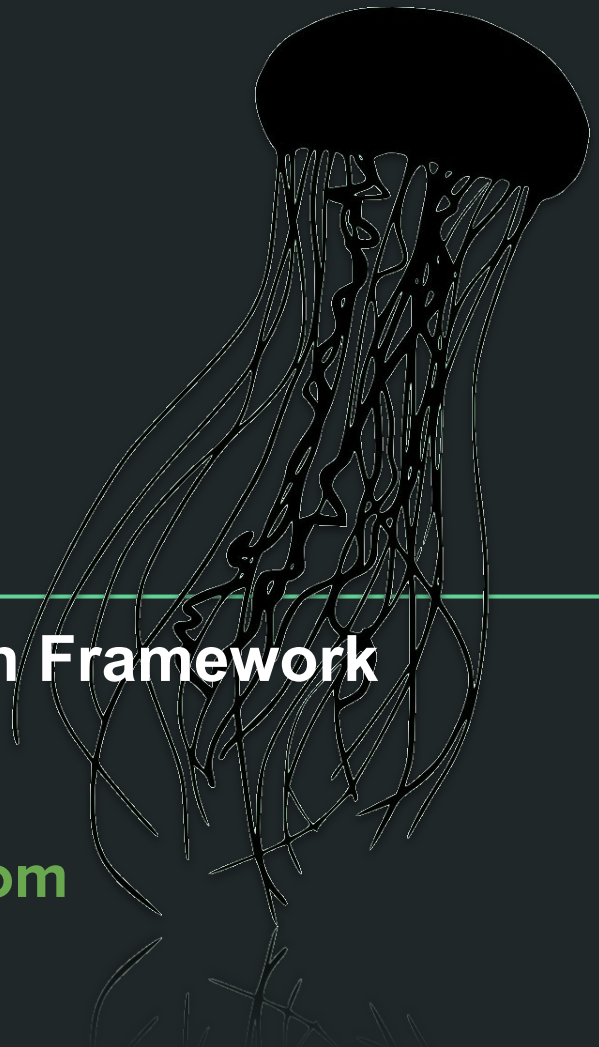
MEDUSA

Powered by FRIDA



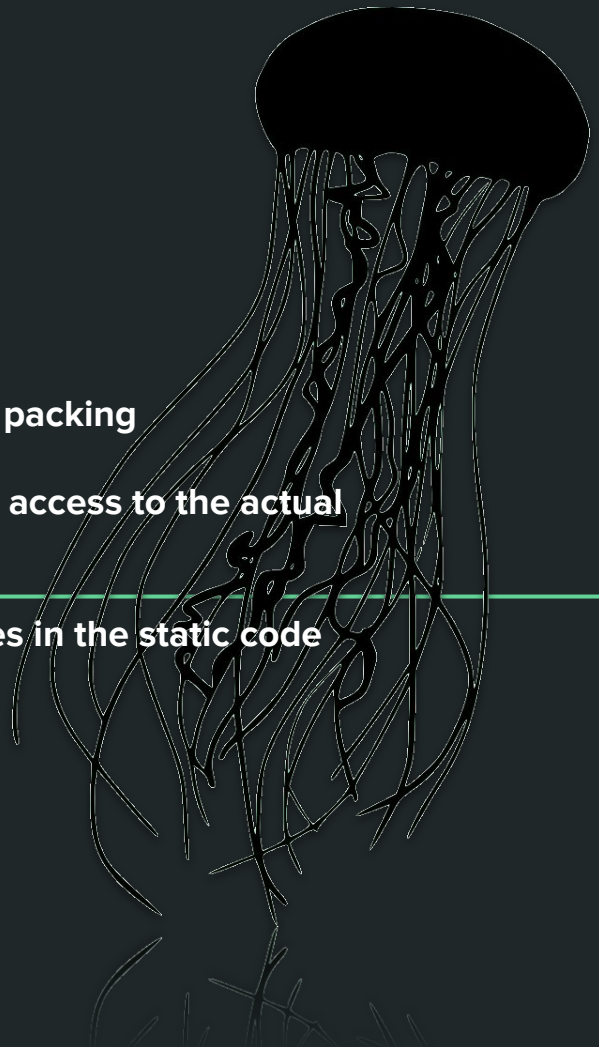
Modularized Binary Instrumentation Framework focusing on Malware Investigation

dimitrios.valsamaras [at] cognizant.com
Malware Analyst / Reverse Engineer

Contents

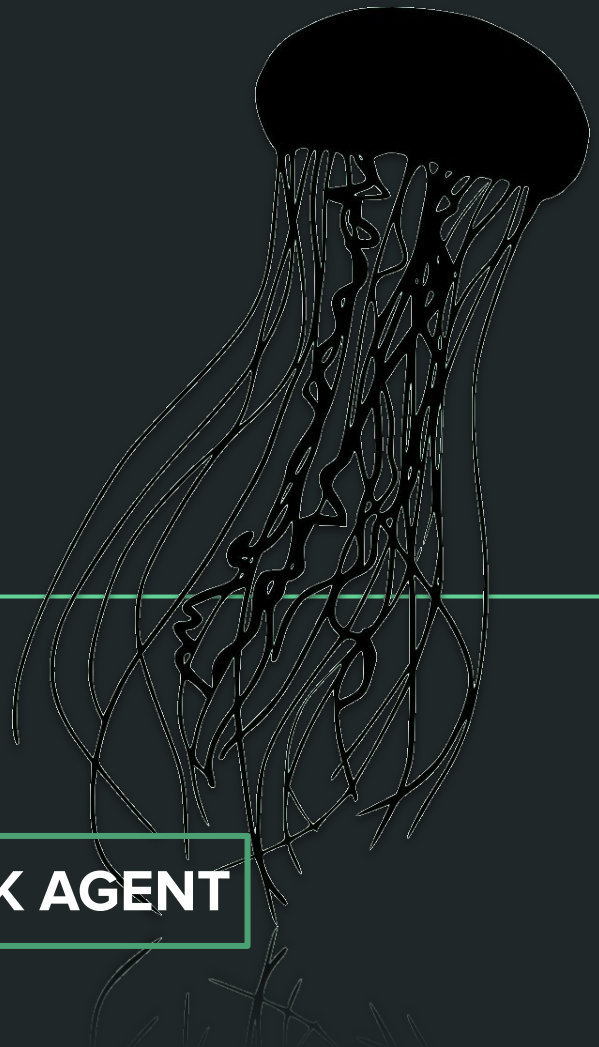
Dynamic Analysis

- Fast !!
- Reliable and Indisputable
- Highly effective in cases of heavy obfuscation / encryption / packing
- Allows the analysis of applications in which you do not have access to the actual code.
- Identifies vulnerabilities that might have been false negatives in the static code analysis.
- Permits the validation of static code analysis findings.
- Can be conducted against any application.



Medusa Milestones

- ❑ Modularity
- ❑ Categorization
- ❑ Extensibility
- ❑ Flexibility

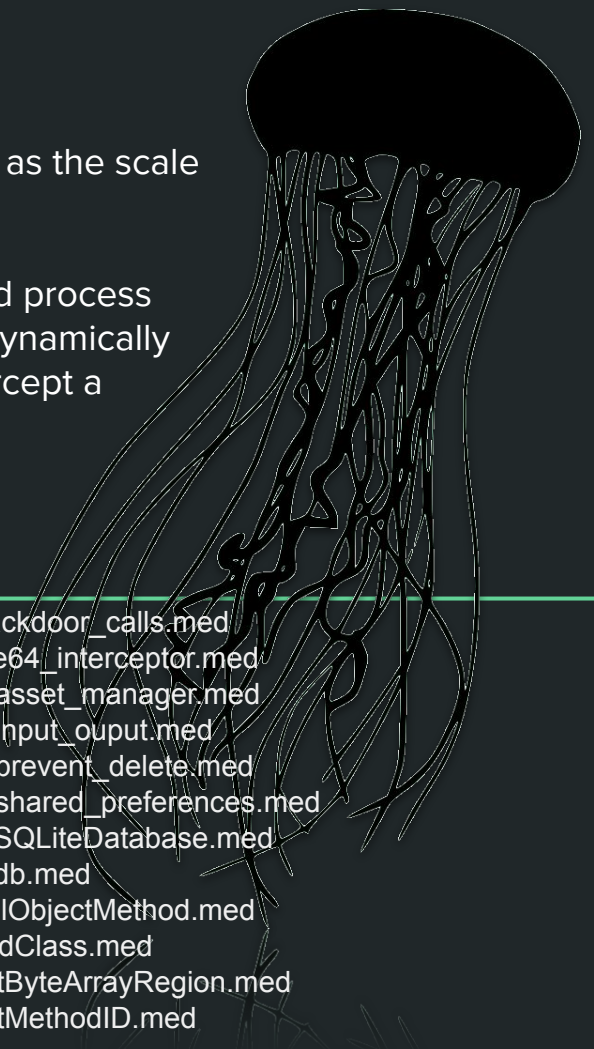


APK UTILITIES | MEDUSA MODULE | APK AGENT

Modularity

Modularity is one of the main characteristics of the specific framework as the scale of interference may be dynamically adjusted using modules.

In its barebone instance a FRIDA agent will be attached to the targeted process without interfering any kind events. The scale of interference can be dynamically adjusted by adding or removing modules that direct the server to intercept a specific set of API calls.



```
medusa> show all
modules/scratchpad.med
modules/encryption/cipher.med
modules/encryption/hash_operations.med
modules/cordova/cordova_enable_debugging.med
modules/http_comnunications/intercept_json_objects.med
modules/http_comnunications/multiple_unpinner.med
modules/http_comnunications/okhttp3_retrofit.med
modules/http_comnunications/universal_SSL_pinning_bypass.med
modules/http_comnunications/volley_request.med
modules/sockets/socket_monitor.med
modules/sockets/socket_monitor_2.med
modules/clickers/click_toll_fraud.med
modules/backdoor/backdoor_calls.med
modules/base64/base64_interceptor.med
modules/file_system/asset_manager.med
modules/file_system/input_ouput.med
modules/file_system/prevent_delete.med
modules/file_system/shared_preferences.med
modules/db_queries/SQLiteDatabase.med
modules/db_queries/db.med
modules/JNICalls/CallObjectMethod.med
modules/JNICalls/FindClass.med
modules/JNICalls/GetByteArrayRegion.med
modules/JNICalls/GetMethodID.med
```

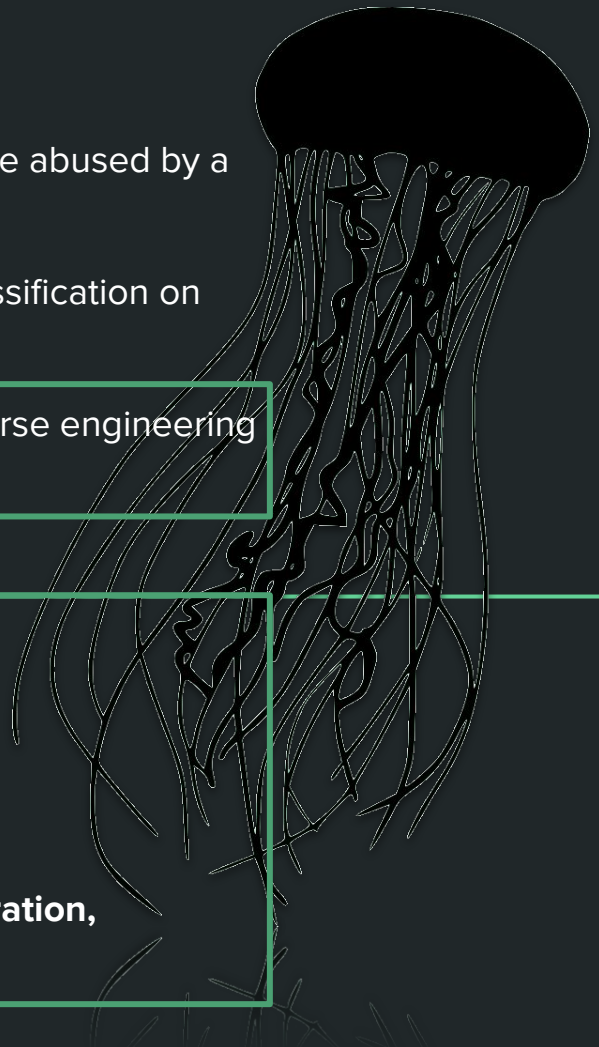
Categorization

- Each module intercepts a subset of API Calls which is likely to be abused by a specific malware category.
- The modules have been categorized according to Google's classification on Potential Harmful Applications [1]
- Additional helper-modules and utilities, meant to automate reverse engineering tasks, have also been added.

- Adb wrappers focusing on reverse engineering operations
- Memory dumping to deal with packing has been ported

What else ?

Native hooks, Batch Hooks, Static patching, Process Memory Exploration, Remote debugging with JDWP, Manifest parsers .....



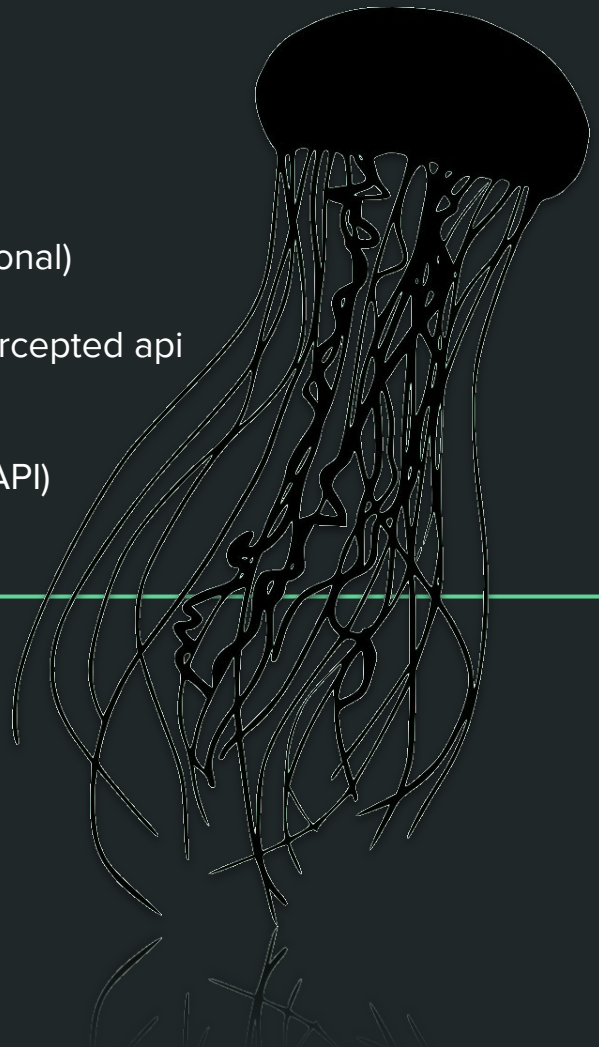
Extensibility

A module consists of three parts:

- **Description**: A short overview of what the module is doing (optional)
- **Help:** An extended description of the module, including the intercepted api calls (optional)
- **Code:** Hooks written in JavaScript language (FRIDA JavaScript API)

modules > helpers > JS cancel_system_exit.med > ...

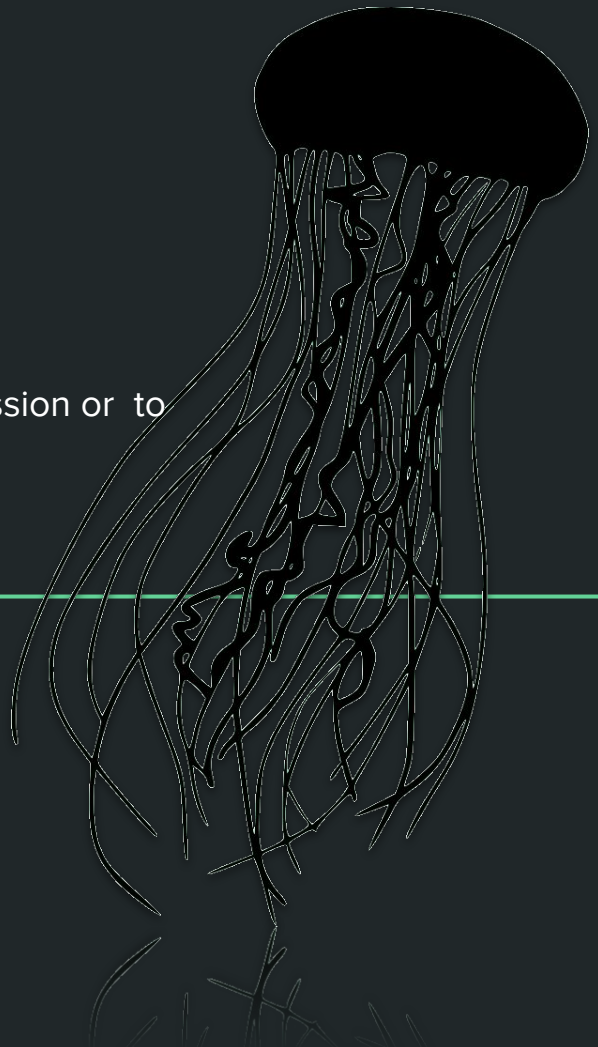
```
#Description: 'Cancels application exit'
#Help: "Hooks system.exit, activity.finish to cancel application's exit"
#Code:


console.log("-----------Hooking SYSTEM EXIT----------------------");

  var sysexit = Java.use("java.lang.System");
  var activity = Java.use('android.app.Activity');

  sysexit.exit.overload("int").implementation = function(var_0) {
    colorLog("[i] Canceling system exit", {c: Color.Green});
  };

  activity.finish.overloads[0].implementation = function(){
    colorLog("[+] Canceling activity's finish" ,{c: Color.Green});

  }
```



Flexibility

Using simple operations the user is able to:

- **Remove** or **Modify Modules**
- **Add, Remove, Modify, Re-Classify Module - Sets**
- **Export Modules** or **Module-Sets** that may be used to save a session or to collaborate within a team



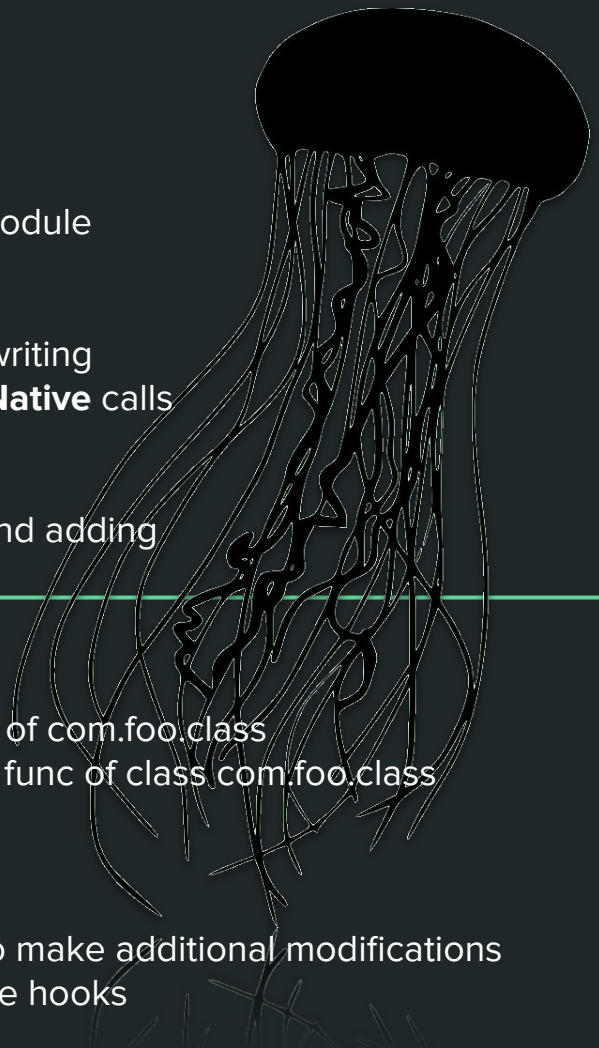
Flexibility (Continued)

Introducing the ScratchPad

- The **ScratchPad** (modules/scratchpad.med) is a special type of module where the user may include application specific hooks.

- A subset of commands has been designed to simplify the script writing process, giving the user the option to intercept multiple **Java** or **Native** calls with a simple command.

- Further modification is also available by editing the ScratchPad and adding the final touches.

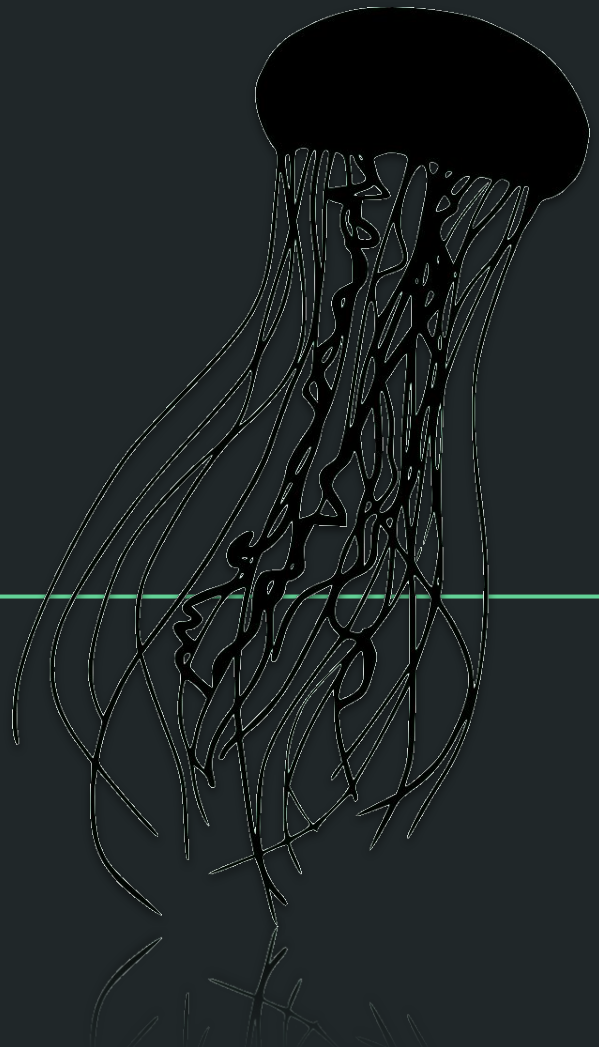


medusa > hook

- ❖ **-a** com.foo.class // Intercept all functions of com.foo.class
- ❖ **-f** com.foo.class.func // Intercept the function func of class com.foo.class
- ❖ **-n** libfoo.so Java_com_foo_class_f // Intercept a native call

medusa > import [snippet name] // import a frida script
medusa > pad // Edit the ScratchPad to make additional modifications
medusa > compile // Last step to enable the hooks

Modules
(Main Categories)



Modules (Spyware)

- Context queries
- Location tracking
- System properties exfiltration
- Media projection
- Camera Usage
- Clipboard Usage
- Keylogging

```
var telephonyManager = Java.use('android.telephony.TelephonyManager');
var mediaRecorder = Java.use('android.media.MediaRecorder')
var audioRecord = Java.use('android.media.AudioRecord');
var abstractCursor = Java.use('android.database.AbstractCursor');
var cursor = Java.use('android.database.Cursor');
var clipboardManager = Java.use('android.content.ClipboardManager');
var Location = Java.use('android.location.Location');
var systemProperties = Java.use('android.os.SystemProperties');
var contextWrapper = Java.use("android.content.ContextWrapper");
var contentResolver = Java.use('android.content.ContentResolver');
var locationListener = Java.use("android.location.LocationListener");
var locationManager = Java.use('android.location.LocationManager');
var systemProperties = Java.use('android.os.SystemProperties');
var screenCapture = Java.use('android.media.projection.MediaProjectionManager');
```

```
Location.getLatitude.implementation = function() {
        var lt = this.getLatitude();
        console.log('[i] Application is getting latitude:'
+ lt);
        return lt;
}
Location.getLongitude.implementation = function() {
        var lgt = this.getLongitude();
        console.log('[i] Application is getting
longitude:'+lgt);
        return lgt;
}
```

```
mediaRecorder.stop.implementation = function(){
        console.log('[i] Media recording stopped');
        this.stop();
}

mediaRecorder.start.implementation = function(){
        console.log('[i] Media recording started');
        this.start();
}

audioRecord.startRecording.overloads[0].implementation =
function(){
        console.log('[i] Application is recording audio');
        this.startRecording();
}
```

Modules (Click - Toll fraud)

- Motion Events
- Automated clicks
- JavaScript Injection
- Telephony Manager Interaction
- Wifi Manager Interaction
- View Transparency
- Dynamic Code Loading

```
var viewClassHook = Java.use('android.view.View');
var motionEvent = Java.use('android.view.MotionEvent');
var webView = Java.use('android.webkit.WebView');
var appInstrumentation = Java.use('android.app.Instrumentation');
var wifiManager = Java.use("android.net.wifi.WifiManager");
var telephonyManager = Java.use('android.telephony.TelephonyManager');
```

```
telephonyManager.getNetworkOperator.overload('int').impl
ementation = function(a){
   console.log('[!] A call to
android.telephony.TelephonyManager.getNetworkOperator
detected');
   return this.getNetworkOperator();
}
telephonyManager.getNetworkOperatorName.overload().imple
mentation = function(){
   console.log('[!] A call to
android.telephony.TelephonyManager.getNetworkOperatorNam
e detected');
   return this.getNetworkOperatorName();

}
```

```
wifiManager.setWifiEnabled.implementation = function(enabled){
   if(enabled == false)
      colorLog('[!] Application is disabling the WiFi',{ c:
Color.Red });
   else
      colorLog('[!] A call to
android.net.wifi.WifiManager.setWifiEnabled detected',{c:
Color.Red});
   return this.setWifiEnabled(enabled);
}
viewClassHook.performClick.implementation = function(){
   colorLog('[+] Perform Click detected on:'+this.$className,
{c:Color.Red});
   return this.performClick();
}
```

Modules (Backdoor & DCL)

- Command Execution
- Dynamic Code Loading
- Native Library Loading
- Sockets

```
var dexclassLoader = Java.use("dalvik.system.DexClassLoader");
var basedexclassLoader = Java.use("dalvik.system.BaseDexClassLoader");
var clazz = Java.use('java.lang.Class');
var targetClass = Java.use("java.lang.Runtime");
var systemA = Java.use('java.lang.System');
var socket = Java.use('java.net.Socket');
var WebSocketClient = Java.use('org.java_websocket.client.WebSocketClient');
```

```
dexclassLoader.$init.implementation = function(dexPath,  optimizedDirectory,  librarySearchPath,  parent)
{       colorLog('DexClassLoader called:', {c: Color.Green});
        console.log("dexPath=" + dexPath );
        console.log("optimizedDirectory=" + optimizedDirectory);
        console.log("librarySearchPath=" + librarySearchPath);
        console.log("parent=" + parent);
        return this.$init(dexPath, optimizedDirectory,  librarySearchPath,  parent);
}
targetClass.exec.overload('java.lang.String').implementation = function (x) {
        console.log("[*] exec() called!: "+x);
        return this.exec(x);
    };
socket.$init.overloads[2].implementation = function(socketImpl){
    console.log('[+] Creating socket for host[2]: ' + socketImpl.address.getHostName() +':'+port);
    return this.$init(host,port);
}
```

Modules (HTTP Communications)

- SSL Pinning bypass
- Common HTTP Libraries (volley, okhttp)
- JSON objects

```
var JSONLogger = Java.use('org.json.JSONObject');
var X509TrustManager = Java.use('javax.net.ssl.X509TrustManager');
var SSLContext = Java.use('javax.net.ssl.SSLContext');
var okhttp3Activity = Java.use('okhttp3.CertificatePinner');
var opnSSLsi = Java.use('com.android.org.conscrypt.OpenSSLSocketImpl');
var Interceptor = Java.use("okhttp3.Interceptor");
var OkHttpClient = Java.use("okhttp3.OkHttpClient");
var okhttp3HeadersBuilder  = Java.use('okhttp3.Headers$Builder');
var request = Java.use('com.android.volley.Request');
```

```
var MyInterceptorObj = MyInterceptor.$new();
var Builder = Java.use("okhttp3.OkHttpClient$Builder");
Builder.build.implementation = function() {
    this.interceptors().clear();
    this.interceptors().add(MyInterceptorObj);
    var result = this.build();
    return result;
      };
okhttp3HeadersBuilder.checkNameAndValue.implementation =
function(key,value){
          console.log(key+' : '+value);
          return this.checkNameAndValue(key,value);
}
```

```
// TrustManager (Android < 7)
var TrustManager = Java.registerClass({
    // Implement a custom TrustManager
    name: 'dev.asd.test.TrustManager',
    implements: [X509TrustManager],
    methods: {
        checkClientTrusted: function (chain, authType) {},
        checkServerTrusted: function (chain, authType) {},
        getAcceptedIssuers: function () {return []; }
    }
});
```

Modules (File-System / Encryption / Services / db Queries)

- Java Cryptographic Framework
- Hash operations
- Encoding / Decoding
- Shared Preferences
- SQL db
- Assets / Storage / File IO
- Compression
- Accessibility
- Intents

```
var cipher = Java.use('javax.crypto.Cipher');
var MessageDigest = Java.use("java.security.MessageDigest");
var base64 = Java.use('android.util.Base64');
var SharedPreferencesImpl = Java.use("android.app.SharedPreferencesImpl");
var fileOutputStream = Java.use('java.io.FileOutputStream');
var fileInputStream= Java.use('java.io.FileInputStream');
var assetManager = Java.use('android.content.res.AssetManager');
var gzipInputStream = Java.use('java.util.zip.GZIPInputStream');
var gzipOutputStream = Java.use('java.util.zip.GZIPOutputStream');
var intent = Java.use('android.content.Intent');
var accessibilityEvent =
Java.use('android.view.accessibility.AccessibilityEvent');
var accessibilityRecord=
Java.use('android.view.accessibility.AccessibilityRecord');
```

```
cipher.init.overload('int', 'java.security.Key', 'java.security.spec.AlgorithmParameterSpec').implementation =
function(mode,key,paramsec){
   var operation = '';
   var algorithm = this.getAlgorithm();
   var castedToIv = Java.cast(paramsec, Java.use('javax.crypto.spec.IvParameterSpec'));
   if(mode == 1)
       operation = "Encrypting";
   else if(mode == 2)
       operation = "Decrypting";
   colorLog('[+] Algorithm: '+ algorithm+ ' Operation: '+operation, {c:Color.Blue});
   colorLog('\t[-] Key (hex): '+ byteArraytoHexString(key.getEncoded()), {c:Color.Gray});
   if (algorithm.startsWith('AES') || algorithm.startsWith("RC4") || algorithm.startsWith("DES")){
       colorLog('\t\t[-] Key (Ascii): '+ byteArrayToStringE(key.getEncoded()), {c:Color.Red});
   }
   colorLog('\t[-] IV (hex): '+ byteArraytoHexString(castedToIv.getIV()), {c:Color.Gray});
   return this.init(mode,sks,paramsec);
```

Modules (Helpers)

- UI Translator
- Reflection
- Remove restrictions (e.g. screencap)
- Anti Debug
- Cancel Exit/Finish
- Root Detection
- Keystore
- Bluetooth

```
var antidebug = Java.use('android.os.Debug');
var sysexit = Java.use("java.lang.System");
var activity = Java.use('android.app.Activity');
var classDef = Java.use('java.lang.Class');
var classLoaderDef = Java.use('java.lang.ClassLoader');
var window = Java.use('android.view.Window');
var textViewClass = Java.use("android.widget.TextView");
var alertDialog = Java.use("android.app.AlertDialog");
var keyStoreLoadStream = Java.use('java.security.KeyStore')
var BluetoothGatt = Java.use("android.bluetooth.BluetoothGatt");
```

```
sysexit.exit.overload("int").implementation =
function(var_0)
{
   colorLog("[i] Canceling system exit", {c:
Color.Green});
}
button.setEnabled.implementation = function(a){
   console.log('Canceling button dissable');
   this.setEnabled(true);
}
classDef.forName.overloads[0].implementation =
function(name){
       var name = this.forName(name);
       console.log("[i] Reflection class: " +
name,{c:Color.Yellow} );
       return name;}
```

```
textViewClass.setText.overload('java.lang.CharSequence').implementation =
function (originalTxt) {
       var string_to_send = originalTxt.toString();
       var string_to_recv = "";
       send("trscrpt|"+string_to_send);
       recv(function (received_json_object) {
           string_to_recv = received_json_object.my_data;
       }).wait();
       colorLog('Translating: ' + string_to_send +" ---> "+ string_to_recv,
{c: Color.Green})
        var castTostring = String.$new(string_to_recv);

       return this.setText(castTostring);
   }
```

WorkFlows
(apkutils - medusa - agent)

WorkFlow (apkutils)

INSTALL

./apkutils.py foo.apk [--patch]

Enable debug flag

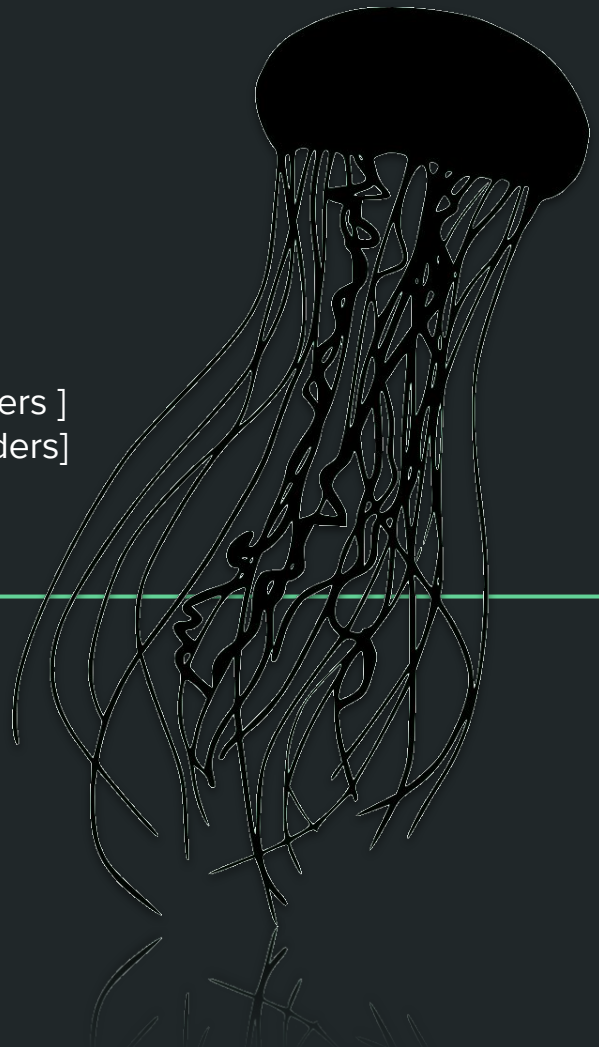
Recon

apkutils> show → [activities, services, receivers]
[permissions, filters, providers]

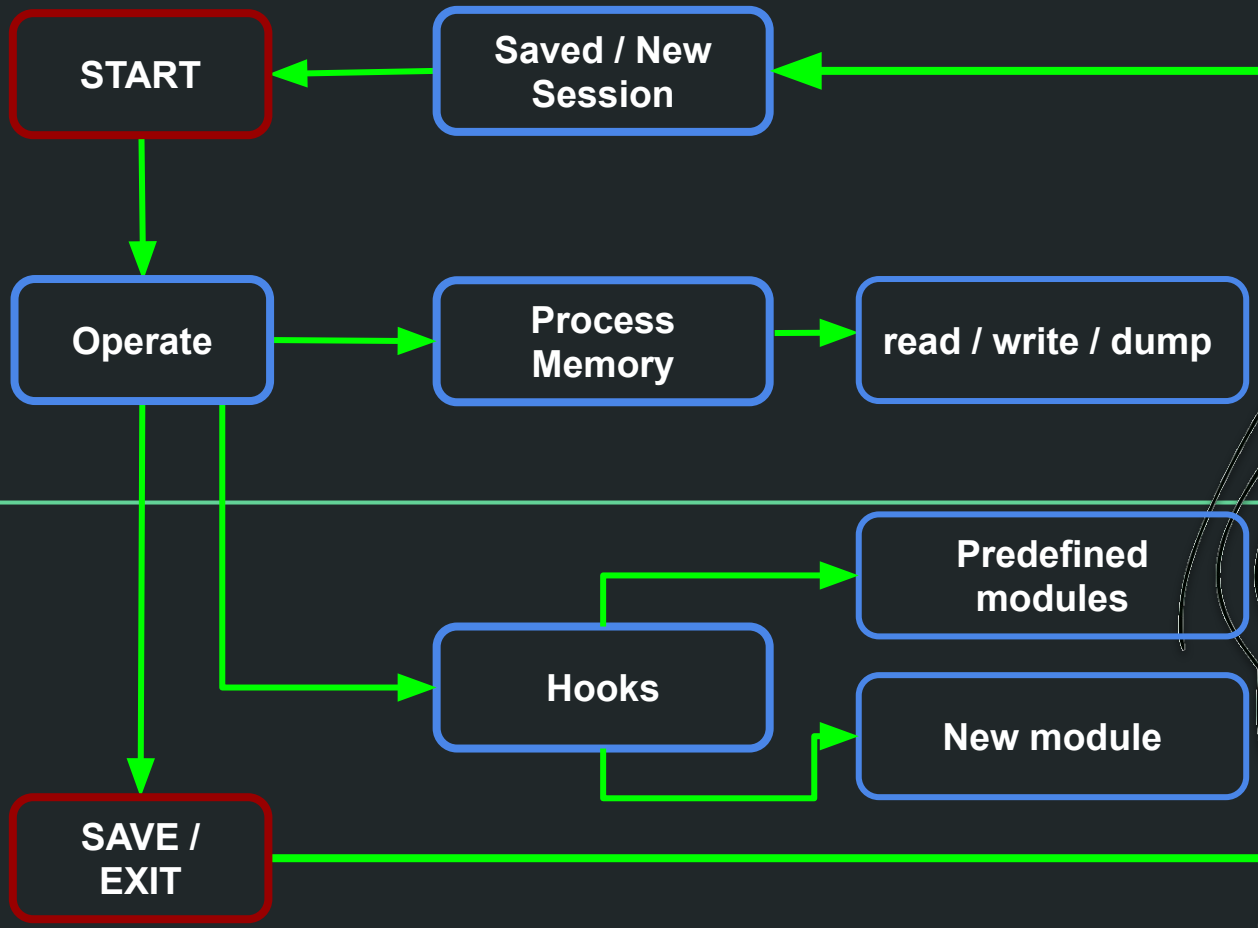
Interact

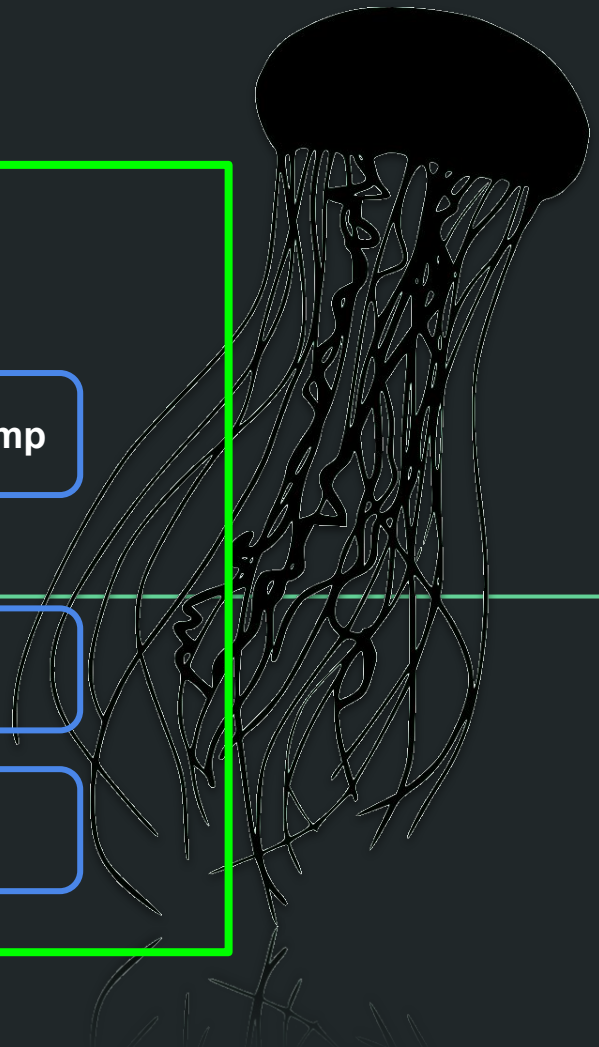
apkutils> start → [activities, services]
[Broadcast Intents]

apkutils> → [adb, shell, jdwp]



WorkFlow (medusa overview)





WorkFlow (Medusa Memory Operations)

Process Memory

Recon

Interact

medusa > libs [option] package_name //print loaded modules

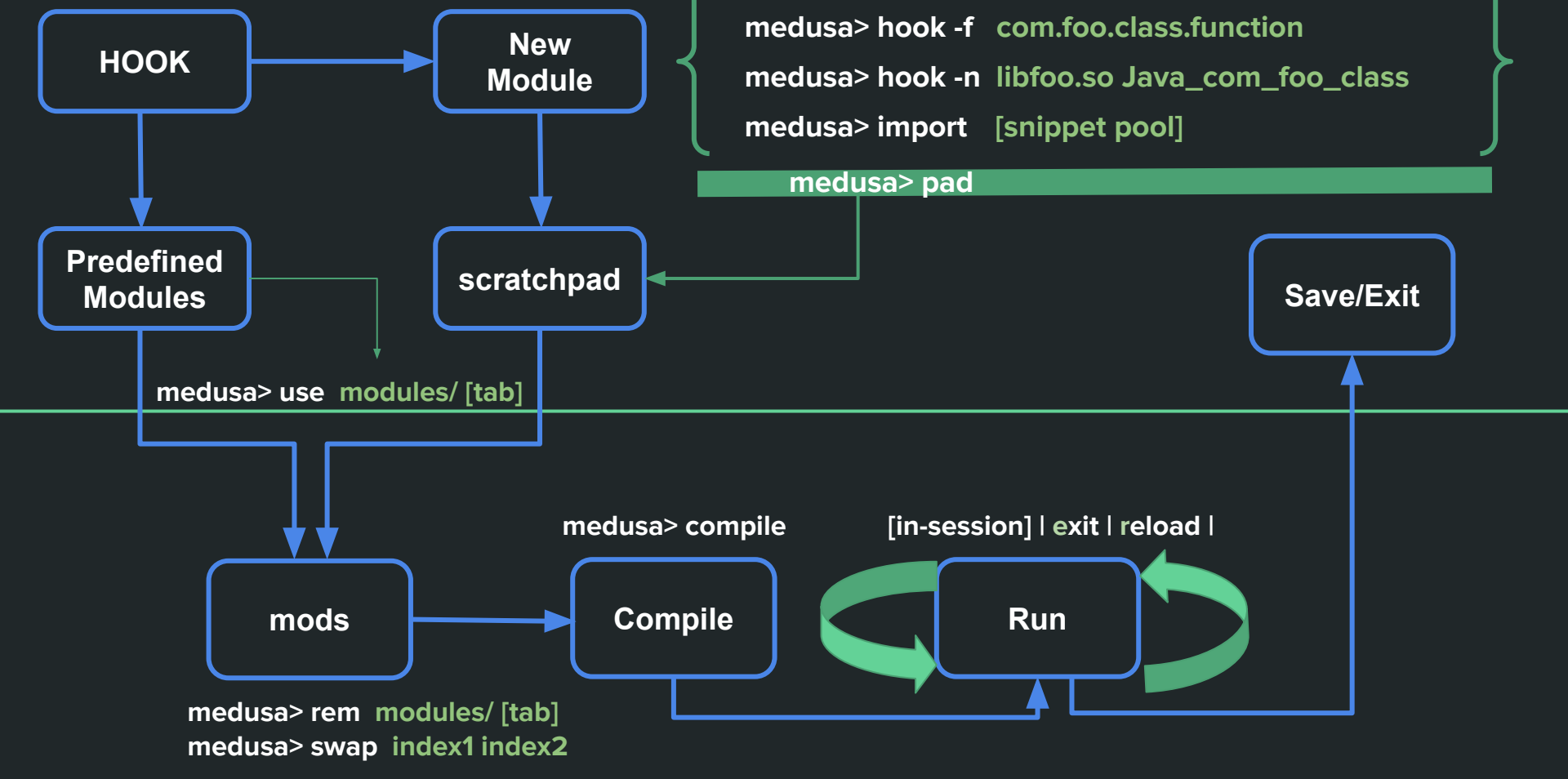
medusa > enumerate package_name libfoo //print libfoo.so exports

medusa > memops package_name libfoo.so

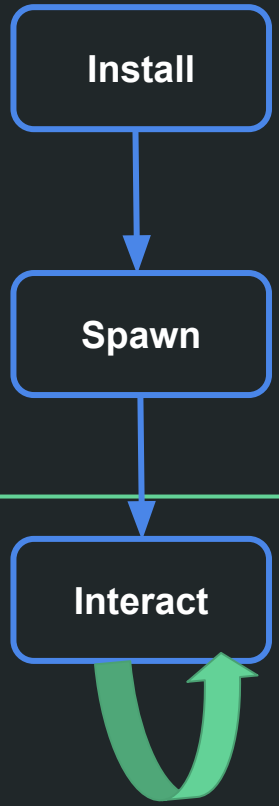
| Read@offset | Write@offset | Scan / Dump Memory |

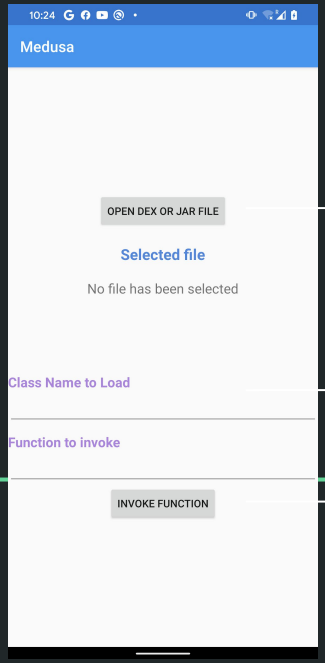
```
|(E)xit |r@offset |w@offset |⏎ |scan |(h)elp| dump|:r@0
[+] Offset:0x0
Address Range:0x6e1fae0000 --> 6e1fc67000
Module Size:1601536 Dumping at:0x6e1fae0000
           0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
00000000  7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00  .ELF............
00000010  03 00 b7 00 01 00 00 00 10 c9 05 00 00 00 00 00  ................
00000020  40 00 00 00 00 00 00 00 a0 37 17 00 00 00 00 00  @.......7......
00000030  00 00 00 00 40 00 38 00 08 00 40 00 19 00 18 00  ....@.8...@.....
```

WorkFlow (medusa hooks)



WorkFlow (medusa agent)

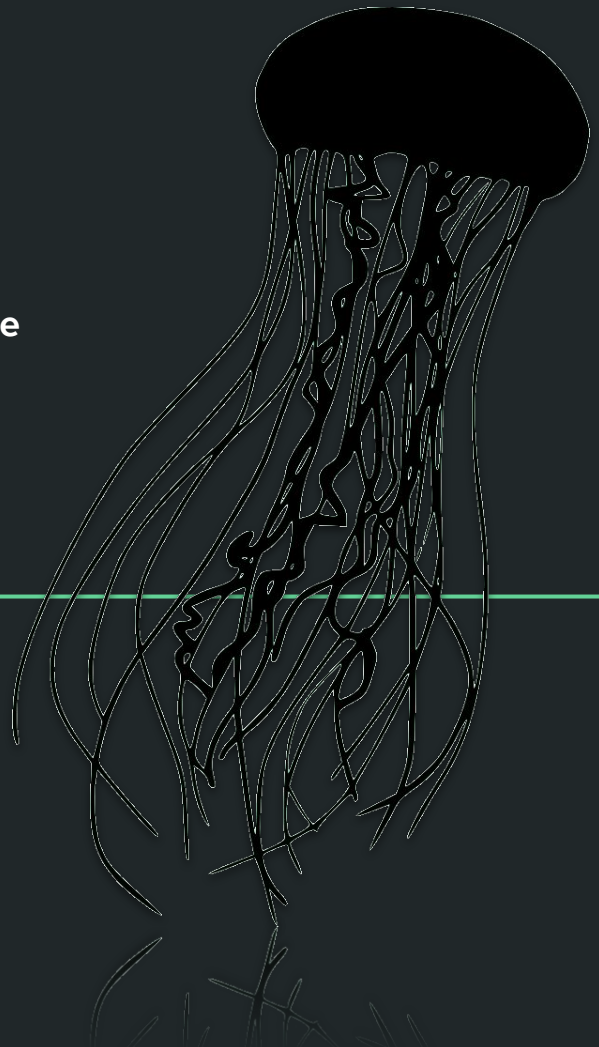




Open Dex/Jar file

Load class

Invoke function



apkutils > notify title msg

References

[1] Google Play Protect - Malware categories
(https://developers.google.com/android/play-protect/phacategories)

Credits

- https://github.com/frida/frida
- https://github.com/dpnishant/appmon
- https://github.com/brompwnie/uitkyk
- https://github.com/hluwa/FRIDA-DEXDump.git
- https://github.com/shivsahni/APKEnum
- https://github.com/0xdea/frida-scripts
- https://github.com/Areizen/JNI-Frida-Hook