### Workshop - Penetration testing Mobile devices with Frida

#### Android

- Wat is er voor nodig:
    + Android Studio / SDK

- Hoe kan je deployen?
    + Just do it ;)

- Reverse engineering
    + Apktool
    + Jadx-Gui

- Frida
    + ...

#### IOS

##### Signing
- Wat is er voor nodig:
    + OSX (virtueel?)
    + XCode
    + Apple ID
    + Apple Signing Certificate (gratis)

- Hoe kan je de eigen App deployen?
    + XCode
    + IOSdeploy
    + Resigning (applesign, codesign, appmon)

- Reverse engineering:
    + Hopper, Binary Ninja, IDA Pro, classdump
    + Strategie voor identificeren functies?

- Frida:
    + Simpel voorbeeld met een credit ophogen
    + Low level i/o: open, write, read, connect, etc
    + File browsing
    + HTTP / HTTPS afbeelden in Frida
    + Certificate pinning uitschakelen om met Burp te kijken
    + CCCRypto (encrypt / decrypt functies hooken tbv data + key)
    + Keystore

===
# Read mobile provision
```shell
MOBILEPROVISION=./embedded.mobileprovision
```

```shell
security cms -D -i "${MOBILEPROVISION}"
```

- Read its certificate

```shell
security cms -D -i "${MOBILEPROVISION}" | xmllint --xpath '/plist/dict/array/data/text(
```

- Find device UDID the profile is provisioned for:

```shell
security cms -D -i "${MOBILEPROVISION}" | xmllint --xpath '/plist/dict/key[text()="Prov
```

- Show the expiration date of the profile

```shell
security cms -D -i "${MOBILEPROVISION}" | xmllint --xpath '/plist/dict/key[text()="Expi
```

===

```
//
// We can use a class, instantiate it and execute a method
//
function example1() {
    // Check if frida has located the JNI
    if (Java.available) {

        // Switch to the Java context
        Java.perform(function() {
            const JavaString = Java.use('java.lang.String');
            var exampleString1 = JavaString.$new('Hello World, this is an example strin
            console.log('[+] exampleString1: ' + exampleString1);
            console.log('[+] exampleString1.length(): ' + exampleString1.length());
        }
    )}
}

//
// We can use overloaded versions as well
//
function example2() {
    // Check if frida has located the JNI
    if (Java.available) {

        // Switch to the Java context
        Java.perform(function() {
            const JavaString = Java.use('java.lang.String');
            const Charset = Java.use('java.nio.charset.Charset');

            // console.log(JSON.stringify(object2String(JavaString.$new.overloads), nul
            // console.log(getOverloads(JavaString.$new.overloads));

            var charset = Charset.defaultCharset()
            // Create a byte array of the string
```

```
            const charArray = "This is a javascript string converted to a byte array".s
                return c.charCodeAt(0)
            })
            exampleString2 = JavaString.$new.overload('[B', 'java.nio.charset.Charset')
            console.log('[+] exampleString2: ' + exampleString2);
            console.log('[+] exampleString2.length(): ' + exampleString2.length());
        }
    )}
}

//
// We can override methods
//
function example3() {
    // Check if frida has located the JNI
    if (Java.available) {

        // Switch to the Java context
        Java.perform(function() {
            const StringBuilder = Java.use('java.lang.StringBuilder');
            //We need to overwrite $init instead of $new, since $new = allocate + init.
            StringBuilder.$init.overload('java.lang.String').implementation = function
                if (a !== null) {
                    console.log('new StringBuilder(' + a.replace("\n", "").slice(0,10)
                }
                return this.$init(a)
            }
            StringBuilder.toString.implementation = function () {
                res = this.toString()
                if (res !== null) {
                    console.log('new StringBuilder.toString() => ' + res.replace("\n",
                }
                return res
            }

            console.log('[+] StringBuilder hooked - click around in the application')

        }
   )}

}

//
// Main
//
example1();

example2();

example3();


//
// Util functions
```

```
//

// Use this function to show an object's contents.
function object2String(obj) {
   var result = null
   if (obj && obj.constructor === Array) {
       result = []
   } else if (obj === null) {
       return null
   } else {
       result = {}
   }
   for (property in obj) {
    console.log('+'  + obj[property].constructor)
       if (obj.hasOwnProperty(property)) {
           var value = obj[property]
           if (typeof(value) === 'object' || typeof(value) === 'function') {
               if (obj.constructor === Array) {
                   result.push(showObject(value))
               } else {
                   result[property] = showObject(value)
               }
           } else {
               result[property] = value
           }
       }
   }
   return result
}

// Give this function the overloads array and it will return their signature.
function getOverloads(overloads) {
   var results = []
   for (i in overloads) {
       if (overloads[i].hasOwnProperty('argumentTypes')) {
           var parameters = []
           for (j in overloads[i].argumentTypes) {
               parameters.push("'" + overloads[i].argumentTypes[j].className + "'")
           }
       }
       // results.push(overloads[i].name + '(' + parameters.join(', ') + ')')
       results.push('.overload(' + parameters.join(', ') + ')')
   }
   return results.join('\n')
}
```