(byte)0x0e,(byte)0x0f,(byte)0x10,(byte)0x11,(byte)0x12,(byte)0x13,(byte)0x14,(byte)0x15,(byte)0x16,(byte)0x17,(byte)0x18,(byte)0x19,(byte)0x1a,(byte)0x1b,(byte)0x1c,(byte)0x1d,(byte)0x1e,(byte)0x1f,(byte)0x20,(byte)0x21,(byte)0x22,(byte)0x23,(byte)0x24,(byte)0x25,(byte)0x26,(byte)0x27,(byte)0x28,(byte)0x29,(byte)0x2a,(byte)0x2b,(byte)0x2c,(byte)0x2d,(byte)0x2e,(byte)0x2f,(byte)0x30,(byte)0x31,(byte)0x32,(byte)0x33,(byte)0x34,(byte)0x35,(byte)0x36,(byte)0x37,(byte)0x38,(byte)0x39,(byte)0x3a,(byte)0x3b,(byte)0x3c,(byte)0x3d,(byte)0x3e,(byte)0x3f,(byte)0x40,(byte)0x41,(byte)0x42,(byte)0x43,(byte)0x44,(byte)0x45,(byte)0x46,(byte)0x47,(byte)0x48,(byte)0x49,(byte)0x4a,(byte)0x4b,(byte)0x4c,(byte)0x4d,(byte)0x4e,(byte)0x4f,(byte)0x50,(byte)0x51,(byte)0x52,(byte)0x53,(byte)0x54,(byte)0x55,(byte)0x56,(byte)0x57,(byte)0x58,(byte)0x59,(byte)0x5a,(byte)0x5b,(byte)0x5c,(byte)0x5d,(byte)0x5e,(byte)0x5f,(byte)0x60,(byte)0x61,(byte)0x62,(byte)0x63,(byte)0x64,(byte)0x65,(byte)0x66,(byte)0x67,(byte)0x68,(byte)0x69,(byte)0x6a,(byte)0x6b,(byte)0x6c,(byte)0x6d,(byte)0x6e,(byte)0x6f,(byte)0x70,(byte)0x71,(byte)0x72,(byte)0x73,(byte)0x74,(byte)0x75,(byte)0x76,(byte)0x77,(byte)0x78,(byte)0x79,(byte)0x7a,(byte)0x7b,(byte)0x7c,(byte)0x7d,(byte)0x7e,(byte)0x7f,(byte)0x80,(byte)0x81,(byte)0x82,(byte)0x83,(byte)0x84,(byte)0x85,(byte)0x86,(byte)0x87,(byte)0x88,(byte)0x89,(byte)0x8a,(byte)0x8b,(byte)0x8c,(byte)0x8d,(byte)0x8e,(byte)0x8f,(byte)0x90,(byte)0x91,(byte)0x92,(byte)0x93,(byte)0x94,(byte)0x95,(byte)0x96,(byte)0x97,(byte)0x98,(byte)0x99,(byte)0x9a,(byte)0x9b,(byte)0x9c,(byte)0x9d,(byte)0x9e,(byte)0x9f,(byte)0xa0,(byte)0xa1,(byte)0xa2,(byte)0xa3,(byte)0xa4,(byte)0xa5,(byte)0xa6,(byte)0xa7,(byte)0xa8,(byte)0xa9,(byte)0xaa,(byte)0xab,(byte)0xac,(byte)0xad,(byte)0xae,(byte)0xaf,(byte)0xb0,(byte)0xb1,(byte)0xb2,(byte)0xb3,(byte)0xb4,(byte)0xb5,(byte)0xb6,(byte)0xb7,(byte)0xb8,(byte)0xb9,(byte)0xba,(byte)0xbb,(byte)0xbc,(byte)0xbd,(byte)0xbe,(byte)0xbf,(byte)0xc0,(byte)0xc1,(byte)0xc2,(byte)0xc3,(byte)0xc4,(byte)0xc5,(byte)0xc6,(byte)0xc7,(byte)0xc8,(byte)0xc9,(byte)0xca,(byte)0xcb,(byte)0xcc,(byte)0xcd,(byte)0xce,(byte)0xcf,(byte)0xd0,(byte)0xd1,(byte)0xd2,(byte)0xd3,(byte)0xd4,(byte)0xd5,(byte)0xd6,(byte)0xd7,(byte)0xd8,(byte)0xd9,(byte)0xda,(byte)0xdb,(byte)0xdc,(byte)0xdd,(byte)0xde,(byte)0xdf,(byte)0xe0,(byte)0xe1,(byte)0xe2,(byte)0xe3,(byte)0xe4,(byte)0xe5,(byte)0xe6,(byte)0xe7,(byte)0xe8,(byte)0xe9,(byte)0xea,(byte)0xeb,(byte)0xec,(byte)0xed,(byte)0xee,(byte)0xef,(byte)0xf0,(byte)0xf1,(byte)0xf2,(byte)0xf3,(byte)0xf4,(byte)0xf5,(byte)0xf6,(byte)0xf7,(byte)0xf8,(byte)0xf9,(byte)0xfa,(byte)0xfb,(byte)0xfc,(byte)0xfd,(byte)0xfe,(byte)0xff};
23
24
25     SecretKeySpec originalKey;
26
27     Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
28
29     //originalKey = new SecretKeySpec(byteKey, 0, byteKey.length, "AES/CBC/PKCS7Padding");
30     originalKey = new SecretKeySpec(bKey, "AES/CBC/PKCS7Padding");
31
32     System.out.println("bKey len: " + bKey.length);
33
34     byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
35     IvParameterSpec ivspec = new IvParameterSpec(iv);
36
37     try {
38
39       //Cipher myCipherIn = Cipher.getInstance("AES/CTR/NoPadding", "BC");
40       Cipher myCipherIn = Cipher.getInstance("AES/CTR/NoPadding", "BC");
41       myCipherIn.init(Cipher.DECRYPT_MODE, originalKey, ivspec);
42       System.out.println("Cipher Init");
43       //String decryptText = new String(myCipherIn.doFinal(plainText.getBytes("UTF-8")));
44       //String decryptText = new String(myCipherIn.doFinal(Base64.getDecoder().decode(plainText)));
45       String decryptText = new String(myCipherIn.doFinal(Base64.getDecoder().decode(plainText.getBytes("UTF-8"))));
46       System.out.println("Len: " + decryptText.length());
47       System.out.println("Decrypted Text: " + decryptText);
48
49     } catch (Exception ex1)
50     {
51       System.out.print(ex1.toString());
52     }
53
54 }
55
```

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



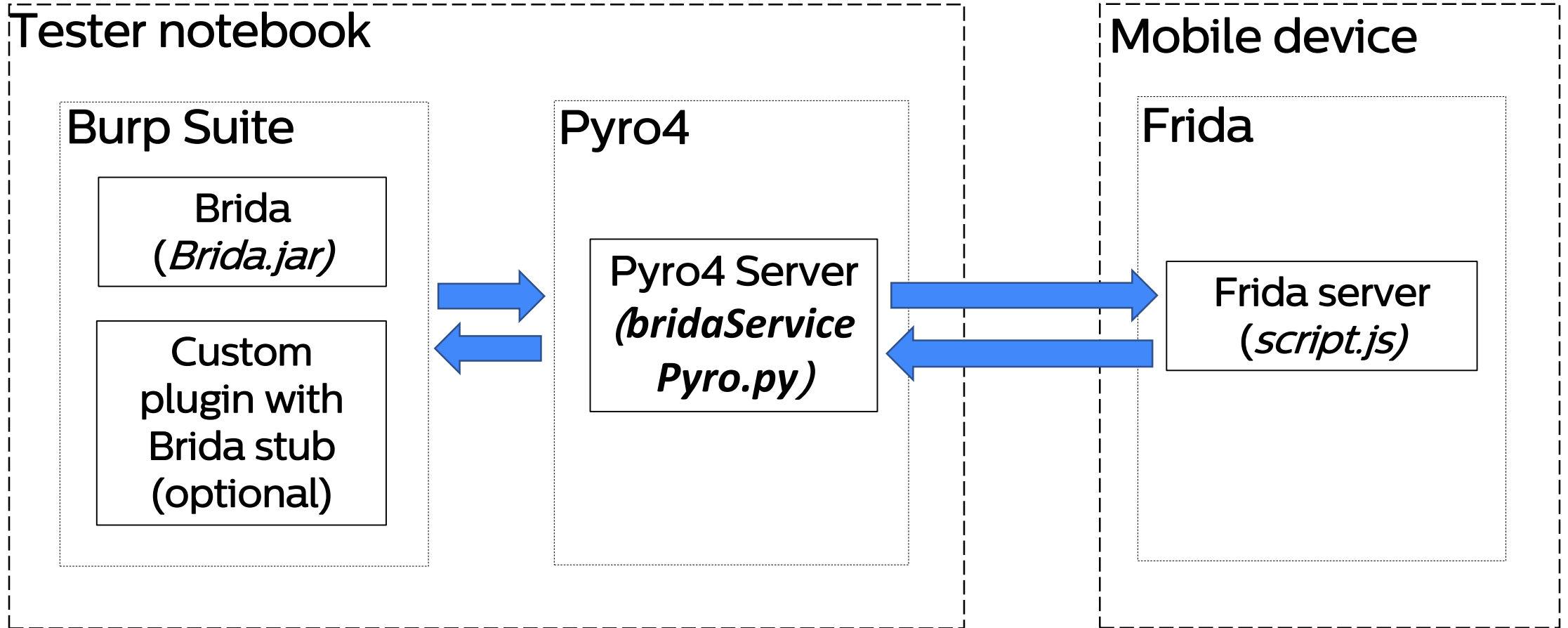
**WHY DON'T WE LET THE MOBILE
APP DO THE DIRTY WORK FOR
US?**





- 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

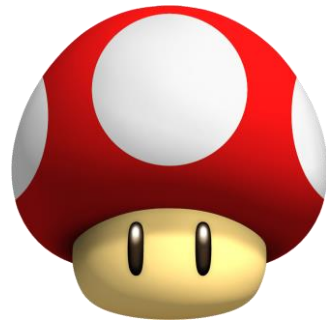
Stub generator Execute method

Method name: encryptbody

Argument: {"username":"test","password":"testPassword"} Add

Argument list: Remove Modify

Kill application
Reload JS
Java Stub
Python Stub
Save settings to file
Load settings from file
Execute Method



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



BRIDA 0.1 - CONTEXT MENU

Request

Raw Hex

POST /login HTTP/1.1
Host: www.test.com

parameters=djshfjdsvcxuchvjsdbfvjbjfndakfdshf
kdnfjnjvxcnjkansdjksncxjndjskjendjshfjdsvcxuc
jsfjdanj fnds jncjxknj skdnfjnjvxcnjkansdjksncxj
jbjfndakfdshfcjxnnvdfjsfjdanj fnds jncjxknj skdn
djshfjdsvcxuchvjsdbfvjbjfndakfdshfcjxnnvdfjsf
jkansdjksncxjndjskjc n%3d%3d

- Send to Spider
- Do an active scan
- Send to Intruder
- Send to Repeater
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser
- Insert collaborator p
- Insert collaborator i
- Brida Custom 1**
- Brida Custom 2

```
contextcustom2: function(message) {  
    var res = ObjC.classes.SampleClass.generatePostBody(message);  
    return res.toString();  
},
```



Request

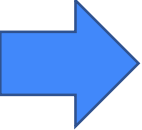
Raw Hex

POST /login HTTP/1.1
Host: www.test.com

parameters={"username": "test", "password": "a"}

- Send to Spider
- Do an active scan
- Send to Intruder
- Send to Repeater
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser
- Insert collaborator p
- Insert collaborator i
- Brida Custom 1
- Brida Custom 2**

```
contextcustom1: function(message) {  
    var res = ObjC.classes.SampleClass.getClearTextMessage(message);  
    return res.toString();  
},
```



BRIDA 0.1 - STUB GENERATOR

The screenshot shows the 'Stub generator' tab of the Brida 0.1 interface. The code editor contains the following Java code:

```
import net.razorvine.pyro.*;

String pyroUrl = "PYRO:BridaServicePyro@localhost:9999";
try {
    PyroProxy pp = new PyroProxy(new PyroURI(pyroUrl));
    String ret = (String)pp.call("callexportfunction", "METHOD_NAME", new String[]{"METHOD_ARG_1", "METHOD_ARG_2", "..."});
    pp.close();
} catch(IOException e) {
    // EXCEPTION HANDLING
}
```

On the right side, a vertical stack of buttons is visible. The 'Java Stub' button is highlighted with a red rectangle. Other buttons include 'Kill application', 'Reload JS', 'Python Stub', 'Save settings to file', 'Load settings from file', and 'Execute Method'.

The screenshot shows the 'Stub generator' tab of the Brida 0.1 interface. The code editor contains the following Python code:

```
import Pyro4

uri = 'PYRO:BridaServicePyro@localhost:9999'
pp = Pyro4.Proxy(uri)
args = []
args.append("METHOD_ARG_1")
args.append("METHOD_ARG_2")
args.append("...")
ret = pp.callexportfunction('METHOD_NAME', args)
pp._pyroRelease()
```

On the right side, a vertical stack of buttons is visible. The 'Python Stub' button is highlighted with a red rectangle. Other buttons include 'Kill application', 'Reload JS', 'Java Stub', 'Save settings to file', 'Load settings from file', and 'Execute Method'.

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

```
1 'use strict';
2
3 var destNum;
4
5 // 1 - FRIDA EXPORTS
6
7 rpc.exports = {
8
9   // Function executed when executed Brida contextual menu option 1. It transforms a string in lower case.
10  // Input: input string ENCODED IN ASCII HEX
11  // Output: lowercase string ENCODED IN ASCII HEX
12  contextcustom1: function(message) {
13    var a1 = ObjC.classes.NSString.stringWithString_(hexToString(message));
14    var a2 = a1.lowercaseString();
15    return stringToHex(a2.toString());
16  },
17
18  // Function executed when executed Brida contextual menu option 2. It encodes input in Base64.
19  // Input: input data ENCODED IN ASCII HEX
20  // Output: output Base64 string ENCODED IN ASCII HEX
21  contextcustom2: function(message) {
22    var inputByte = hexToBytes(message);
23    var ptrMessage = Memory.alloc(inputByte.length);
24    Memory.writeByteArray(ptrMessage, inputByte);
25    var objMessage = ObjC.classes.NSData.alloc().initWithBytes_length_(ptrMessage, inputByte.length);
26    var encodedMessage = objMessage.base64EncodedString();
27    return stringToHex(encodedMessage.toString());
28  },
29
30  // Function executed when executed Brida contextual menu option 3. It transforms a string in upper case.
31  // Input: input string ENCODED IN ASCII HEX
```

BRIDA 0.2 - FRIDA CONSOLE

The screenshot displays the BRIDA 0.2 web interface. At the top, there are tabs for various tools: Target, Proxy, Spider, Scanner, Intruder, Repeater, Sequencer, Decoder, Comparer, Extender, Project options, User options, Alerts, Logger++, and Brida. Below these are sub-tabs for Configurations, JS Editor, Analyze binary, Generate stubs, Execute method, and Trap methods.

The main configuration area shows the following settings:

- Server status: **running**
- Application status: **spawned**
- Python binary path: C:\python27\python
- Pyro host: localhost
- Pyro port: 9999
- Frida JS file path: o:\Materiale\Brida\Demo\Presentazione\Ricerca\scriptBrida\Ricerca.js
- Application ID: org.hitb.BridaDemo
- Mode: Frida Remote Frida Local

On the right side, there is a control panel with the following status and buttons:

- Server running
- App running
- Start server
- Kill server
- Spawn application
- Kill application
- Reload JS
- Clear console
- Save settings to file
- Load settings from file

The console log at the bottom, highlighted with a red border, shows the following output:

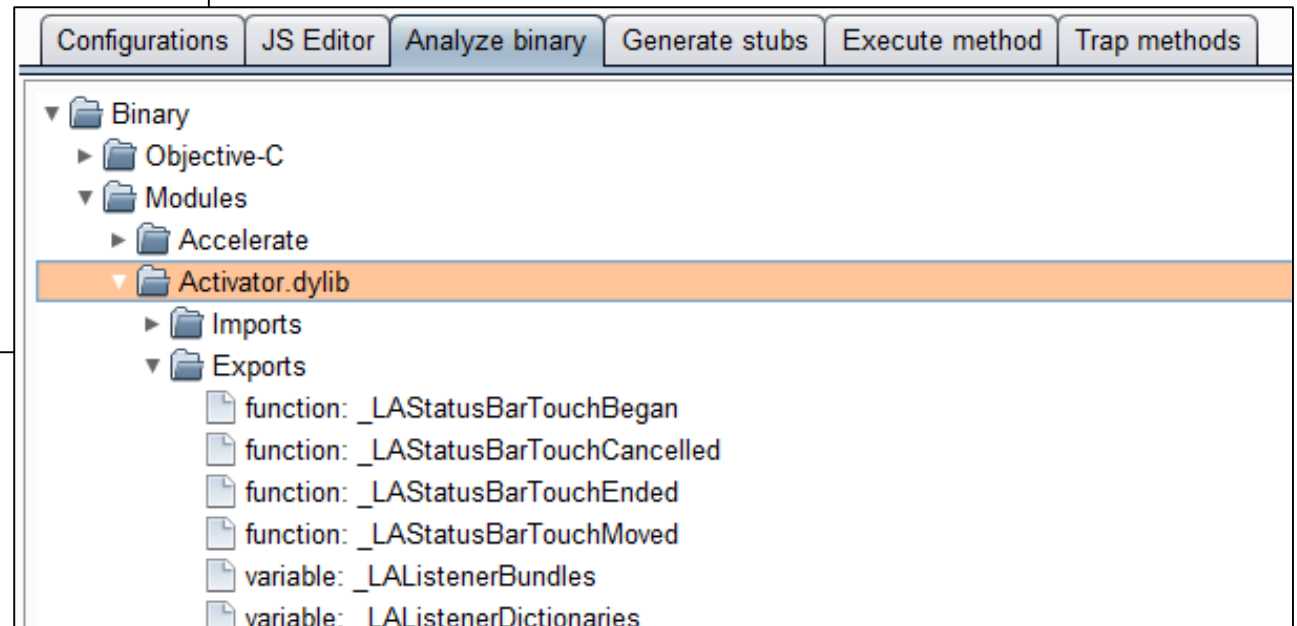
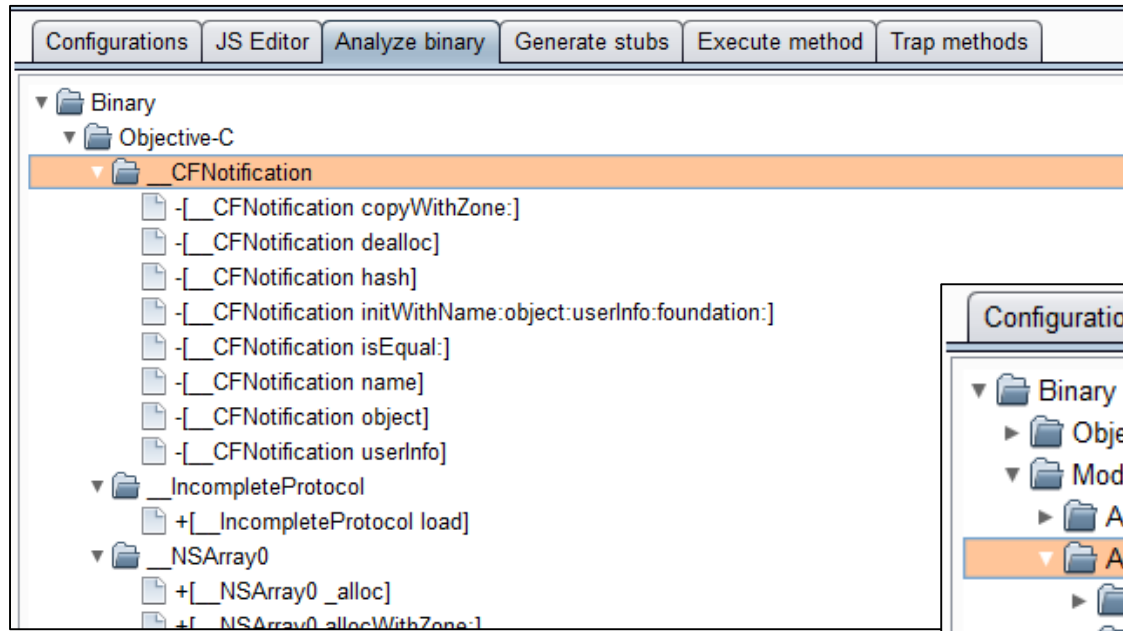
```
*** entered +[Encryption encryptRequest:]
Caller: 0x1000fc640 Hello Fede!_T010Hello_Fede18EncryptionRequestsC20sendEncryptedMessageySS7message_tF
Parameters:
  (NSTaggedPointerString) encryptRequest: phone
*** exiting +[Encryption encryptRequest:]
Return value:
  (_NSCFString) retval: FDhJA/MfD9GcEC+e0D+7Eg==
*** entered +[Encryption decryptResponse:]
Caller: 0x1000f2368 Hello Fede!_T010Hello_Fede21EncryptViewControllerCl8onResponseReceivedySS09encryptedG0_tF
Parameters:
  (_NSCFString) decryptResponse: bs91DnYe91qPCWt5Wv/MILMdHra9AvRXy0zEnUxXM34=
Backtrace:
0x1000f2368 Hello Fede!_T010Hello_Fede21EncryptViewControllerCl8onResponseReceivedySS09encryptedG0_tF
0x1000fd6b0 Hello Fede!_T010Hello_Fede18EncryptionRequestsC20sendEncryptedMessageySS7message_tFy10Foundation4DataVSg_SollURLResponseCSgs5Error_pSgtcFU_ycfU_
0x1000f012c Hello Fede!_T0Ix_IyB_TR
0x183639200 libdispatch.dylib!_dispatch_call_block_and_release
0x1836391c0 libdispatch.dylib!_dispatch_client_callout
```



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

```
Search: decrypt Search

**** Result of the search of decrypt
OBJC: +[Encryption decryptResponse:]
OBJC: +[FWEncryptorAES decrypt:Key:IV:]
OBJC: +[FWEncryptorAES decryptFromBase64:Key:IV:]
OBJC: +[FWEncryptorAES decryptStrFromBase64:Key:IV:]
OBJC: -[Hello_Fede.EncryptViewController decryptButton]
OBJC: -[Hello_Fede.EncryptViewController decryptedReceivedM
OBJC: -[Hello_Fede.EncryptViewController decryptMessage:]
OBJC: -[NSData decryptedAES256DataUsingKey:error:]
OBJC: -[NSData decryptedCASTDataUsingKey:error:]
OBJC: -[NSData decryptedDataUsingAlgorithm:key:error:]
OBJC: -[NSData decryptedDataUsingAlgorithm:key:initializati
```

```
Search: decrypt Search

IMPORT: /usr/lib/system/libcorecrypto.dylib!ccpad_xts_decrypt
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrc2_cbc_decrypt_mode
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrc2_cfb8_decrypt_mode
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrc2_cfb_decrypt_mode
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrc2_ecb_decrypt_mode
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrsa_decrypt_eme_pkcs1v15
IMPORT: /usr/lib/system/libcorecrypto.dylib!ccrsa_decrypt_oaep
EXPORT: /usr/lib/libcoretls.dylib!tls_record_decrypt
EXPORT: /usr/lib/libcoretls.dylib!tls_record_decrypted_size
EXPORT: /usr/lib/libSystem.B.dylib!ccaes_cbc_decrypt_mode
EXPORT: /usr/lib/libSystem.B.dylib!ccaes_ccm_decrypt_mode
EXPORT: /usr/lib/libSystem.B.dylib!ccaes_cfb8_decrypt_mode
```



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

The screenshot shows the Frida class hierarchy. The 'Encryption' class is selected and highlighted in orange. A context menu is open over it, showing three options: 'Inspect!', 'Inspect with backtrace!', and 'Change return value'. Below the class list is a search bar with the text 'Search:'.

```
Pyro server started correctly
Application org.hitb.BridaDemo spawned correctly
Platform: iOS

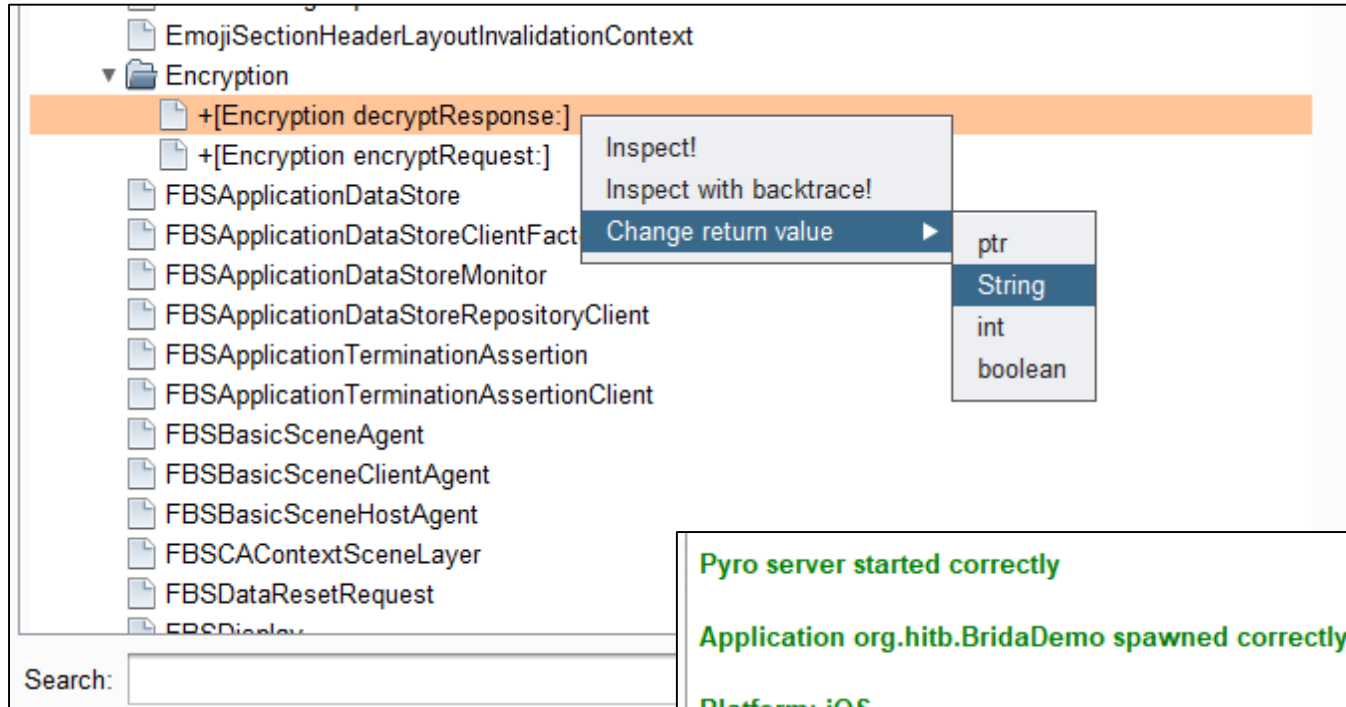
*** Tracing +[Encryption encryptRequest:]
*** Tracing +[Encryption decryptResponse:]
*** entered +[Encryption encryptRequest:]
Caller: 0x1000fc640 Hello Fede!_T010Hello_Fede!8EncryptionRequestsC20sendEncryptedMessageySS7message_tF
Parameters:
    (NSTaggedPointerString) encryptRequest: phone
```

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

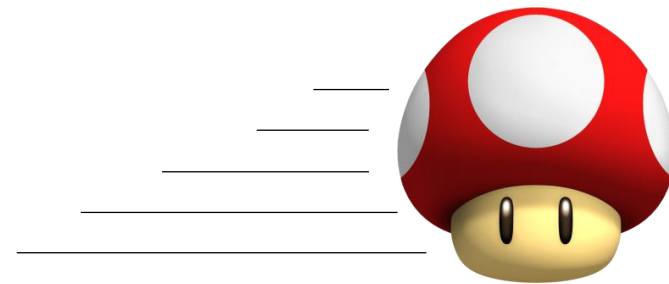


LET'S SEE HOW TO
HANDLE THESE
SITUATIONS IN THE
DEMO!



DEMO APPLICATION

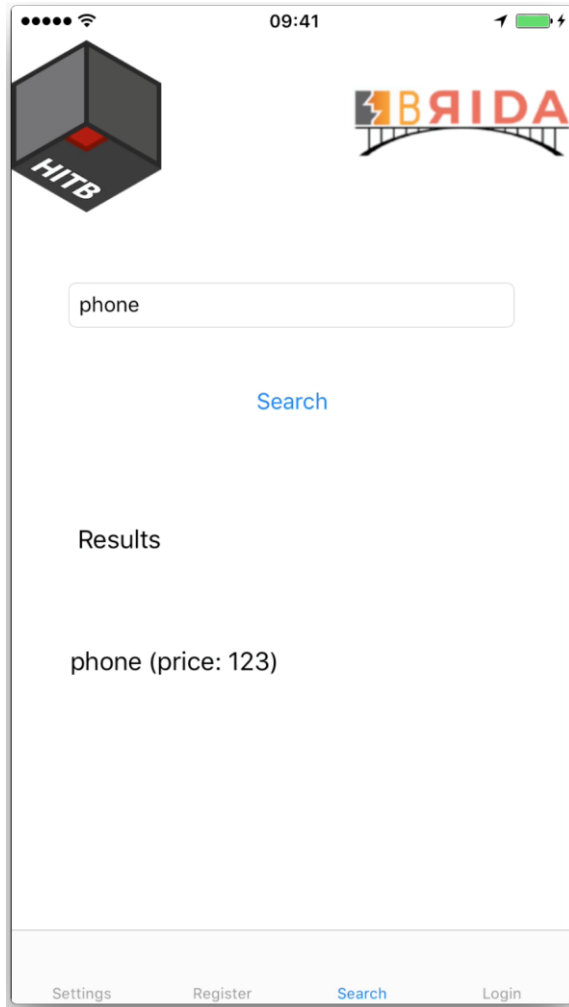
- Three different use cases
- Each use case is a simplification of a real situation we faced during penetration tests conducted on mobile applications
- In all those situations Brida was almost essential



USE CASE 1

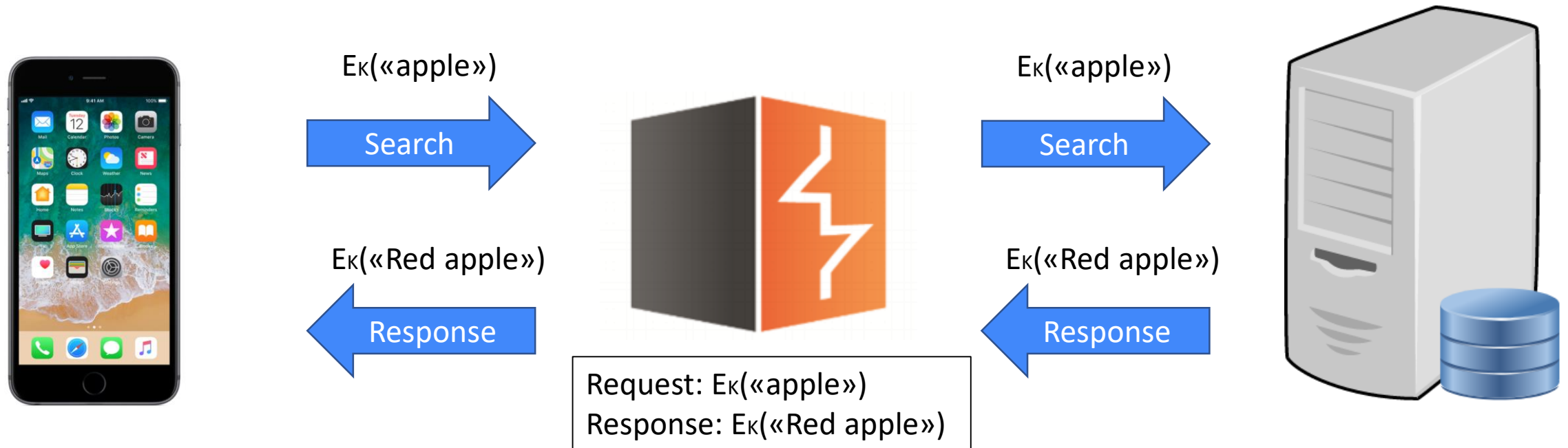


USE CASE 1



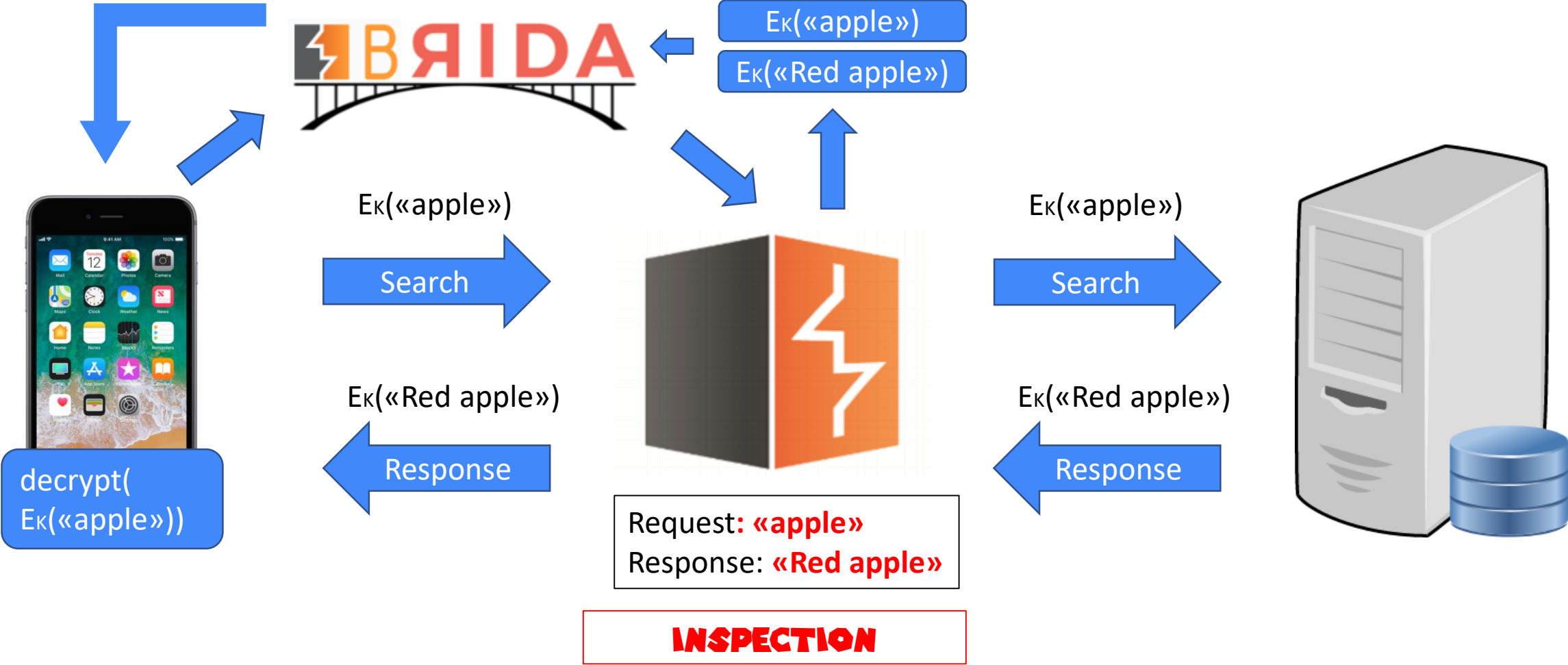
- We have a simple iOS app that provides a search functionality
- If we click on the «Search» button, the results are printed below the search form

USE CASE 1

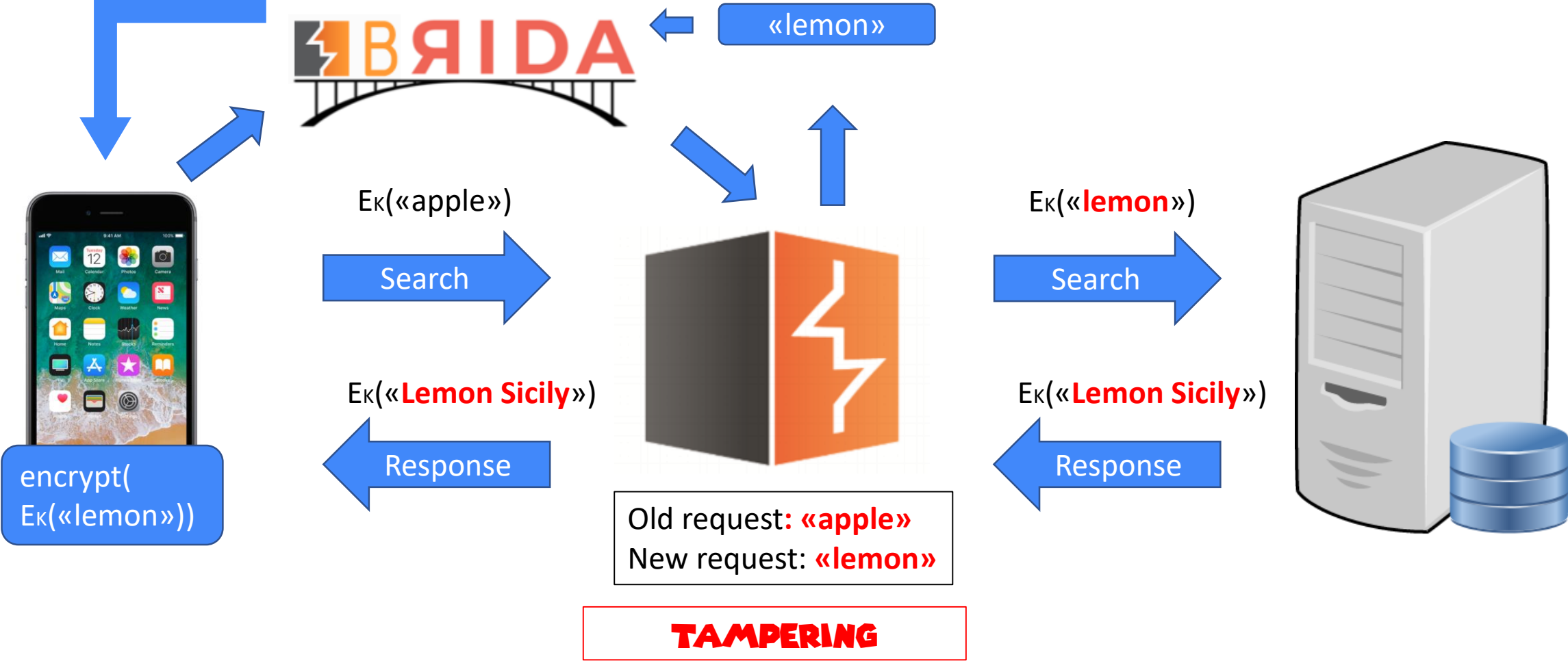


NO INSPECTION - NO TAMPERING

USE CASE 1 - INSPECTION



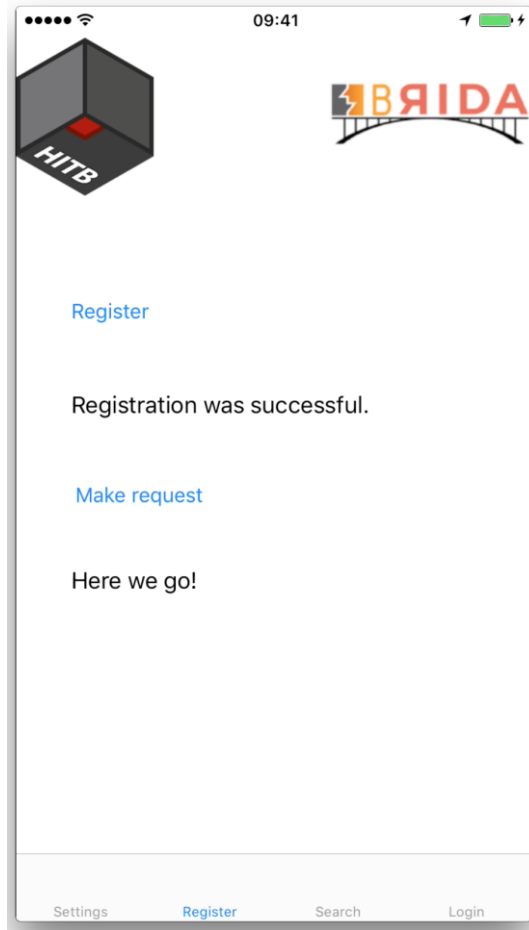
USE CASE 1 - TAMPERING



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



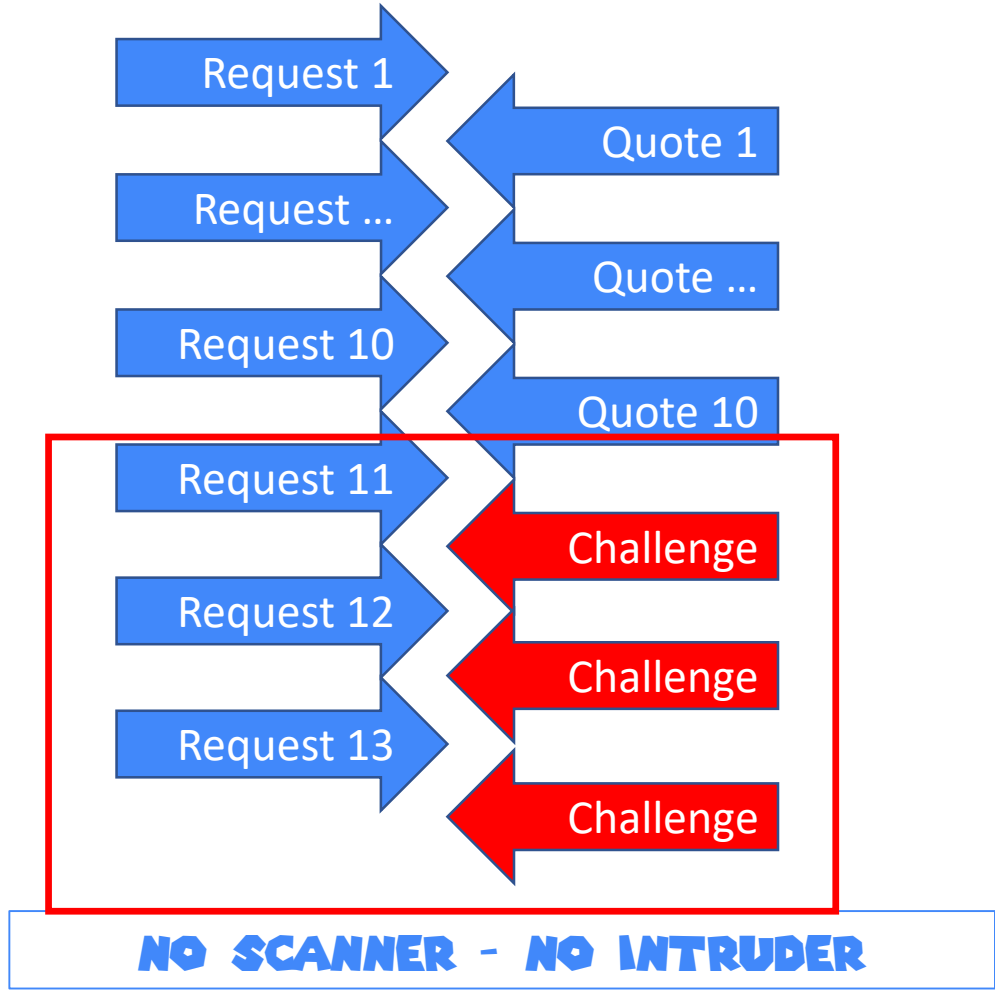
USE CASE 2 - MAKE REQUEST FIRST 10 REQUESTS...



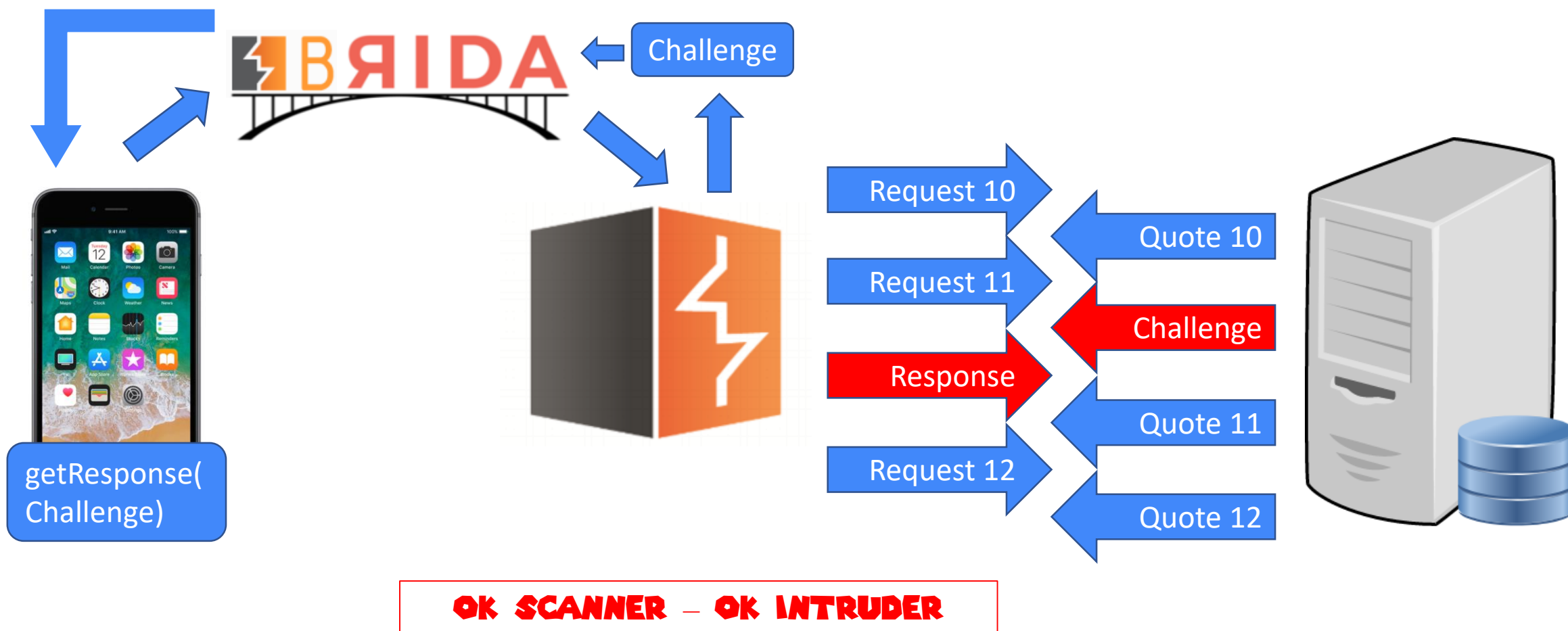
USE CASE 2 - MAKE REQUEST ...REQUEST 11



USE CASE 2 - SCANNER



USE CASE 2 - SCANNER

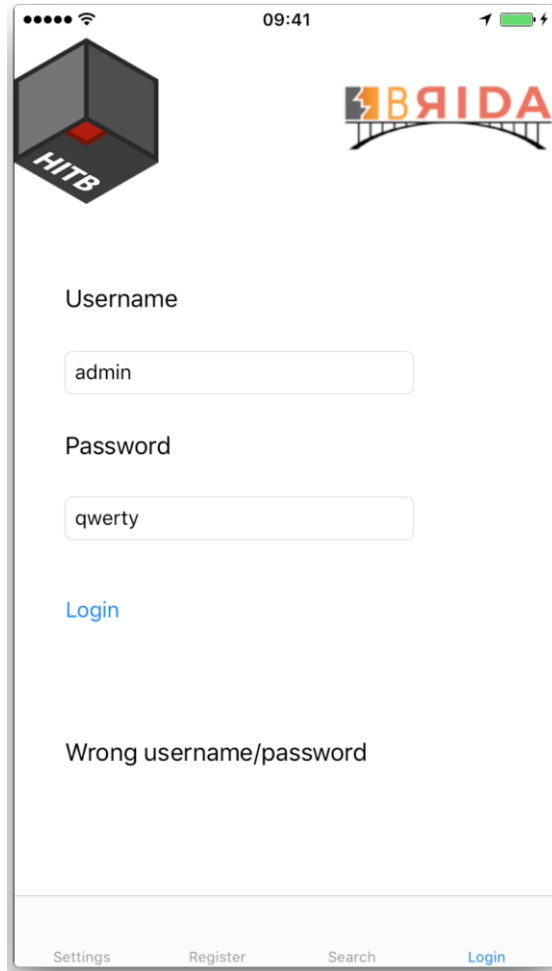


OK SCANNER – OK INTRUDER

USE CASE 3

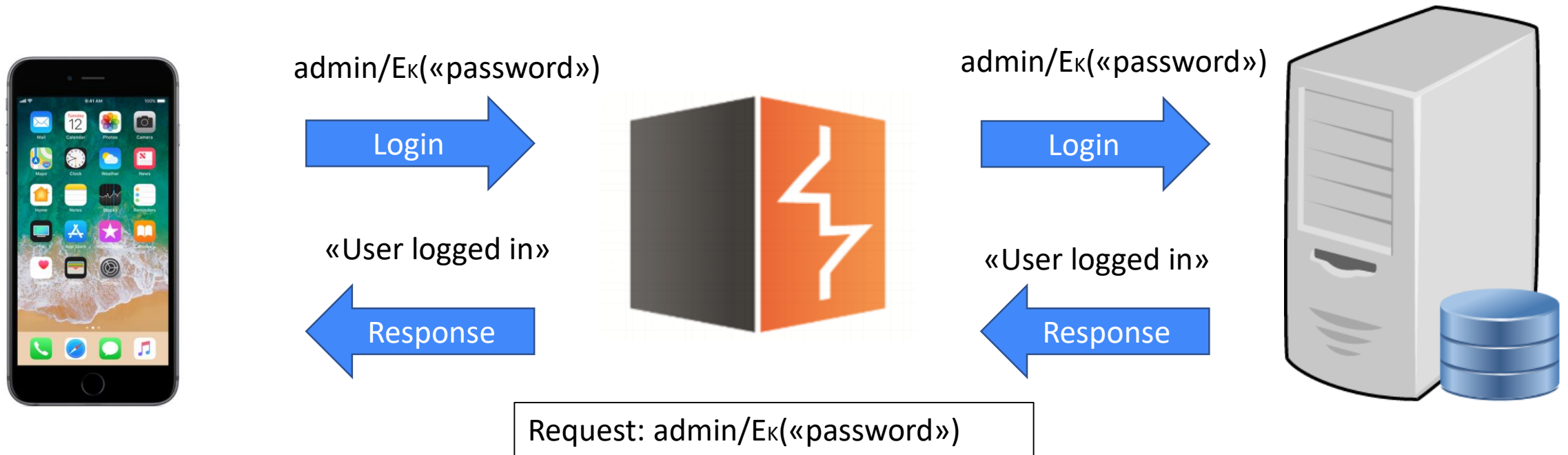


USE CASE 3

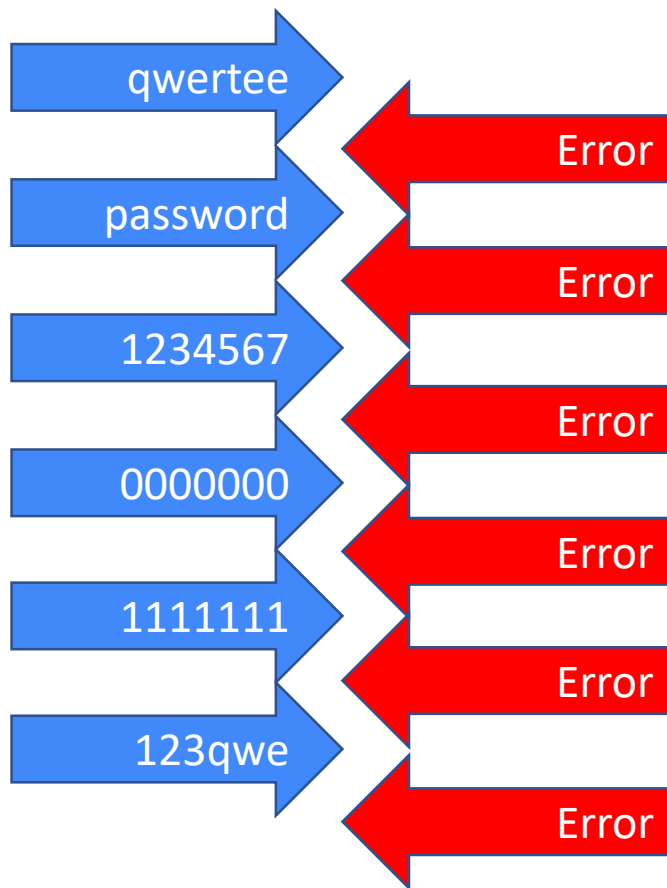


- We have a simple iOS app with a login form
- The application returns «User logged in» if the correct username and password are inserted, «Wrong username/password» otherwise

USE CASE 3

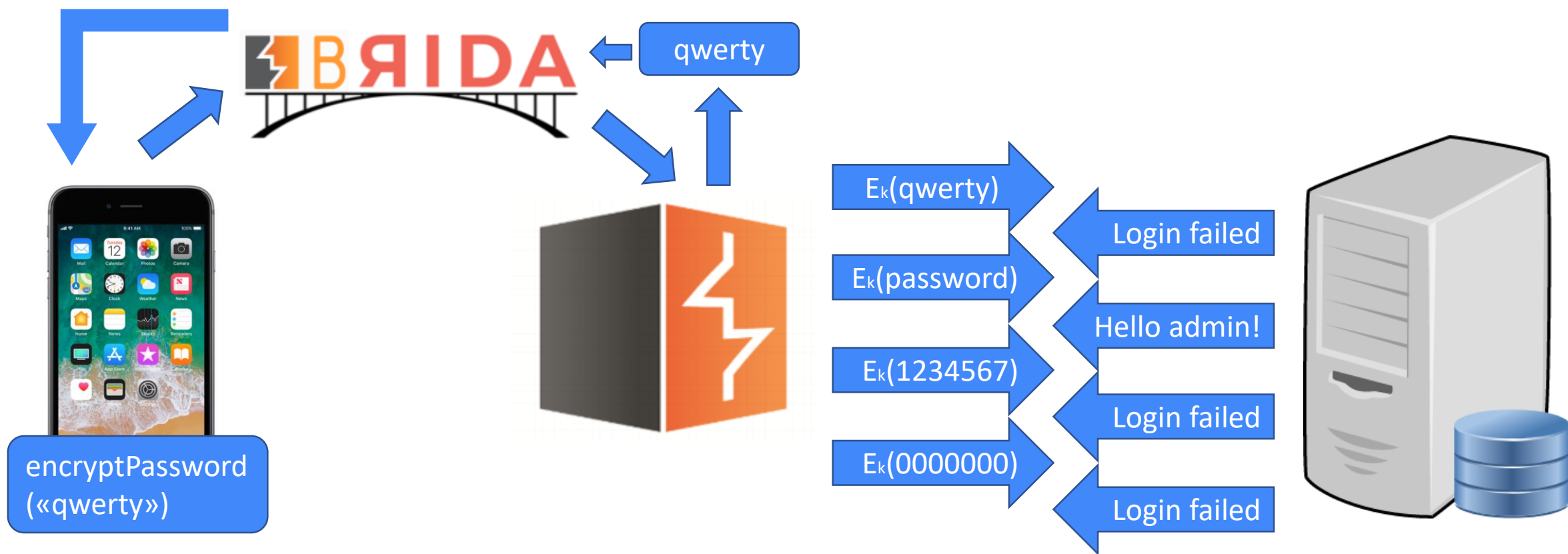


USE CASE 3 - INTRUDER



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/30587)

[HTTP://PNGIMG.COM/DOWNLOAD/30532](http://PNGIMG.COM/DOWNLOAD/30532)

[HTTP://PNGIMG.COM/DOWNLOAD/30516](http://PNGIMG.COM/DOWNLOAD/30516)

[HTTP://PNGIMG.COM/DOWNLOAD/30540](http://PNGIMG.COM/DOWNLOAD/30540)

[HTTP://PNGIMG.COM/DOWNLOAD/30595](http://PNGIMG.COM/DOWNLOAD/30595)

[HTTP://PNGIMG.COM/DOWNLOAD/30521](http://PNGIMG.COM/DOWNLOAD/30521)

[HTTP://PNGIMG.COM/DOWNLOAD/30534](http://PNGIMG.COM/DOWNLOAD/30534)

[HTTP://PNGIMG.COM/DOWNLOAD/30545](http://PNGIMG.COM/DOWNLOAD/30545)

[HTTP://PNGIMG.COM/DOWNLOAD/30551](http://PNGIMG.COM/DOWNLOAD/30551)

[HTTP://PNGIMG.COM/DOWNLOAD/30555](http://PNGIMG.COM/DOWNLOAD/30555)

[HTTP://PNGIMG.COM/DOWNLOAD/30548](http://PNGIMG.COM/DOWNLOAD/30548)

[HTTP://PNGIMG.COM/DOWNLOAD/30539](http://PNGIMG.COM/DOWNLOAD/30539)

[HTTP://PNGIMG.COM/DOWNLOAD/30480](http://PNGIMG.COM/DOWNLOAD/30480)

[HTTP://PNGIMG.COM/DOWNLOAD/30588](http://PNGIMG.COM/DOWNLOAD/30588)

[HTTP://PNGIMG.COM/DOWNLOAD/30584](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.

