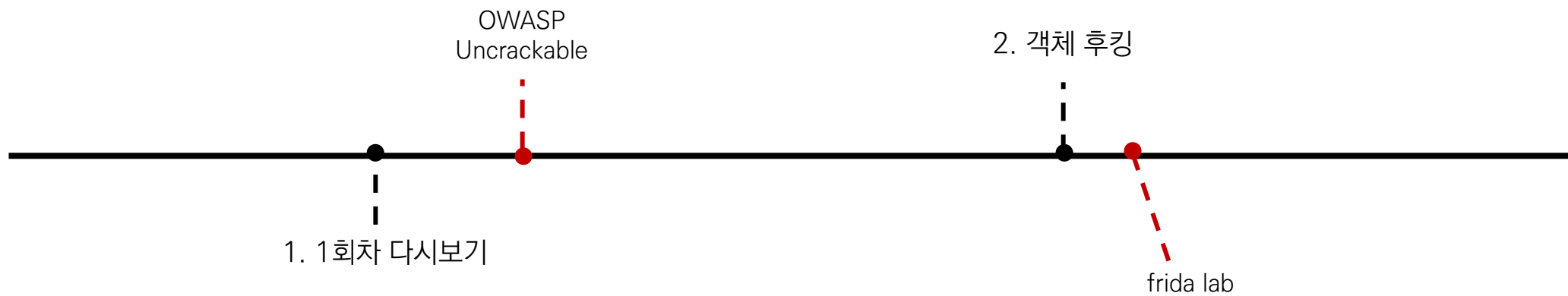


2020 안드로이드 앱 해킹 교육

PART 2. 객체 후킹

금일 교육 강의 구성

강의 타임라인



주의 사항

환경 세팅이 안되었을 때 확인해볼 사항

- 환경 변수 설정시 ADB가 아닌 녹스 설치 경로로 잘 설정 되어있는가?
- 환경변수 설정 후 환경 변수 창을 “확인“ 버튼을 통해 종료하고 cmd를 꺾다 켜는가?
- frida는 rooting이 된 상태에서에서만 동작
- 안될 경우 오류 메시지 구글에 검색

1회차 다시보기

메서드 후킹

```
Java.perform(function() {
```

```
    var _class = Java.use(클래스 이름);
```

```
    _class.[메서드 이름].implementation = function() {
```

```
        변경할 루틴
```

```
    }
```

```
});
```

메서드 후킹

`Java.perform(fn)` : ensure that the current thread is attached to the VM and call `fn`. (This isn't necessary in callbacks from Java.) Will defer calling `fn` if the app's class loader is not available yet. Use `Java.performNow()` if access to the app's classes is not needed.

해당 스레드가 VM에 올라가면 fn(function)을 실행

1회차 다시보기

메서드 후킹

```
Java.perform(function() {
```

```
    var _class = Java.use(클래스 이름);
```

```
    _class.[메서드 이름].implementation = function() {
```

```
        변경할 루틴
```

```
    }
```

```
});
```

메서드 후킹

`Java.use(className)` : dynamically get a JavaScript wrapper for `className` that you can instantiate objects from by calling `$new()` on it to invoke a constructor. Call `$dispose()` on an instance to clean it up explicitly (or wait for the JavaScript object to get garbage-collected, or script to get unloaded). Static and non-static methods are available, and you can even replace a method implementation and throw an exception from it:

인스턴스 생성, 메서드 호출, 후킹을 가능하게 해줌

1회차 다시보기

메서드 후킹

```
Java.perform(function() {
```

```
    var _class = Java.use(클래스 이름);
```

```
    _class.[메서드 이름].implementation = function() {
```

```
        변경할 루틴
```

```
    }
```

```
});
```


안드로이드 APK에 존재하는 메서드를 후킹해보자

메서드 후킹

android.hello.practice

```
class practice {  
    public int hooktarget() {  
        return 1;  
    }  
}
```

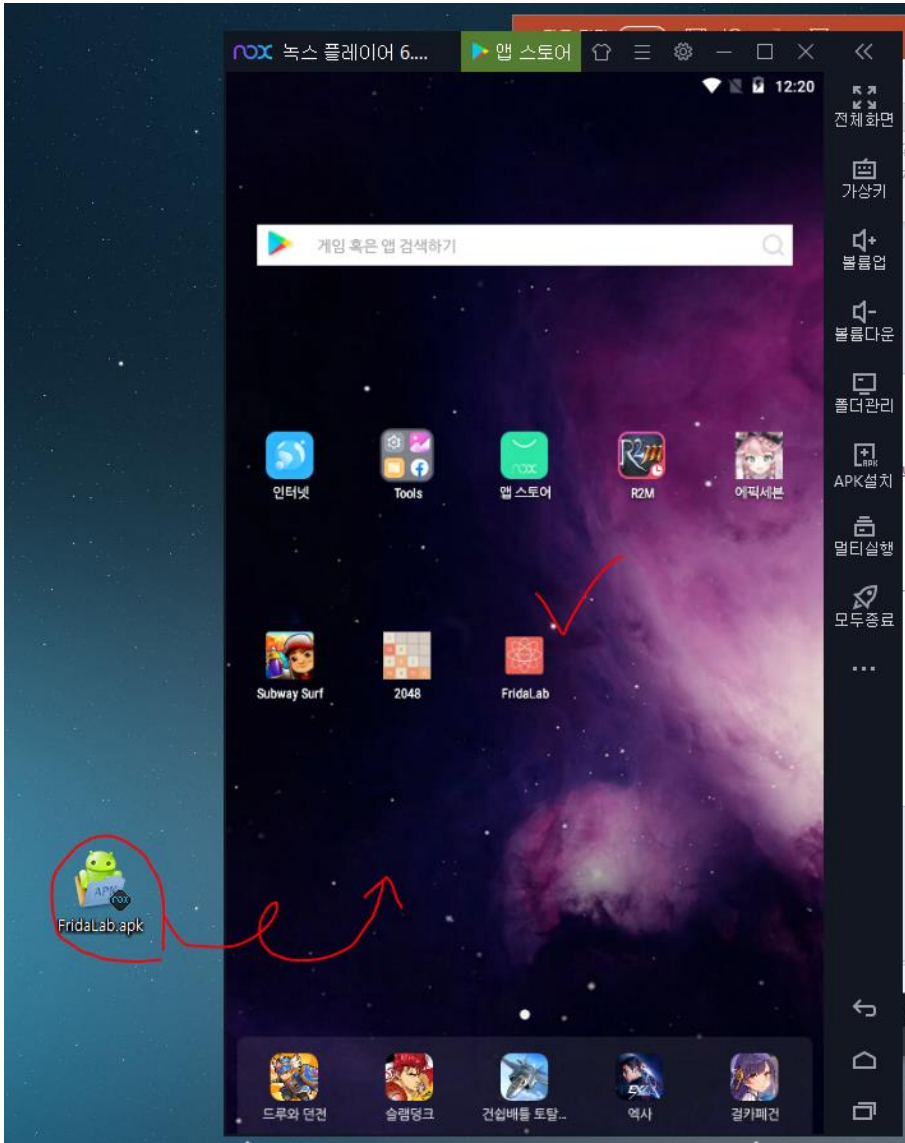
```
class practice {  
    public int hooktarget() {  
        return 0;  
    }  
}
```

후킹 코드

```
Java.perform(function() {  
    var _class = Java.use("android.hello.practice");  
    _class.hooktarget.implementation = function() {  
        return 0;  
    }  
});
```

method hooking

실습 1 – fridalab 01

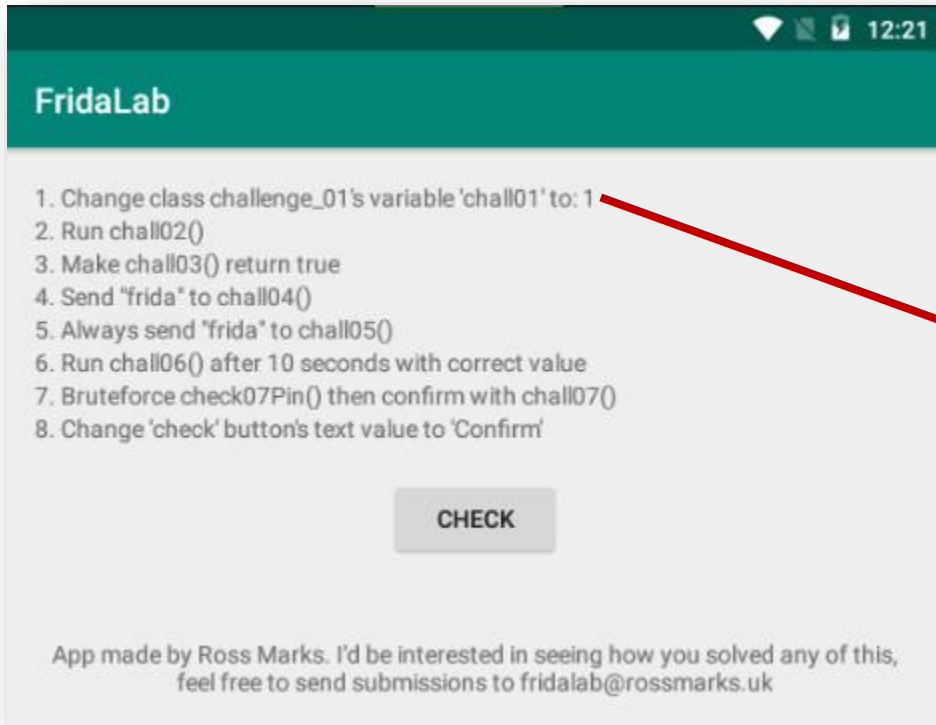


1. APK 다운로드 후 Nox에 drag&drop하여 설치
2. 실행
3. 루팅 모드 ON

method hooking

실습 1 – fridalab 01

frida lab : frida를 이용한 후킹을 연습하기 위해 만들어진 워게임



1번 문제는 특정 클래스의
멤버 변수를 변경하는 것이다

method hooking

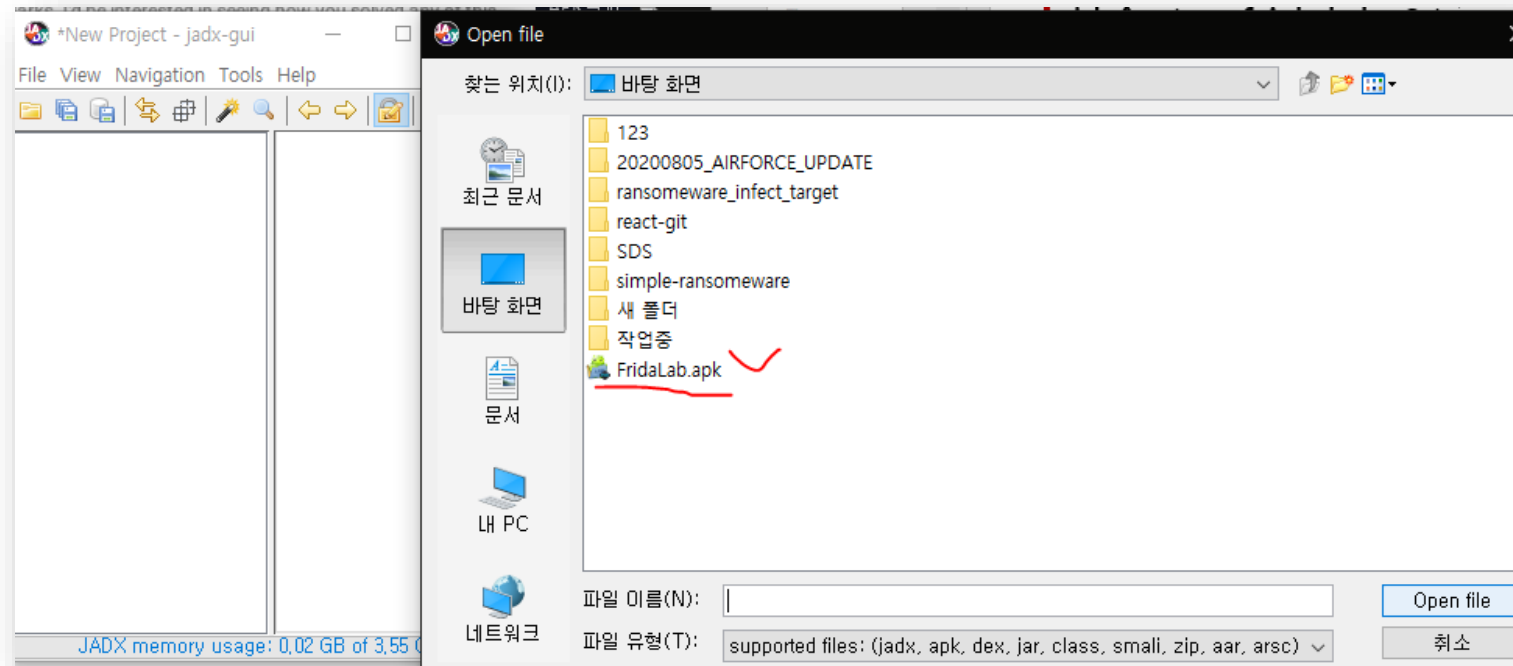
실습 1 – fridalab 01

1. `adb connect 127.0.0.1:62001`
2. `adb shell`
3. `./data/local/tmp/frida(tab) &`

3번의 tab은 자동완성

method hooking

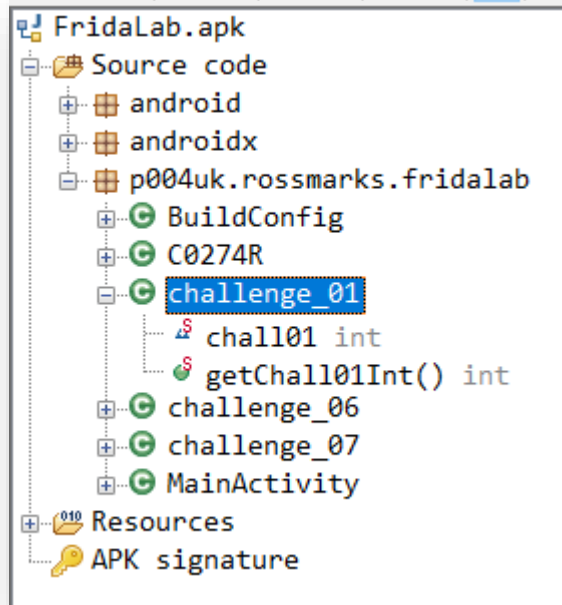
실습 1 – fridalab 01



jadx 실행 후 FridaLab.apk 열기 -> APK를 동적 분석하기 위함

method hooking

실습 1 – fridalab 01



좌측 메뉴 바에서 challenge_01로 이동

```
1 package p004uk.rossmarks.fridalab;
2
3 /* renamed from: uk.rossmarks.fridalab.challenge_01 */
4 public class challenge_01 {
5     static int chall01;
6
7     public static int getChall01Int() {
8         return chall01;
9     }
10 }
```

challenge_01의 구현 코드가 나와있음
어디를 후킹하는 것이 좋을까?

method hooking

실습 1 – fridalab 01

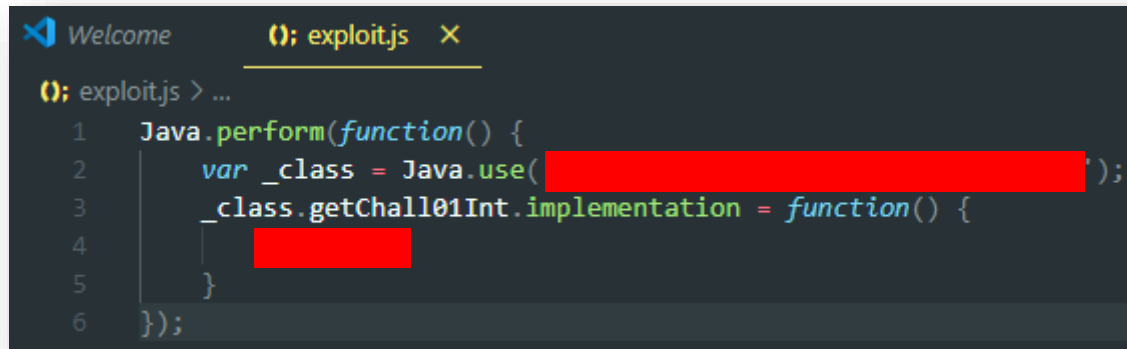
```
1 package p004uk.rossmarks.fridalab;  
2  
3 /* renamed from: uk.rossmarks.fridalab.challenge_01 */  
4 public class challenge_01 {  
5     static int chall01;  
6  
7     public static int getChall01Int() {  
8         return chall01;  
9     }  
10 }
```

앞서 배운 메서드 후킹(implementation)을 이용하여

getChall01Int()의 리턴 값을 바꾼다면?

method hooking

실습 1 – fridallab 01



```
1  Java.perform(function() {
2    var _class = Java.use( [REDACTED] );
3    _class.getChall01Int.implementation = function() {
4      [REDACTED]
5    }
6  });
```

exploit.js라는 이름의 파일을 만들고 코드를 위와 같이 작성

visual studio code가 없다면 메모장도 가능!

method hooking

실습 1 – fridalab 01

frida 후킹 스크립트 실행 방법

Frida -U -i [후킹 스크립트 이름] [대상 프로세스 이름]

우리가 작성한 js파일

frida-ps -U

method hooking

실습 1 – fridalab 01

```
1416  sdcard
1470  servicemanager
3620  su
1472  surfaceflinger
1772  system_server
1016  ueventd
3845  uk.rossmarks.fridalab
1475  vinput
1471  vold
2056  wpa_supplicant
1490  zygote

C:\Users\miny7>
```

프로세스 이름은 uk.rossmarks.fridalab

method hooking

실습 1 – fridalab 01

```
C:\Users\miny7\Desktop\123>frida -U -l exploit.js uk.rossmarks.fridalab

  ____  _
 / ___|| | | |
| |___| |_| |
 \___|_____|_|

Frida 12.8.20 - A world-class dynamic instrumentation toolkit

Commands:
  help      -> Displays the help system
  object?   -> Display information about 'object'
  exit/quit -> Exit

More info at https://www.frida.re/docs/home/

[SM G965N::uk.rossmarks.fridalab]-> _
```

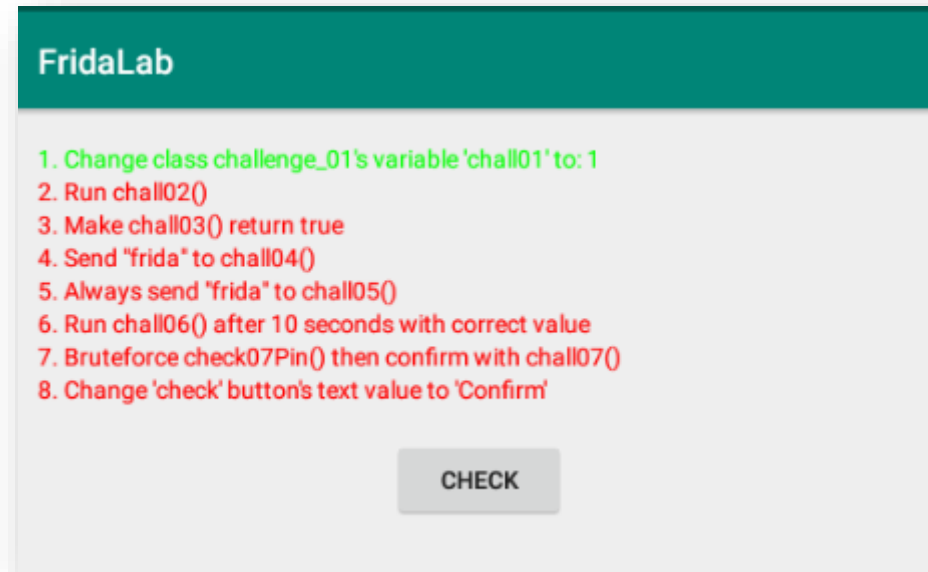
FridaLab

1. Change class challenge_01's variable 'chall01' to: 1
2. Run chall02()
3. Make chall03() return true
4. Send "frida" to chall04()
5. Always send "frida" to chall05()
6. Run chall06() after 10 seconds with correct value
7. Bruteforce check07Pin() then confirm with chall07()
8. Change 'check' button's text value to 'Confirm'

CHECK

method hooking

실습 2 – fridablab 03



Make chall03() return true

직접 풀어 보기!

인스턴스 후킹과 메서드 강제 호출

객체 후킹

객체(Object 혹은 Instance)

클래스에 선언된 모양 그대로 생성된 실체를 객체라 부름

인스턴스 후킹과 메서드 강제 호출

객체 후킹

메서드 강제 호출 과정

클래스 후킹 -> 객체 생성 -> 메서드 호출

객체 후킹

`Java.choose(className, callbacks)` : enumerate live instances of the `className` class by scanning the Java heap, where `callbacks` is an object specifying:

- `onMatch: function (instance)` : called with each live instance found with a ready-to-use `instance` just as if you would have called `Java.cast()` with a raw handle to this particular instance. This function may return the string `stop` to cancel the enumeration early.
- `onComplete: function ()` : called when all instances have been enumerated

heap space에서 동작 가능한 객체를 다 가져옴.

onMatch : 객체가 준비된 시점의 callback

onComplete : 모든 객체를 스캔했을 때 callback

객체 후킹

```
Java.perform(function() {
```

```
    Java.choose(클래스 이름, {
```

```
        onMatch: function(instance) {
```



인스턴스 저장, 변수 후킹 등

```
    },
```

```
    onComplete: function() {
```



종료시 로그 등

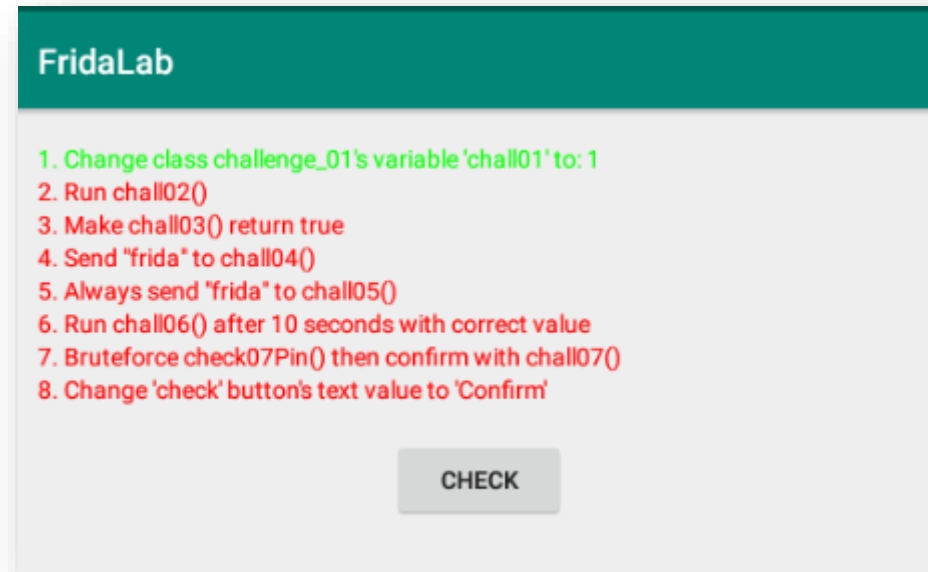
```
    }
```

```
});
```

```
});
```


object hooking

실습 3 – fridalab 02



Run chall02()

object hooking

실습 3 – fridalab 02

MainActivity에 private으로 선언되어 있음 -> 인스턴스를 통해서만 호출이 가능하다

```
private void chall02() {  
    this.completeArr[1] = 1;  
}
```

object hooking

실습 3 – fridalab 02

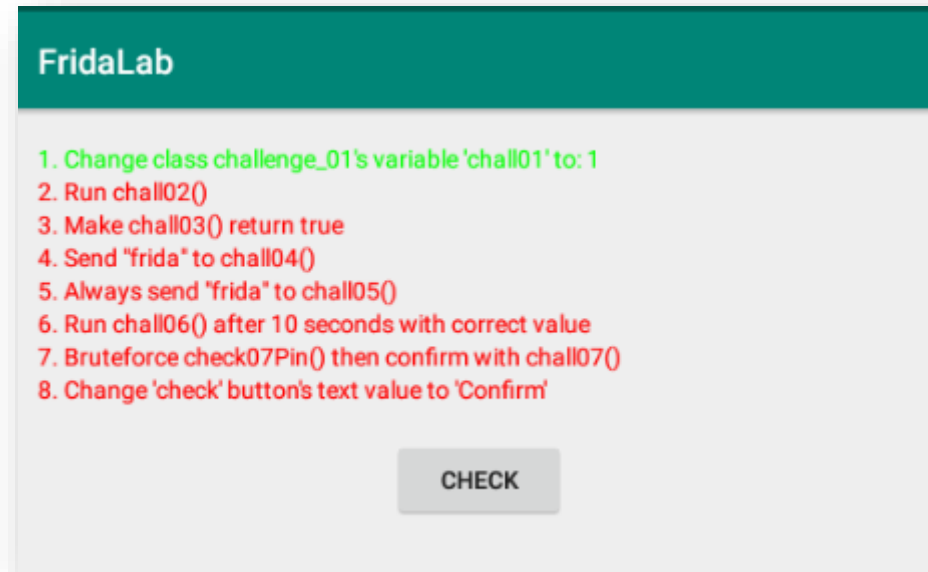
```
Java.perform(function() {  
    Java.choose("uk.rossmarks.fridalab.MainActivity", {  
        onMatch: function(instance) {  
            instance.chall02();  
        },  
  
        onComplete: function() {}  
    });  
});
```

1. Change class challenge_01's variable 'chall01' to: 1
2. Run chall02()
3. Make chall03() return true
4. Send "frida" to chall04()
5. Always send "frida" to chall05()
6. Run chall06() after 10 seconds with correct value
7. Bruteforce check07Pin() then confirm with chall07()
8. Change 'check' button's text value to 'Confirm'

CHECK

object hooking

실습 4 – fridalab 04



Send "frida" to chall04()

object hooking

실습 4 – fridalab 04

```
Java.perform(function() {  
  Java.choose("uk.rossmarks.fridalab.MainActivity", {  
    onMatch: function(instance) {  
      instance.chall04("frida");  
    },  
    onComplete: function() {}  
  });  
});
```

1. Change class challenge_01's variable 'chall01' to: 1
2. Run chall02()
3. Make chall03() return true
4. Send "frida" to chall04()
5. Always send "frida" to chall05()
6. Run chall06() after 10 seconds with correct value
7. Bruteforce check07Pin() then confirm with chall07()
8. Change 'check' button's text value to 'Confirm'

CHECK

| 과제

- frida 1~7번을 해결해오기
- 플레이스토어에 아무 어플이나 잡고 분석해보기