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
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
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

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/plain; charset=UTF-8
Content-Length: 62
Connections: close

query=0105b79c1a639e1fe1ea416f28ff9e001c4c9e352f28b430b4241c9e1d04252e26e38c3e10e328c121

Response

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/x-www-form-urlencoded
Content-Length: 90
Connections: close

query=0105b79c1a639e1fe1ea416f28ff9e001c4c9e352f28b430b4241c9e1d04252e26e38c3e10e328c121

Copy to clipboard

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 an 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.
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

Before Frida

public class jaden {
    public static void main(String[] args) {
        String plainText = "HELLO WORLD"
        byte[] key = new byte[] {0x01, 0x02, 0x03, 0x04}
        SecretKeySpec originalKey;
        Security.addProvider(new org.bouncycastle.jce.provider.BouncyCastleProvider());
        // AES/CBC/PKCS7Padding
        SecretKeySpec newKey = new SecretKeySpec(key, "AES/CBC/PKCS7Padding");
        System.out.println("My Key = " + newKey.getKey().getEncoded());
        try {
            Cipher myCipher = Cipher.getInstance("AES/ECB/PKCS5Padding");
            myCipher.init(Cipher.ENCRYPT_MODE, originalKey,GOODS); // GOODS is an object
            byte[] encryptedText = myCipher.doFinal(plainText.getBytes());
            byte[] decryptedText = myCipher.doFinal(encryptedText);
            System.out.println("Cipher Text = " + new String(decryptedText, "UTF-8"));
            String decryptedText = new String(decryptedText, "UTF-8");
            System.out.println("Decrypted Text = " + decryptedText);
        }
    }
}

10

F. Dotta and P. Cipolloni – BRIDA: When Burp Suite Meets Frida
```javascript
var encryptionClass = Java.use("a.b.b");

encryptionClass.decryptMethod.overload("java.lang.String").implementation = function(cypherText) {
    var plainText = this.decryptMethod.overload("java.lang.String").call(this, cypherText);
    console.log(plainText);
    return plainText;
};
```
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
Brida (Brida.jar) - Custom plugin with Brida stub (optional)

Pyro4
- Pyro4 Server (bridaService Pyro.py)

Frida
- Frida server (script.js)

Tester notebook
- Burp Suite

Mobile device
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
• 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

Method name: encryptbody
Argument: {
  "username": "test",
  "password": "testPassword"
}

rpc.exports = {
  encryptbody: function(body) {
    var res = ObjC.classes.SampleClass.generatePostBody(body);
    return res.toString();
  }
}
BRIDA 0.1 - CONTEXT MENU

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

```
contextcustom1: function(message) {
    var res = Obj.classes.SampleClass.getClearTextMessage(message);
    return res.toString();
}
```
**BRIDA 0.1 - STUB GENERATOR**

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

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

```python
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.callexporttfunction("METHOD_NAME",args)
pp.close()
```
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     context_custom1: function(message) {
13         var a1 = ObjC.classesNSString.stringWithString_hexToHexString(message);
14         var a2 = a1.toLowerCaseString();
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     context_custom2: function(message) {
22         var inputBytes = hexToBytes(message);
23         var ptrMessage = Memory.alloc(inputBytes.length);
24         Memory.writeBytesArray(ptrMessage,inputBytes);
25         var objMessage = ObjC.classesNSData.alloc().initWithBytes_length_(ptrMessage,inputBytes.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
32     // Output: output string ENCODED IN ASCII HEX
33     context_custom3: function(message) {
34         var inputBytes = hexToBytes(message);
35         var ptrMessage = Memory.alloc(inputBytes.length);
36         Memory.writeBytesArray(ptrMessage,inputBytes);
37         var objMessage = ObjC.classesNSString.stringWithString_hexToHexString(message);
38         return objMessage.toUpperCaseString();
39     }
```

### BRIDA 0.2 - FRIDA CONSOLE

<table>
<thead>
<tr>
<th>Target</th>
<th>Proxy</th>
<th>Spider</th>
<th>Scanner</th>
<th>Monitor</th>
<th>Sequence Decoder</th>
<th>Company</th>
<th>Use Option</th>
<th>Logger</th>
<th>Bible</th>
</tr>
</thead>
</table>

- **Server status:** running
- **Application status:** connected
- **Python binary path:** /usr/bin/python
- **Pyroot:** local

**FRIDA JS file path:** /usr/local/bin/FRIDAJS

**Application ID:** (app id)

**FRIDA REMOTE:** FRIDA Local

```python
*** entered = {Encryption event=remoteClient}:
  ** Caller:** (remoteClient)
  ** Parameters:**
  (NLString) encryptRequest: ** message
*** entering = {Encryption event=remoteClient}:
  ** Return value:**

```
- 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

Result of the search of decrypt:

- [Encryption decryptResponse]
- [FWEncryptorAES decrypt:Key:IV]
- [FWEncryptorAES decryptFromBase64:Key:IV]
- [FWEncryptorAES decryptWithGUID:Key:IV]
- [Hello_Fede_encryptViewController decryptButton]
- [Hello_Fede_encryptViewController decryptReceived]

Search: decrypt

- [NSData decryptFromAES128 enlightenment:Key:Error]
- [NSData decryptWithAES128 enlightenment:Key:Error]
- [NSData decryptWithAES128 enlightenment:Algorithm:Key:Error]
- [NSData decryptWithAES128 enlightenment:Algorithm:Key:Initialized]

- [import:/usr/lib/system/libcorecrypto.dylib:ocspd_xte_decrypt]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp2_cbc_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp2_cfb5_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp2_cfb_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp2_ecb_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp3_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp3_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:ocsp3_decrypt_mode]
- [import:/usr/lib/system/libcorecrypto.dylib:tls_record_decrypt]
- [import:/usr/lib/system/libcorecrypto.dylib:tls_record_decrypted_size]
- [import:/usr/lib/system.B.dylib:ocsp_cbc_decrypt_mode]
- [import:/usr/lib/system.B.dylib:ocsp_cbc_decrypt_mode]
- [import:/usr/lib/system.B.dylib:ocsp_cbc_decrypt_mode]
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

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

*** Tracing +[Encryption encryptRequest:]
*** Tracing +[Encryption decryptResponse:]
*** entered +[Encryption encryptRequest:]
Caller: 0x1000fe640 Hello Fede!_T010Hello_Fede!8EncryptionRequestsC20sendEncryptedMessageySS7message_T
Parameters:
(MutablePointerString) encryptRequest: phone
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.hib.BridaDomo 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
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!
• 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!
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

Request: «apple»
Response: «Red apple»

decrypt( Eκ(«apple»))

Search

Eκ(«apple»)

Eκ(«Red apple»)

Search

Eκ(«apple»)

Eκ(«Red apple»)

INSPECTION
USE CASE 1 - TAMPERING

Old request: «apple»
New request: «lemon»

Response
Ek(«apple»)
Search
Ek(«Lemon Sicily»)

Response
Ek(«lemon»)
Search
Ek(«Lemon Sicily»)

encrypt(Ek(«lemon»))
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

deviceld
Register
OK

deviceld
Register
OK

Register

=deviceId

Register
USE CASE 2 - MAKE REQUEST FIRST 10 REQUESTS...
USE CASE 2 - MAKE REQUEST
...REQUEST 11
USE CASE 2 - SCANNER

Request 1
Request ...
Request 10
Request 11
Request 12
Request 13

Quote 1
Quote ...
Quote 10

Challenge
Challenge
Challenge

NO SCANNER - NO INTRUDER
USE CASE 2 - SCANNER

getResponse(Challenge)

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

Request: admin/E_k("password")

Response

«User logged in»

Login

admin/E_k("password")

Response

«User logged in»

Login
USE CASE 3 - INTRUDER

NO SCANNER - NO INTRUDER
USE CASE 3 - INTRUDER

encryptPassword («qwerty»)

OK SCANNER - OK INTRUDER

Hello admin!
• 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/
ANY QUESTION?

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

THANKS!
CONGRATULATIONS MARIO!

AUTHORS
FEDERICO DOTTA - PIERGIOVANNI CIPOLLONI

REVIEW
MAURIZIO AGAZZINI - MARCOIVALDI

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.