**MobSF**

**STATIC ANALYSIS**

**Prebuilt Docker image from DockerHub**

* docker pull opensecurity/mobile-security-framework-mobsf
* docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest

**For persistence**

* docker run -it --name mobsf -p 8000:8000 -v <your\_local\_dir>:/root/.MobSF opensecurity/mobile-security-framework-mobsf:latest

**DYNAMIC ANALYSIS**

**Create emulator:**

Open Android Studio and create emulator with configuration - API 26, CPU x86

**Start emulator:**

* LTB0206984-Mac:/ b0097042$ cd /Users/b0097042/Library/Android/sdk/emulator/
* LTB0206984-Mac:emulator b0097042$ emulator -avd Nexus\_4\_API\_28 -writable-system -no-snapshot

**Issue**:

* [PANIC: Missing emulator engine program for 'x86' CPU on Mac](https://github.com/ionic-team/ionic-framework/issues/14291#)

Make sure you're not using anything too new, I tried using 28 and 27 and neither worked but 26 (Android Oreo 8.0) did.

**Clone Repo :**

* <https://github.com/MobSF/Mobile-Security-Framework-MobSF.git>

**Setup**:

* LTB0206984-Mac:Mobile-Security-Framework-MobSF b0097042$ ./setup.sh

**Run**:

* LTB0206984-Mac:Mobile-Security-Framework-MobSF b0097042$ ./run.sh 127.0.0.1:8000
* Open browser : 127.0.0.1:8000
* Do static analysis
* Then start Dynamic analysis

**Links:**

* <https://mobsf.github.io/docs/#/dynamic_analyzer?id=android-studio-emulator>
* <https://resources.infosecinstitute.com/android-penetration-tools-walkthrough-series-mobsf/#gref>
* <https://hydrasky.com/mobile-security/how-to-use-mobile-security-frameworkmobsf/>
* <https://www.youtube.com/watch?v=hD2zK0agMJk>
* <https://kalilinuxtutorials.com/mobsf-mobile-security-framework/>
* For shell scripting : <https://stackoverflow.com/questions/39651946/find-the-value-of-key-from-json/39653436>
* <https://stackoverflow.com/questions/28185913/how-to-read-and-parse-the-json-file-and-add-it-into-the-shell-script-variable/28185962>

**Settings.py**

* ANDROID\_DYNAMIC\_ANAlYZER = "MobSF\_AVD"
* AVD\_EMULATOR = r"/Users/b0097042/Library/Android/sdk/emulator/emulator"
* ADV\_PATH = r"/Users/b0097042/.android/avd"
* ADV\_REFERENCE\_NAME = r'Nexus\_4\_API\_28'
* ADV\_DUP\_NAME = r'Nexus\_4\_API\_28'
* AVD\_ADB\_PORT = 5554

**Through Terminal**

**Start Docker container :**

* LTB0206984-Mac:MobSF b0097042$ docker run -it --name mobsf -p 8000:8000 -v $(pwd):/root/.MobSF opensecurity/mobile-security-framework-mobsf:latest | grep 'REST API Key'

**Output** : REST API Key: **28a16834ca92f19be2e349216d111eefb7f68a1c7ce3dbf9a856f33af70ed422**

**Upload build API:**

LTB0206984-Mac:MobSF b0097042$ curl -F 'file=@/Users/b0097042/Downloads/AirtelXstream-release-1.30.3-12657-playStore.apk' http://localhost:8000/api/v1/upload -H "Authorization: **28a16834ca92f19be2e349216d111eefb7f68a1c7ce3dbf9a856f33af70ed422**"

**Output** : {"scan\_type": "apk", "hash": "e964a732d9e14d8aa14827ac30f9c5d6", "file\_name": "AirtelXstream-release-1.30.3-12657-playStore.apk"}

**File scan :**

LTB0206984-Mac:MobSF b0097042$ curl -X POST --url http://localhost:8000/api/v1/scan --data "scan\_type=apk&file\_name=AirtelXstream-release-1.30.3-12657-playStore.apk&hash=e964a732d9e14d8aa14827ac30f9c5d6" -H "Authorization:**28a16834ca92f19be2e349216d111eefb7f68a1c7ce3dbf9a856f33af70ed422**"

**Display Scan:**

LTB0206984-Mac:MobSF b0097042$ curl --url "http://localhost:8000/api/v1/scans?page=1&page\_size=10" -H "Authorization:**28a16834ca92f19be2e349216d111eefb7f68a1c7ce3dbf9a856f33af70ed422**"

**PDF Report:**

LTB0206984-Mac:MobSF b0097042$ curl -X POST --url http://localhost:8000/api/v1/download\_pdf --data "hash=e964a732d9e14d8aa14827ac30f9c5d6&scan\_type=apk" -H "Authorization:**28a16834ca92f19be2e349216d111eefb7f68a1c7ce3dbf9a856f33af70ed422**"

**MobSF Jenkins Pipeline**

**Setup 1st Jenkins job to start docker container :**

* Paste following shell script in Build > Execute shell

|  |
| --- |
| *cID=$(docker run -d --rm -it --name mobsf -p 8000:8000 -v $(pwd):/root/.MobSF opensecurity/mobile-security-framework-mobsf:latest)*  *sleep 5*  *content=$(curl -L http://127.0.0.1:8000/api\_docs)*  *echo $content*  *docker logs -f $cID* |

* Then run job and copy REST API KEY from console output

**Then setup 2nd Jenkins job to upload, scan and generate report**

* Paste following shell script in Pipeline > Definition (Pipeline script) > Script

|  |
| --- |
| *#!/usr/bin/env python*  *pipeline {*  *agent any*  *parameters {*  *string (defaultValue: '/Users/b0097042/Downloads/AirtelXstream-release-1.30.3-12657-playStore.apk', description: '', name: 'INPUT\_LOCATION', trim: true)*  *string (defaultValue: '76c7a771e3f8ffdd99e829b71f1a3ed6319ce08db1831a641cf1978c659bac24', description: '', name: 'AUTH\_KEY', trim: true)*  *}*    *stages {*  *stage('Analysis') {*  *steps {*  *script {*  *upload\_cmd = "curl -s -S -F 'file=@${env.INPUT\_LOCATION}' http://localhost:8000/api/v1/upload -H 'Authorization:${env.AUTH\_KEY}'"*  *upload\_result = sh label: 'Upload Binary', returnStdout: true, script: upload\_cmd*    *def response\_map = readJSON text: upload\_result*  *def app\_type = response\_map["scan\_type"]*  *def app\_hash = response\_map["hash"]*  *def app\_name = response\_map["file\_name"]*    *scan\_start\_cmd = "curl -s -S -X POST --url http://localhost:8000/api/v1/scan --data 'scan\_type=${app\_type}&file\_name=${app\_name}&hash=${app\_hash}' -H 'Authorization:${env.AUTH\_KEY}'"*  *sh label: 'Start Scan of Binary', returnStdout: true, script: scan\_start\_cmd*    *// sh "qark --apk ${env.INPUT\_LOCATION}"*  *report = "curl POST --url http://localhost:8000/api/v1/download\_pdf --data 'hash=${app\_hash}&scan\_type=${app\_type}' -H 'Authorization:${env.AUTH\_KEY}' > Report.pdf"*  *sh label: 'Generate PDF', returnStdout: true, script: report*    *jsonData = "curl POST --url http://localhost:8000/api/v1/report\_json --data 'hash=${app\_hash}&scan\_type=${app\_type}' -H 'Authorization:${env.AUTH\_KEY}' > Report.json"*  *sh label: 'Generate JSON', returnStdout: true, script: jsonData*    *archive (includes: '\*.pdf')*  *archive (includes: '\*.json')*  *}*  *}*  *}*  *}*  *}* |

* Pass path of build (app or ipa) in first parameter
* Paste REST API KEY in second parameter
* Run the job