DBI-Assisted Android Application Reverse Engineering

26 July 2019

Sahil Dhar

xen1thLabs

A DARKMATTER COMPANY

SMART AND SAFE DIGITAL

Whoami

Sahil Dhar

Security researcher

My area of expertise include Web and Mobile application security. Prior to joining Xen1thLabs, I have worked on numerous projects involving the security assessment of Web, Mobile, Thick clients and Network and Cloud Infrastructure for multiple fortune 500 companies.

@0x401

A former

- Security Engineer
- Security Consultant
- Bug Bounty Hunter

Currently

- Security researcher
- @Xen1th Labs

Content

- What is DBI ?
- What is Frida ?
- Frida JavaScript APIs
    - Console
    - RPC
    - Java
    - Interceptor
- Example Use cases
- Frameworks built using Frida

01

Dynamic Binary Instrumentation

Dynamic Binary Instrumentation ?

```
1. #include<stdio.h>
2. #include<string.h>
3.
4. int main(int argc, char *argv[]){
5.   if (strcmp(argv[1], "admin\x0a")){
6.     printf("Access Granted");
7.   }else{
8.     printf("Access Denied");
9.   }
10.    }
```

Dynamic Binary Instrumentation ?

```
1. #include<stdio.h>
2. #include<string.h>
3.
4. int main(int argc, char *argv[]){
5.   printf("Inside main function");
6.   if (strcmp(argv[1], "admin\x0a")){
7.     printf("comparison successful");
8.     printf("Access Granted");
9.   }else{
10.         printf("comparison failed");
11.         printf("Access Denied");
12.       }
13.     }
```

Compile-time Instrumentation (AFL)

```
var_A0= qword ptr -0A0h
var_98= qword ptr -98h
var_90= qword ptr -90h
var_88= qword ptr -88h

argc = rdi              ; int
argv = rsi              ; char **
; __unwind {
lea     rsp, [rsp-98h]
mov     [rsp+98h+var_98], rdx
mov     [rsp+98h+var_90], rcx
mov     [rsp+98h+var_88], rax
mov     rcx, 0B692h
call    __afl_maybe_log
mov     rax, [rsp+98h+var_88]
mov     rcx, [rsp+98h+var_90]
mov     rdx, [rsp+98h+var_98]
lea     rsp, [rsp+98h]
sub     rsp, 8
mov     argv, [argv+8]
lea     argc, aAdmin    ; "admin\n"
mov     ecx, 7
repe cmpsb
setnbe  al
sbb     al, 0
test    al, al
jnz     loc_8C7
```

```
nop
lea     rsp, [rsp-98h]
mov     [rsp+0A0h+var_A0], rdx
mov     [rsp+0A0h+var_98], rcx
mov     [rsp+0A0h+var_90], rax
mov     rcx, 0CABFh
call    __afl_maybe_log
mov     rax, [rsp+0A0h+var_90]
mov     rcx, [rsp+0A0h+var_98]
mov     rdx, [rsp+0A0h+var_A0]
lea     rsp, [rsp+98h]
lea     rsi, aAccessDenied ; "Access Denied"
mov     edi, 1
xor     eax, eax
call    ___printf_chk
```

```
loc_8C7:
nop
lea     rsp, [rsp-98h]
mov     [rsp+0A0h+var_A0], rdx
mov     [rsp+0A0h+var_98], rcx
mov     [rsp+0A0h+var_90], rax
mov     rcx, 0E7B9h
call    __afl_maybe_log
mov     rax, [rsp+0A0h+var_90]
mov     rcx, [rsp+0A0h+var_98]
mov     rdx, [rsp+0A0h+var_A0]
lea     rsp, [rsp+98h]
lea     rsi, aAccessGranted ; "Access Granted"
mov     edi, 1
xor     eax, eax
call    ___printf_chk
jmp     loc_887
; } // starts at 7E0
main endp
```

```
loc_887:
nop
lea     rsp, [rsp-98h]
mov     [rsp+0A0h+var_A0], rdx
mov     [rsp+0A0h+var_98], rcx
mov     [rsp+0A0h+var_90], rax
mov     rcx, 94Dh
call    __afl_maybe_log
mov     rax, [rsp+0A0h+var_90]
mov     rcx, [rsp+0A0h+var_98]
mov     rdx, [rsp+0A0h+var_A0]
lea     rsp, [rsp+98h]
xor     eax, eax
add     rsp, 8
retn
```

02

Frida

Getting Started

- Installation Commands
  - `apt-get install python3`
  - `pip3 install Frida`
  - Download platform specific (x86, x64, ARM) Frida Server & Agents
    - https://github.com/frida/frida/releases
  - Transfer frida-server binary to `/data/local/tmp` using `adb`
    - `adb push frida-server /data/local/tmp`
    - `adb shell; su`
    - `cd /data/local/tmp; chmod +x frida-server`
    - `./frida-server &`

Frida Command line Tools

- **Common Options:**

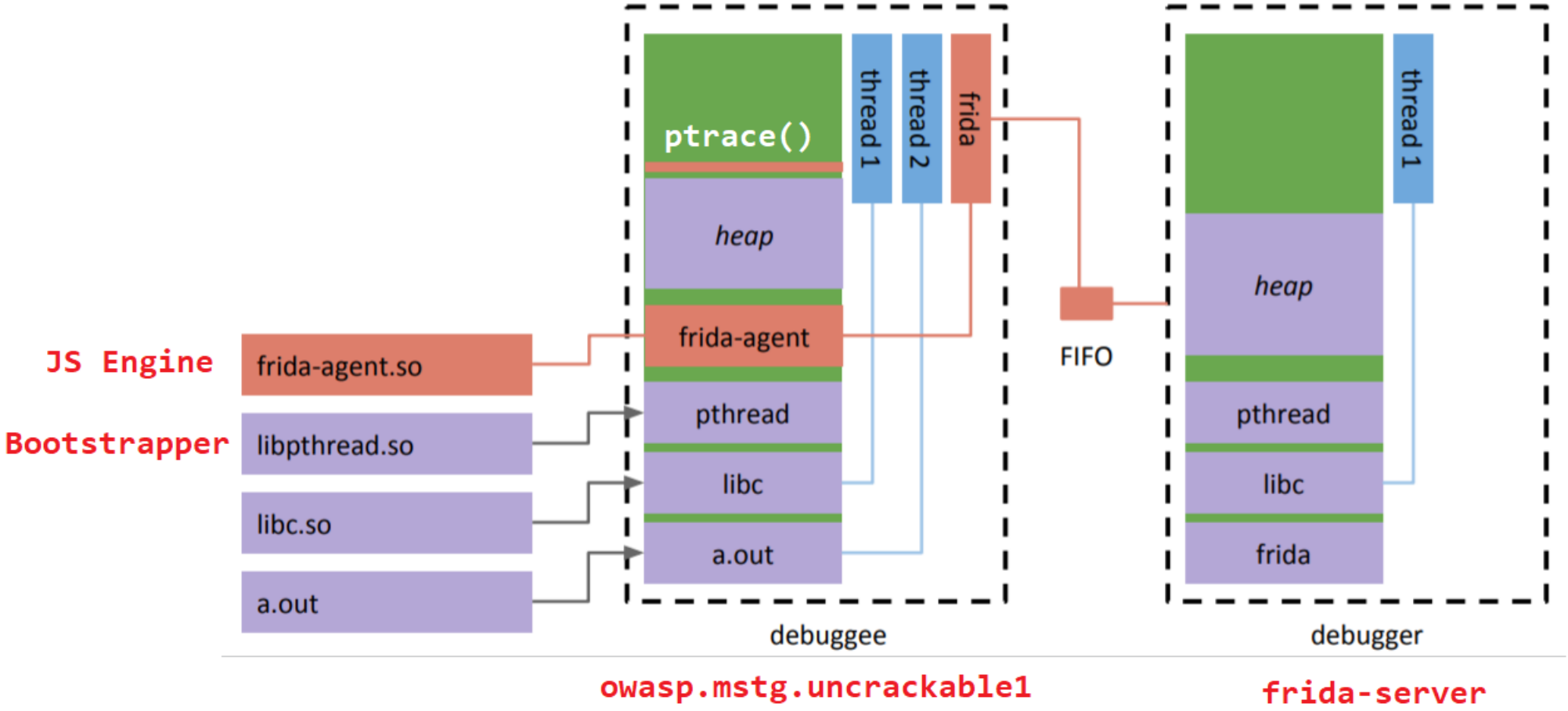
```
-D ID, --device=ID    connect to device with the given ID
-U, --usb             connect to USB device
-R, --remote          connect to remote frida-server
-H HOST, --host=HOST  connect to remote frida-server on HOST
-f FILE, --file=FILE  spawn FILE
-n NAME, --attach-name=NAME
```

- **frida-ps:** List all running and installed applications on the mobile device.
- **frida-ls-devices:** Shows the list of attached devices.
- **frida-kill:** Kill the process running on the device by providing process name or process id.
- **frida-discover:** Tool for discovering internal functions in the program.
- **frida-trace:** Useful for tracing native function calls. This won't work for high-level function calls such as **`loadLibrary()`** from **`java.lang.System`** class in java.

Frida Internals

- Hijack remote thread - **`ptrace()`**
- Allocating memory for Bootstrapper: **`frida_resolve_library_function - mmap()`**
- Load libpthread.so - **`dlopen()`**
- Create new thread - **`thread_create()`** - executing libpthread.so
- Notify debugger
  - Load frida-agent.so **`dlopen()`** in process memory
  - Locate entry point of frida-agent
- Resume hijacked thread execution
  - Execute entry point of frida-agent
- Resume execution

Frida Internals



Source: https://www.frida.re/docs/presentations/osdc-2015-the-engineering-behind-the-reverse-engineering.pdf

Frida Internals

```
SD-DM-C02TR049HTD6 ॐ ~/Desktop/fridatalk:
[5543 ◯ frida-ps -U -a -i|grep -i owasp
1939  Uncrackable1                          owasp.mstg.uncrackable1
SD-DM-C02TR049HTD6 ॐ ~/Desktop/fridatalk:
[5544 ◯ adb shell cat /proc/1939/maps|grep -i frida
d0226000-d03a9000 r-xp 00000000 08:13 627094                    /data/local/tmp/re.frida.server/frida-agent-32.so
d03a9000-d03aa000 rwxp 00183000 08:13 627094                    /data/local/tmp/re.frida.server/frida-agent-32.so
d03aa000-d16d0000 r-xp 00184000 08:13 627094                    /data/local/tmp/re.frida.server/frida-agent-32.so
d16d1000-d1718000 r--p 014aa000 08:13 627094                    /data/local/tmp/re.frida.server/frida-agent-32.so
d1718000-d1760000 rw-p 014f1000 08:13 627094                    /data/local/tmp/re.frida.server/frida-agent-32.so
```

Operation Modes

- Injected
- Embedded
  - Interaction types
    - listen
    - script
    - script-directory
- Preloaded **(LD_PRELOAD)**

Embedded mode automation: https://www.github.com/sahildhar/mlibinjector

03

Frida JavaScript APIs

Frida Console – Hello World

DEMO

RPC

```
1. import frida
2.
3. # Frida-python Api documentation can be found at ;)
4. # https://github.com/frida/frida-python/blob/master/frida/__init__.py
5.
6. rpc_hello = """
7. rpc.exports = {
8.     helloWorld: function (){
9.         return "Hello World";
10.         },
11.     };
12.     """
13.
14.     device = frida.get_usb_device(timeout=1);
15.     pid = device.spawn("owasp.mstg.uncrackable1")
16.     session = device.attach(pid)
17.     script = session.create_script(rpc_hello)
18.     script.load()
19.     api = script.exports
20.     print(api.hello_world());
```

Java

- `Java.Perform(fn):` **`fn`** will execute after the application class loader is initialized.
- `Java.PerformNow(fn):` Very useful for early instrumentation i.e. hooking the implementation of system classes and functions.
- `Java.use(className):` Get wrapper for className which can be initialized by calling **`$new()`** on it.
- `Java.enumerateLoadedClasses(callback):` Enumerate all the classes loaded and execute callback functions on each of them.
- `Java.enumerateLoadedClassesSync():` Similar to **`enumerateLoadedClasses`** but returns an array of all of the loaded classes
- `Java.scheduleOnMainThread(fn):` Run **`fn`** on main thread, useful for tinkering around the objects that must execute in the main thread. Hint: UI
- `Java.choose (classname, callbacks):` Search for initialized instances of **`classname`** and execute the functions defined in the **`callbacks`**.
- `Java.array(dataType, []):` useful when dealing with java arrays.

04

Frida + Java Reflection API

Enumerating Class Methods

```
1. Java.perform(function(){
2.   Java.enumerateLoadedClasses({
3.     onMatch: function(className){
4.       try{
5.         var _class = Java.use(className);
6.         if(className.includes("rootinspector")){
7.           var methods = _class.class.getDeclaredMethods();
8.           methods.map(function(methodName){
9.             if(methodName.toString().includes("native")){
10.                  console.log(methodName);
11.                }
12.              });
13.            }
14.          }
15.          catch(err){}
16.        },
17.        onComplete:function(){}
18.      });
19.    });
```

Defining Method Implementation

```
1. Java.perform(function(){
2.     var _rootClass = Java.use("com.devadvance.rootinspector.Root");
3.     var _checkRootMethod0 = _rootClass.checkRootMethod0
4.     _checkRootMethod0.implementation = function()
5.     {
6.         return false;
7.
8.     }
9. });
```

Method Overloading

```
1. Java.performNow(function(){
2.
3.   var _alertDialog = Java.use("android.app.AlertDialog");
4.   _alertDialog.setMessage.overload("java.lang.CharSequence").implementation = function(message){
5.     console.error(message.toString());
6.     return _alertDialog.setMessage.overload("java.lang.CharSequence").apply(this,arguments);
7.   }
8. });
```

setMessage(int messageId)

Set the message to display using the given resource id.

setMessage(CharSequence message)

Set the message to display.

Source: https://developer.android.com/reference/android/app/AlertDialog.Builder

Hooking Class Constructors

```
1.Java.perform(function () {
2.    var mainactivity = Java.use("com.pragyan.circle.a");
3.    mainactivity.$init.overloads[0].implementation = function(a,b,c,d,e) {
4.        console.log(d);
5.        return mainactivity.$init.overloads[0].apply(this,arguments);
6.    };
7.
8.    send("Hooks installed.");
9.});
```

Calling Pre-initialized Class Instances

```
1. Java.perform(function () {
2.     var _javaString = Java.use("java.lang.String");
3.     var _hexValue = _javaString.$new("65544231587a52794d3138316458417a636c396d4e445
   53343673d3d");
4.     Java.choose("com.pragyan.circle.Main",{
5.         onMatch: function(instance){
6.             var flag = instance.a(_hexValue);
7.             console.log(flag);
8.         },
9.         onComplete: function(){}
10.         });
11.     });
```

Making sense out of [Object ----]

- Java Reflection API(s)

```
1. Java.cast(obj.getClass(), Java.use("java.lang.Class")).getDeclaredFields();
```

- Iterating over returned [Object Object] array.

```
1. function getObjectDetails(obj){
2.   for (var k in obj){
3.     console.log(obj[k]);
4.   }
5.   console.log("done");
6. }
```

Interceptor

```
Java.perform(function(){
  var checkfopen = Module.findExportByName("libnative2.so", "Java_com_devadvance_rootinspector_Root_checkfopen");
  if(checkfopen){
    console.log("checkfopen is at: "+checkfopen)
    Interceptor.attach(checkfopen,{
      onEnter: function(args){
        //casting jstring object to String
        console.log(Java.cast(ptr(args[2]), Java.use("java.lang.String")));
        console.log("checkfopen called");
          },
          onLeave: function(retval){
            // returning false for the check
            retval.replace(ptr(0));
          }
        });
      }
    });
```

05

Some Use Cases

Root Detection Bypass

```
1. Java.performNow(function(){
2.
3.   var _system = Java.use("java.lang.System");
4.   _system.exit.implementation = function(){
5.     console.log("Exit called");
6.   }
7.
8. });
```

SQL Cipher – Find and Decrypt

```
1. Java.choose("net.sqlcipher.database.SQLiteDatabase",{
2.   onMatch: function(ins){
3.     var path_parts = ins.getPath().split("/");
4.     var db_name = path_parts[(path_parts.length -1)].split(".")[0]
5.     console.log("DB Name: "+ db_name)
6.     var rnd_db_name = db_name + "_" +random_name(5);
7.     var sql1 = String.$new("ATTACH DATABASE '/data/user/0/com.test.production/databases/"+rnd
   _db_name+".sql.plaintext' as "+rnd_db_name+" KEY '';");
8.     var sql2 = String.$new("SELECT sqlcipher_export('"+rnd_db_name+"');");
9.     var sql3 = String.$new("DETACH DATABASE "+rnd_db_name);
10.         ins.rawExecSQL(sql1);
11.         ins.rawExecSQL(sql2);
12.         ins.rawExecSQL(sql3);
13.         console.log("Found SqlCipherDatabaseProvider instance");
14.       },
15.       onComplete: function(ins){}
16.     });
```

Tracing arbitrary Java methods

DEMO

06

Frameworks Built Over Frida

Awesome Frida Frameworks

- Appmon (https://github.com/dpnishant/appmon)
- House (https://github.com/nccgroup/house)
- Objection (https://github.com/sensepost/objection)

References

- https://www.frida.re
- https://github.com/frida/frida-core/blob/dd69ad9be80eda6e172d16d924c3db3080f6e40c/src/linux/frida-helper-backend-glue.c#L393
- https://github.com/dweinstein/awesome-frida
- https://github.com/sahildhar/mlibinjector
- https://docs.oracle.com/javase/8/docs/technotes/guides/reflection/index.html
- https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/
- https://developer.android.com/reference/android/app/AlertDialog.Builder

Questions ?

Thank you ☺