Part 1

Static Analysis

- For static analysis i used MobSf tool.
- First I cloned the repository from git hub.(https://github.com/MobSF/Mobile-Security-Framework-MobSF)

```
┌──(kali㉿kali)-[~/Desktop/MobSf]
└─$ git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 17597, done.
remote: Counting objects: 100% (500/500), done.
remote: Compressing objects: 100% (288/288), done.
remote: Total 17597 (delta 281), reused 370 (delta 208), pack-reused 17097
Receiving objects: 100% (17597/17597), 1.11 GiB | 5.59 MiB/s, done.
Resolving deltas: 100% (8488/8488), done.
Updating files: 100% (369/369), done.
```

- Then installed mobsf python dependencies using pip

```
┌──(kali㉿kali)-[~/Desktop/MobSf/Mobile-Security-Framework-MobSF]
└─$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  python-pip-whl python3-wheel
The following NEW packages will be installed:
  python-pip-whl python3-pip python3-wheel
0 upgraded, 3 newly installed, 0 to remove and 1341 not upgraded.
Need to get 2,284 kB/2,308 kB of archives.
After this operation, 3,669 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://kali.itsec.am/kali kali-rolling/main amd64 python-pip-whl all 20.3.4-2 [1,947 kB]
Get:2 https://kali.itsec.am/kali kali-rolling/main amd64 python3-pip all 20.3.4-2 [337 kB]
Fetched 2,284 kB in 6s (367 kB/s)
Selecting previously unselected package python-pip-whl.
(Reading database ... 262673 files and directories currently installed.)
Preparing to unpack .../python-pip-whl_20.3.4-2_all.deb ...
Unpacking python-pip-whl (20.3.4-2) ...
Selecting previously unselected package python3-wheel.
Preparing to unpack .../python3-wheel_0.34.2-1_all.deb ...
```

```
┌──(kali㉿kali)-[~/Desktop/MobSf]
└─$ sudo apt install python3-dev python3-venv python3-pip build-essential libffi-dev libssl-dev libxml2-dev libxslt1-dev libjpeg8-dev zlib1g-dev wkhtmltopdf
```

- To install run - ./setup.sh

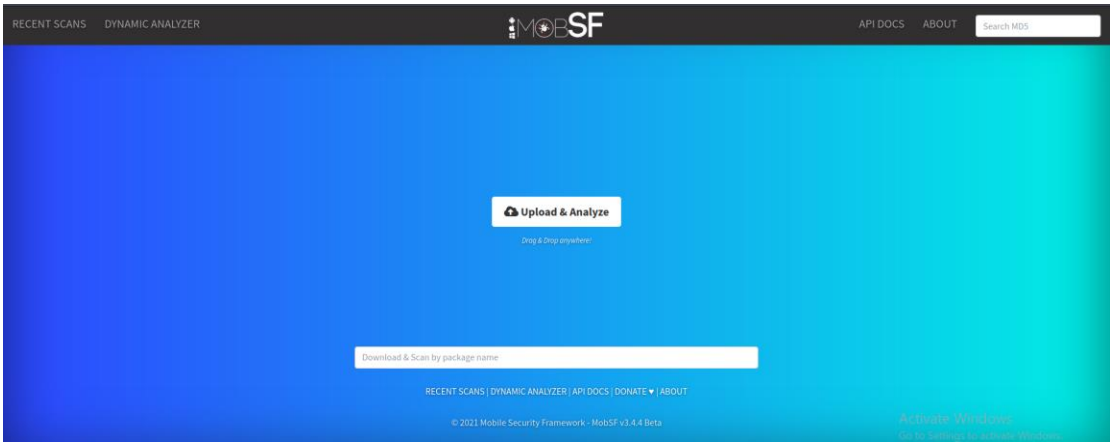
```
┌──(kali㉿kali)-[~/Desktop/MobSf/Mobile-Security-Framework-MobSF]
└─$ ./setup.sh
[INSTALL] Found Python 3.8.6
pip 20.3.4 from /usr/lib/python3/dist-packages/pip (python 3.8)
[INSTALL] Found pip
Requirement already satisfied: pip in /usr/lib/python3/dist-packages (20.3.4)
Collecting pip
  Downloading pip-21.1.2-py3-none-any.whl (1.5 MB)
     |████████████████████████████████| 1.5 MB 513 kB/s
Installing collected packages: pip
  WARNING: The scripts pip, pip3 and pip3.8 are installed in '/home/kali/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed pip-21.1.2
[INSTALL] Using python virtualenv
```

- To run mobsf- ./run.sh 127.0.0.1:8000

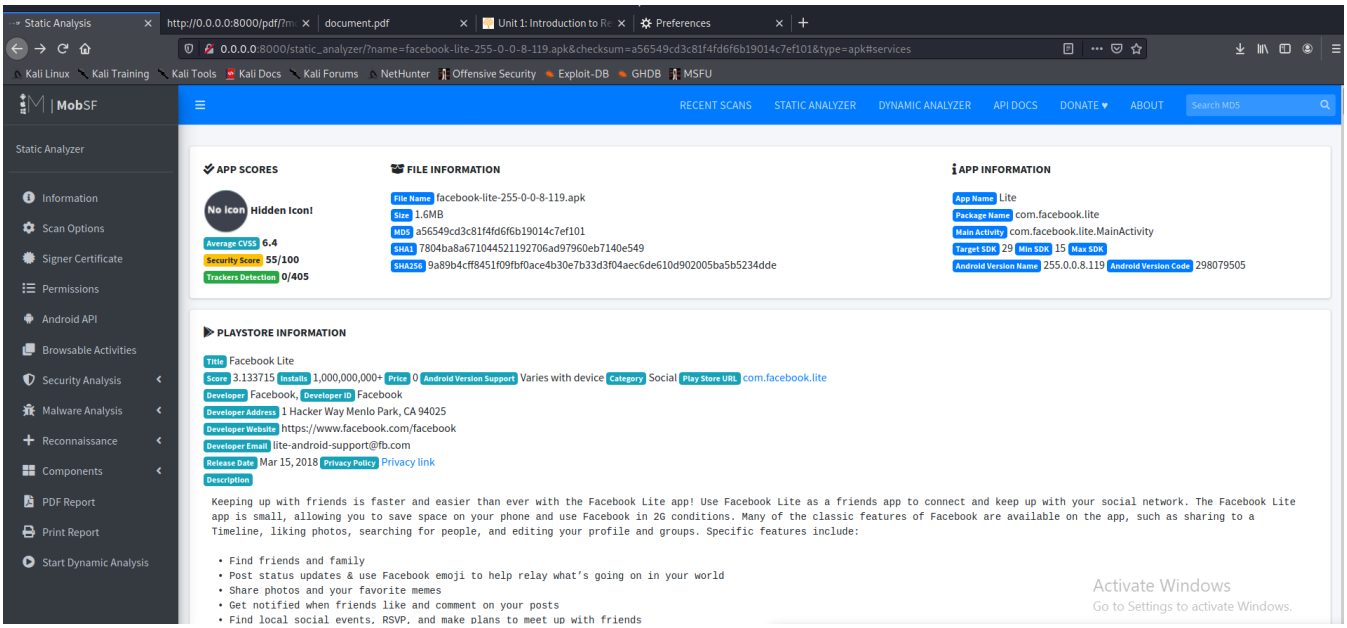
```
┌──(kali㉿kali)-[~/Desktop/MobSf/Mobile-Security-Framework-MobSF]
└─$ ./run.sh 127.0.0.1:8000
[2021-06-22 15:57:02 -0400] [9767] [INFO] Starting gunicorn 20.1.0
[2021-06-22 15:57:02 -0400] [9767] [INFO] Listening at: http://127.0.0.1:8000 (9767)
[2021-06-22 15:57:02 -0400] [9767] [INFO] Using worker: gthread
[2021-06-22 15:57:02 -0400] [9768] [INFO] Booting worker with pid: 9768
[INFO] 22/Jun/2021 19:57:41 -
 __  __       _     ____  _____        _____ _____ 
|  \/  | ___ | |__/ ___||  ___| __   _|___ /| || |
| |\/| |/ _ \| '_ \___ \| |_    \ \ / / |_ \| || |_
| |  | | (_) | |_) |__) |  _|    \ V / ___) |__   _|
|_|  |_|\___/|_.__/____/|_|       \_/ |____(_) |_|

[INFO] 22/Jun/2021 19:57:41 - Mobile Security Framework v3.4.4 Beta
REST API Key: 3ecd040056f8fd11012fb4af6f3081b3a71f42bc3cd7a1a42c051532f407d4bd
[INFO] 22/Jun/2021 19:57:41 - OS: Linux
[INFO] 22/Jun/2021 19:57:41 - Platform: Linux-5.9.0-kali1-amd64-x86_64-with-glibc2.31
[INFO] 22/Jun/2021 19:57:41 - Dist: kali 2020.4 kali-rolling
[INFO] 22/Jun/2021 19:57:41 - MobSF Basic Environment Check
[WARNING] 22/Jun/2021 19:57:41 - Dynamic Analysis related functions will not work.
Make sure a Genymotion Android VM/Android Studio Emulator is running before performing Dynamic Analysis.
[INFO] 22/Jun/2021 19:57:42 - Checking for Update.
[INFO] 22/Jun/2021 19:57:42 - No updates available.
```

- You can open the mobsf interface in browser by typing http://127.0.0.1:8000



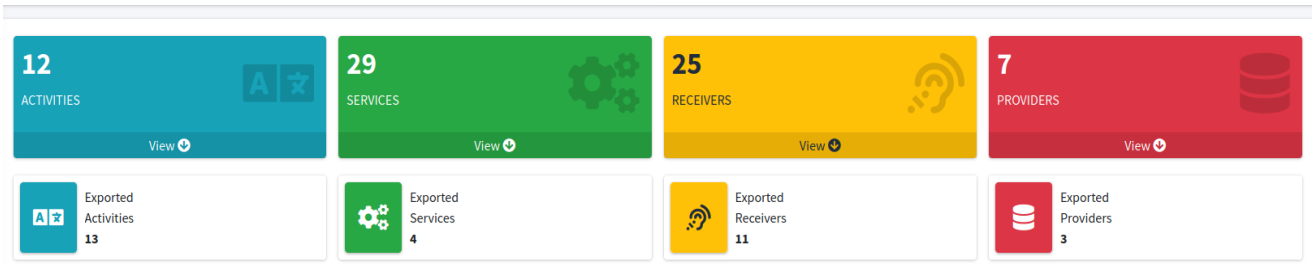
- For static analysis I use facebook-lite.apk and here is the result(You can see the complete report from this link.)



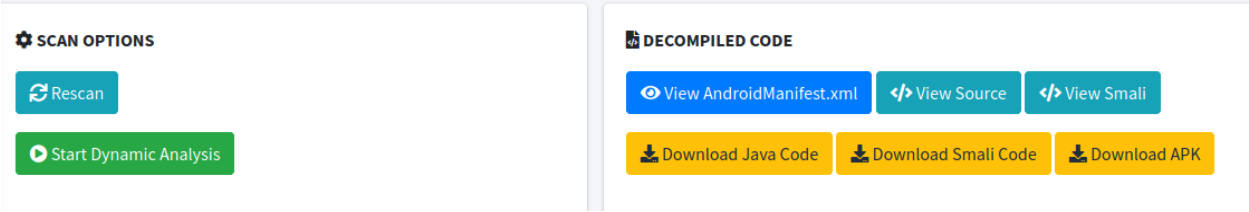
- In this report have different sections so we will go through each of them.

- On the top you have the information section where it will actually show you the app scores the various app scores like average cvs's ,the security score ,the number of trackers identified from the app ,etc

- Then we have the file information section that shows the name of the file, the size, the basic hashes of the file and the app information section shows different information about the app like the package name, the main activity ,the version,etc

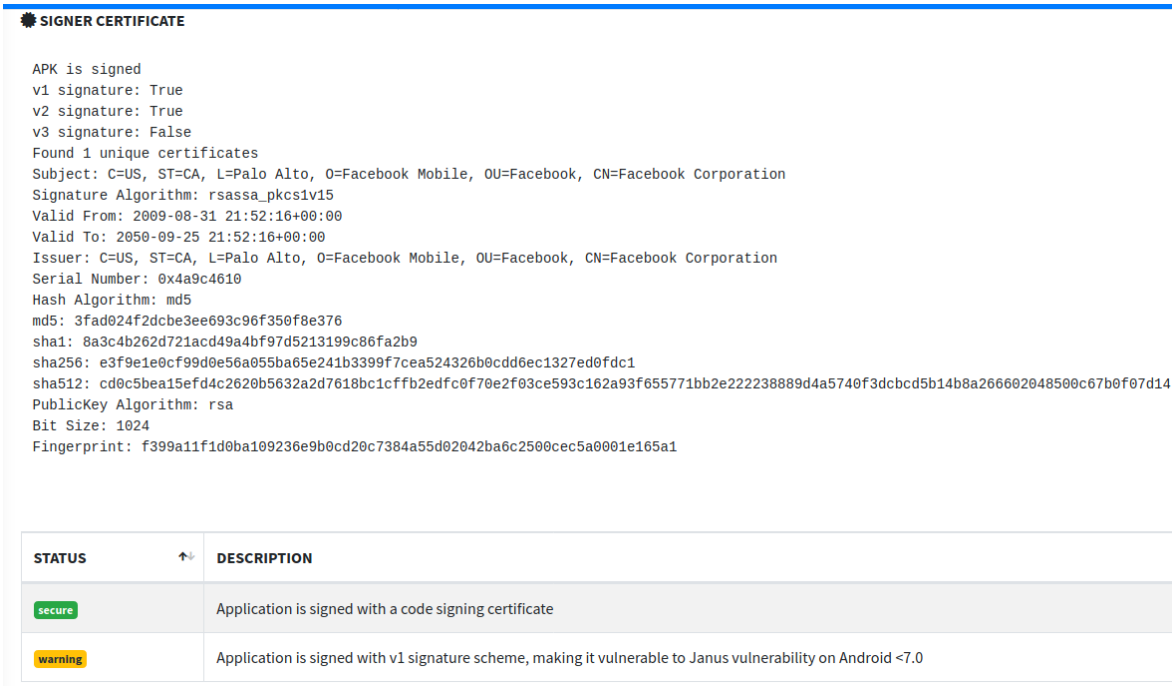


- And after that you have a section that will show you the activities ,services ,receivers and providers like the number of these components. These are like the fundamental components of an android application.It will also list out what components are exported, when you talk about exported components these are the components which can be invoked or called upon by a different app running inside the device. So if you look it from an application security perspective there are certain cases when you do not want a third party application to access or invoke a particular activity or a service in your application, so you need to be careful about that .This needs to be manually investigated and see if this is a possible security issue.



- Then comes the scan options where you can actually do a rescan or you can perform dynamic analysis,we'll come to that later. Also it shows you the decompiled code where it actually shows you the decompile android manifest file, and then you can also go through the java source code.

- similarly you can even view the smali code corresponding to the binary.mobsf also convert the dex code into smali. And this section will also allow you to download the java source code,download the smali source code or even you can download the apk itself.



- Then comes signer certificate,this section lists out the basic information about the code signing certificate like the version of the signature, the hash algorithm used ,the fingerprint ,the issuer identification,etc.Also checks on the signer certificate to see if there are any misconfigurations like ,whether the app is signed with the debug certificate or if it's actually using a weak hash algorithm things like that. if it detects anything it will actually show that in the certificate status section with a small description.That's actually quite handy when analyzing production ready android applications.According to this apk there are two status ,Secure and Warning.

APPLICATION PERMISSIONS

Search:

PERMISSION	STATUS	INFO	DESCRIPTION
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	read/modify/delete external storage contents	Allows an application to write to external storage.
android.permission.RECORD_AUDIO	dangerous	record audio	Allows application to access the audio record path.
android.permission.GET_TASKS	dangerous	retrieve running applications	Allows application to retrieve information about currently and recently running tasks. May allow malicious applications to discover private information about other applications.
android.permission.CAMERA	dangerous	take pictures and videos	Allows application to take pictures and videos with the camera. This allows the application to collect images that the camera is seeing at any time.
android.permission.CHANGE_NETWORK_STATE	normal	change network connectivity	Allows applications to change network connectivity state.
android.permission.READ_PHONE_STATE	dangerous	read phone state and identity	Allows the application to access the phone features of the device. An application with this permission can determine the phone number and serial number of this phone, whether a call is active, the number that call is connected to and so on.
android.permission.CALL_PHONE	dangerous	directly call phone numbers	Allows the application to call phone numbers without your intervention. Malicious applications may cause unexpected calls on your phone bill. Note that this does not allow the application to call emergency numbers.

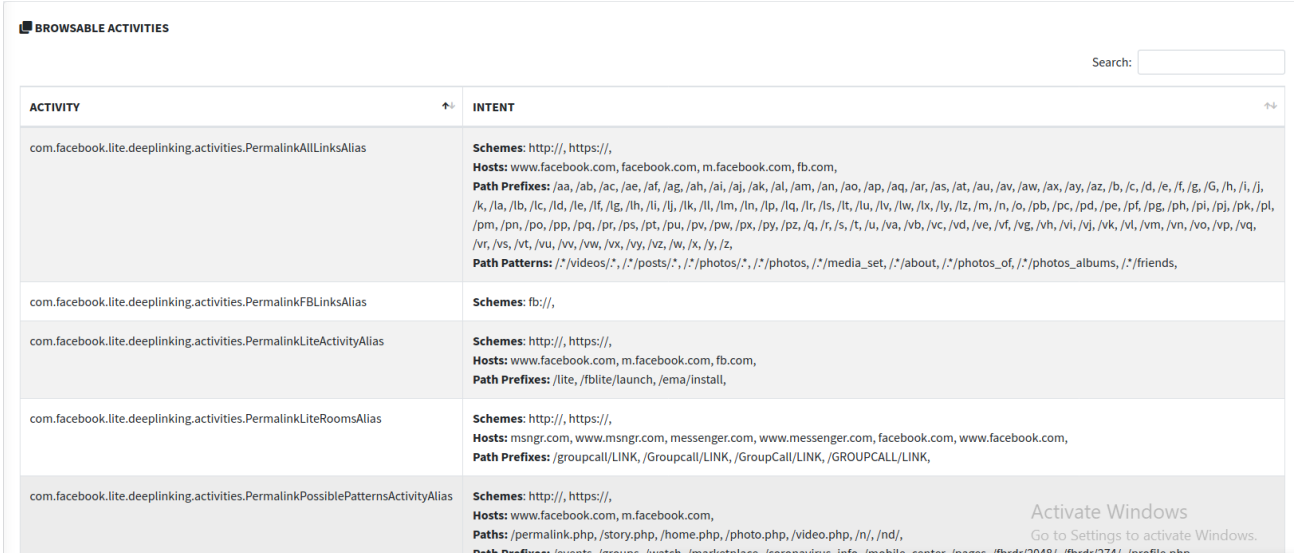
- The next section will list out permissions.You can see the table shows the permission that's used by the app ,status, info and description.So this is something that you can actually use to check and verify that the permissions used by the applications are actually necessary.As an example if we get the first permission **android.permission.WRITE_EXTERNAL_STORAGE,**It has allowed the permission to access external storage in manifest file,Which means that if you install this apk it has permission to access your data,also to modify the data.

ANDROID API

Search:

API	FILES
Base64 Decode	X/C006502p.java X/C016906u.java X/AnonymousClass068.java
Base64 Encode	X/C012605d.java X/C006502p.java X/C016906u.java X/AnonymousClass068.java X/C018207h.java X/AnonymousClass096.java
Execute OS Command	X/AnonymousClass04E.java X/AnonymousClass078.java X/AnonymousClass008.java
Get System Service	X/C008103f.java X/AnonymousClass05A.java X/AnonymousClass00R.java X/AbstractC011804v.java X/C015206d.java X/AnonymousClass002.java X/C013005h.java

- This section give you a quick idea about what this app is capable of doing from looking at the apis that's actually being used. it will give you an idea about what are the core functionalities of the app.

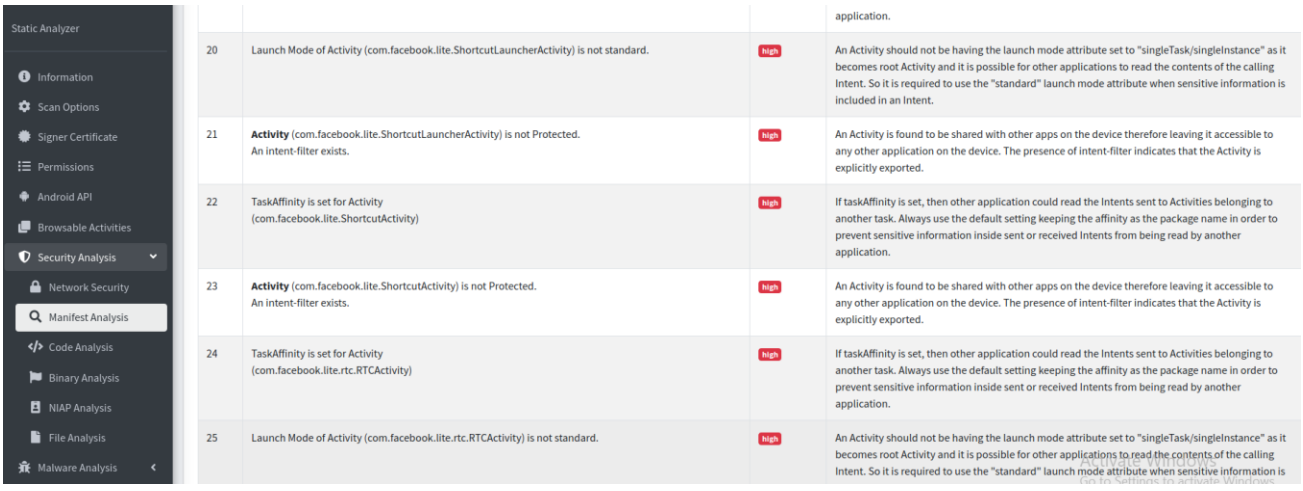


BROWSABLE ACTIVITIES

Search:

ACTIVITY	INTENT
com.facebook.lite.deeplinking.activities.PermalinkAllLinksAlias	**Schemes**: http://, https://, **Hosts:** www.facebook.com, facebook.com, m.facebook.com, fb.com, **Path Prefixes:** /aa, /ab, /ac, /ae, /af, /ag, /ah, /ai, /aj, /ak, /al, /am, /an, /ao, /ap, /aq, /ar, /as, /at, /au, /av, /aw, /ax, /ay, /az, /b, /c, /d, /e, /f, /g, /G, /h, /i, /j, /k, /la, /lb, /lc, /ld, /le, /lf, /lg, /lh, /li, /lj, /lk, /ll, /lm, /ln, /lp, /lq, /lr, /ls, /lt, /lu, /lv, /lw, /lx, /ly, /lz, /m, /n, /o, /pb, /pc, /pd, /pe, /pf, /pg, /ph, /pi, /pj, /pk, /pl, /pm, /pn, /po, /pp, /pq, /pr, /ps, /pt, /pu, /pv, /pw, /px, /py, /pz, /q, /r, /s, /t, /u, /va, /vb, /vc, /vd, /ve, /vf, /vg, /vh, /vi, /vj, /vk, /vl, /vm, /vn, /vo, /vp, /vq, /vr, /vs, /vt, /vu, /vv, /vw, /vx, /vy, /vz, /w, /x, /y, /z, **Path Patterns:** /.*/videos/.*, /.*/posts/.*, /.*/photos/.*, /.*/photos, /.*/media_set, /.*/about, /.*/photos_of, /.*/photos_albums, /.*/friends,
com.facebook.lite.deeplinking.activities.PermalinkFBLinksAlias	**Schemes**: fb://,
com.facebook.lite.deeplinking.activities.PermalinkLiteActivityAlias	**Schemes**: http://, https://, **Hosts:** www.facebook.com, m.facebook.com, fb.com, **Path Prefixes:** /lite, /fblite/launch, /ema/install,
com.facebook.lite.deeplinking.activities.PermalinkLiteRoomsAlias	**Schemes**: http://, https://, **Hosts:** msngr.com, www.msngr.com, messenger.com, www.messenger.com, facebook.com, www.facebook.com, **Path Prefixes:** /groupcall/LINK, /Groupcall/LINK, /GroupCall/LINK, /GROUPCALL/LINK,
com.facebook.lite.deeplinking.activities.PermalinkPossiblePatternsActivityAlias	**Schemes**: http://, https://, **Hosts:** www.facebook.com, m.facebook.com, **Paths:** /permalink.php, /story.php, /home.php, /photo.php, /video.php, /n/, /nd/,

- After that you have the browsable activities.This section lists out all the browsable activities like all the activities that can be browsed by a particular scheme.The scheme is a good place to start fuzzing the application. you can take out the scheme and then append the first string and perform the fuzzing.You can see how the application or the particular activity is behaving with your fuzzing input.



Static Analyzer

- Information
- Scan Options
- Signer Certificate
- Permissions
- Android API
- Browsable Activities
- Security Analysis
  - Network Security
  - Manifest Analysis
  - Code Analysis
  - Binary Analysis
  - NIAP Analysis
  - File Analysis
- Malware Analysis

			application.
20	Launch Mode of Activity (com.facebook.lite.ShortcutLauncherActivity) is not standard.	high	An Activity should not be having the launch mode attribute set to "singleTask/singleInstance" as it becomes root Activity and it is possible for other applications to read the contents of the calling Intent. So it is required to use the "standard" launch mode attribute when sensitive information is included in an Intent.
21	**Activity** (com.facebook.lite.ShortcutLauncherActivity) is not Protected. An intent-filter exists.	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.
22	TaskAffinity is set for Activity (com.facebook.lite.ShortcutActivity)	high	If taskAffinity is set, then other application could read the Intents sent to Activities belonging to another task. Always use the default setting keeping the affinity as the package name in order to prevent sensitive information inside sent or received Intents from being read by another application.
23	**Activity** (com.facebook.lite.ShortcutActivity) is not Protected. An intent-filter exists.	high	An Activity is found to be shared with other apps on the device therefore leaving it accessible to any other application on the device. The presence of intent-filter indicates that the Activity is explicitly exported.
24	TaskAffinity is set for Activity (com.facebook.lite.rtc.RTCActivity)	high	If taskAffinity is set, then other application could read the Intents sent to Activities belonging to another task. Always use the default setting keeping the affinity as the package name in order to prevent sensitive information inside sent or received Intents from being read by another application.
25	Launch Mode of Activity (com.facebook.lite.rtc.RTCActivity) is not standard.	high	An Activity should not be having the launch mode attribute set to "singleTask/singleInstance" as it becomes root Activity and it is possible for other applications to read the contents of the calling Intent. So it is required to use the "standard" launch mode attribute when sensitive information is

- After that comes security analysis.In security analysis we have manifest analysis, where we actually perform a static analysis on the android manifest files and pull out any insecure findings. You can see the issue ,the severity and the description corresponding to each issue.To explain,in 21 issue in the above figure an intent-

filter exists.That means intent is a filter which is protecting any activity.This will not secure because when intent is use ,the activity will be exposed publicly.

</> CODE ANALYSIS

Search:

NO	ISSUE	SEVERITY	STANDARDS	FILES
6	SHA-1 is a weak hash known to have hash collisions.	warning	**CVSS V2:** 5.9 (medium) **CWE:** CWE-327 Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	X/AnonymousClass096.java
7	MD5 is a weak hash known to have hash collisions.	warning	**CVSS V2:** 7.4 (high) **CWE:** CWE-327 Use of a Broken or Risky Cryptographic Algorithm **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-4	X/AnonymousClass02V.java
5	IP Address disclosure	warning	**CVSS V2:** 4.3 (medium) **CWE:** CWE-200 Information Exposure **OWASP MASVS:** MSTG-CODE-2	X/AnonymousClass02O.java
2	The App uses an insecure Random Number Generator.	warning	**CVSS V2:** 7.5 (high) **CWE:** CWE-330 Use of Insufficiently Random Values **OWASP Top 10:** M5: Insufficient Cryptography **OWASP MASVS:** MSTG-CRYPTO-6	X/AnonymousClass01Q.java X/AnonymousClass032.java X/AnonymousClass004.java X/AnonymousClass04R.java
4	This App uses SSL certificate pinning to detect or prevent MITM attacks in secure communication channel.	secure	**CVSS V2:** 0 (info) **OWASP MASVS:** MSTG-NETWORK-4	X/C0169060.java X/C0184071.java

- And then moving on you have the code analysis section.so this is the section where mobsf have performed a static analysis on all the decompile java files and if it finds any issues it will list out the issues.It's not just issues it will also list out informational issues it will also list out any good findings as well.

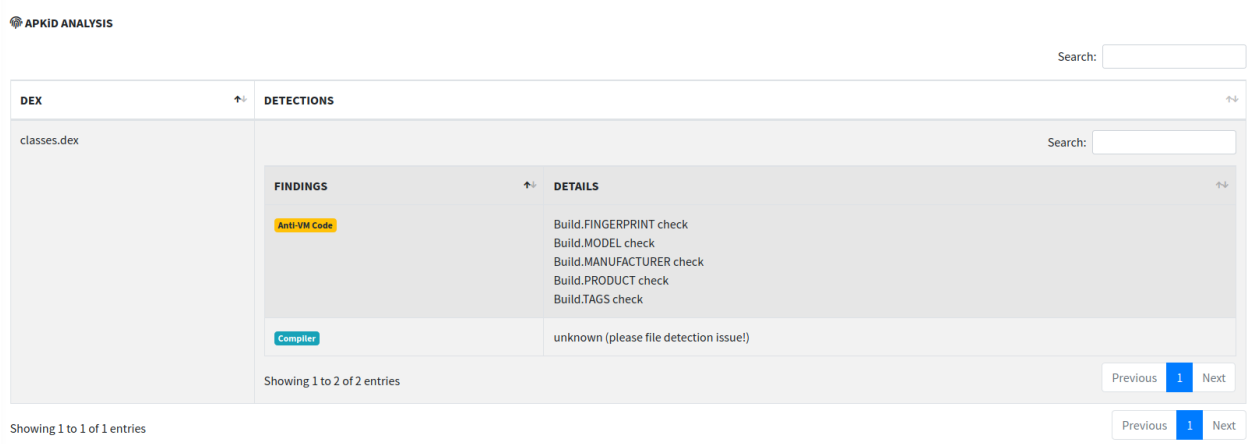
SHARED LIBRARY BINARY ANALYSIS

Search:

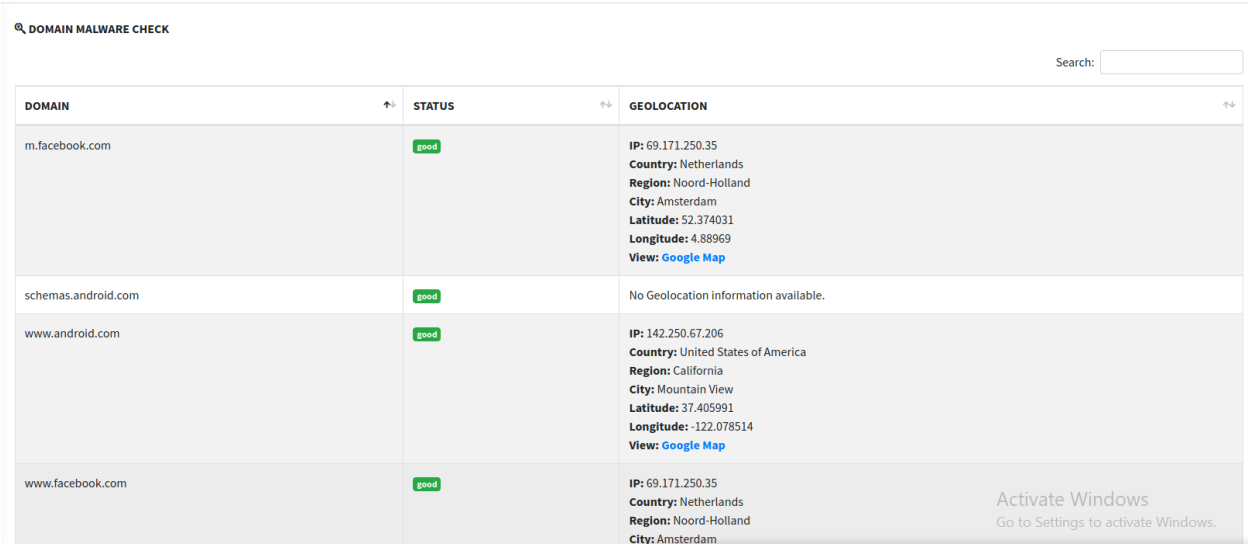
NO	SHARED OBJECT	NX	STACK CANARY	RELRO	RPATH	RUNPATH	FORTIFY	SYMBOLS STRIPPED
1	lib/armeabi-v7a/libc++_shared.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**Full RELRO** info This shared object has full RELRO enabled. RELRO ensures that the GOT cannot be overwritten in vulnerable ELF binaries. In Full RELRO, the entire GOT (.got and .got.plt both) is marked as read-only.	**False** info The shared object does not have run-time search path or RPATH set.	**False** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.
2	lib/armeabi-v7a/libsuperpack-jni.so	**True** info The shared object has NX bit set. This marks a memory page non-executable making attacker injected shellcode non-executable.	**True** info This shared object has a stack canary value added to the stack so that it will be overwritten by a stack buffer that overflows the return address. This allows detection of overflows by verifying the integrity of the canary before function return.	**Full RELRO** info This shared object has full RELRO enabled. RELRO ensures that the GOT cannot be overwritten in vulnerable ELF binaries. In Full RELRO, the entire GOT (.got and .got.plt both) is marked as read-only.	**False** info The shared object does not have run-time search path or RPATH set.	**False** info The shared object does not have RUNPATH set.	**False** warning The shared object does not have any fortified functions. Fortified functions provides buffer overflow checks against glibc's commons insecure functions like strcpy, gets etc. Use the compiler option -D_FORTIFY_SOURCE=2 to fortify functions.	**True** info Symbols are stripped.

- Next section is shared library binary analysis.what happens here is if your application is having any native code ,for example if you're using an sdk which has a native code then you will have shared objects.we do binary analysis on the shared objects,we check and see if there are some misconfigurations while it's

actually being compiled or built things like that. if you identify any security issues on the shared object that will be listed under binary analysis.

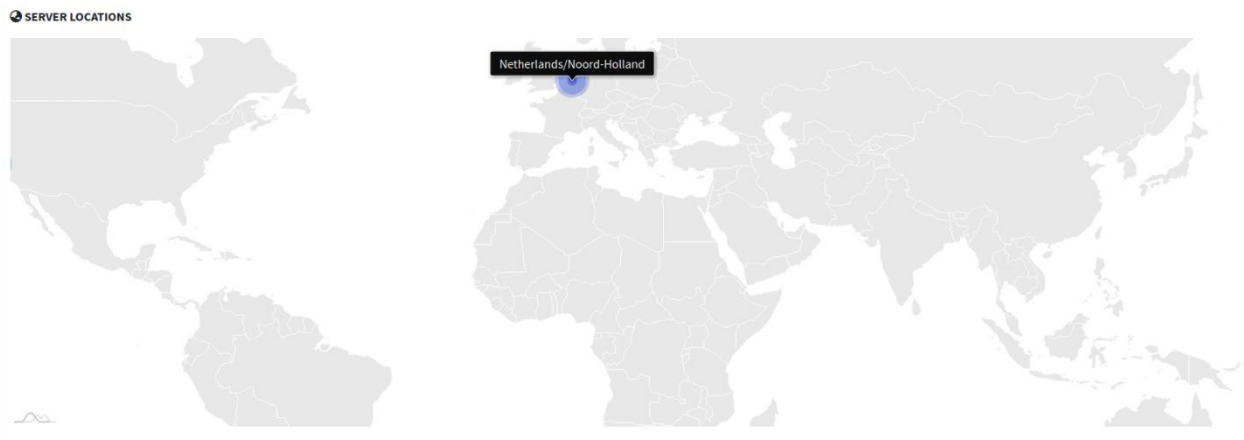


- After that we have malware analysis.Under that we have apkid analysis .apkid looks into the dex files and tells us the possible behavioral patterns like the kind of compiler being used if an obfuscator was used or anti-debugging checks.it gives you a good idea about the behavior of the application from code perspective.



- And after that we have a section called domain malware check.What it does is it actually extract out all the domains from the binary and check that against a non-list of rogue or bad domains or ips. MobSf have a database of non malware related ips and domains and all the domains inside the binary is actually checked

against those to see if it's actually present in that database.Based on that you have a status that tells if that this particular domain is good or bad.



- And it also do geolocation on all the domains that's available inside the app. you can even plot that to google map to see what's the location of the particular ip address.

URLS

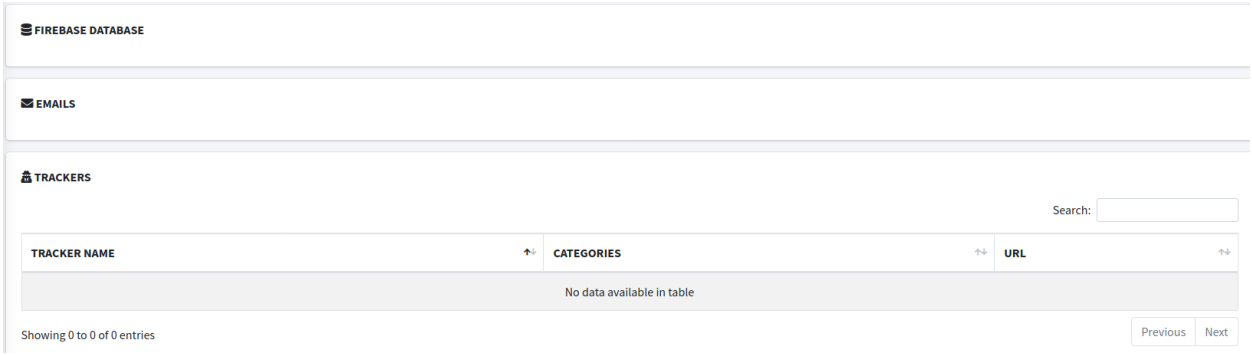
Search:

URL	FILE
http://schemas.android.com/apk/res/android	X/AnonymousClass08D.java
http://schemas.android.com/apk/res/android	X/AnonymousClass078.java
http://www.android.com/	X/AnonymousClass02S.java
https://www.facebook.com/.well-known/assetlinks.json https://m.facebook.com https://www.facebook.com	Android String Resource
www.facebook.com	X/C003701l.java
www.facebook.com	X/AnonymousClass04X.java
www.facebook.com	X/AnonymousClass08X.java

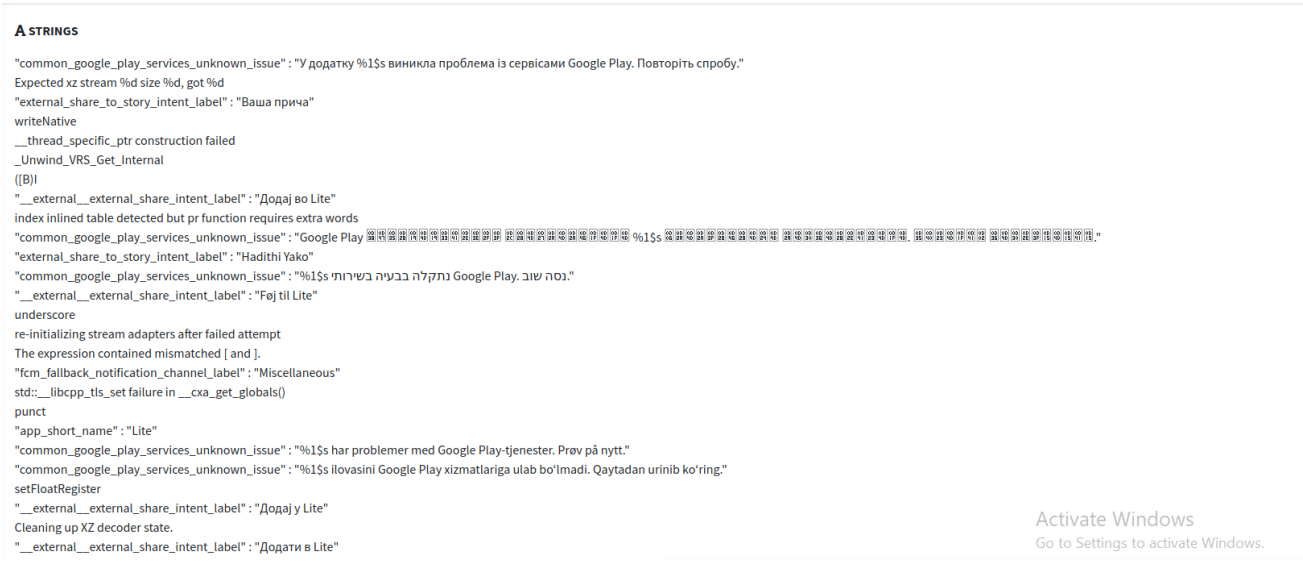
Showing 1 to 7 of 7 entries

Previous 1 Next

- Now comes the reconnaissance section.here mobsf will list out all the URLs extracted from the different source code files.



- similarly if your application is having firebase database,mobsf will also extract all the firebase database url from the app.It also performs an additional check to see if the database is exposed to the internet.
- in the next section emails ,mobsf will extract out all the emails that it found from the source code.
- And after that comes trackers.trackers are possible sdks or add-ons that actually do delemetry collection on the applications behalf.sometimes it can be a privacy concern ,it will start all the trackers that's used inside the app.



- Finally in this section we have strings,this will listout all the hardcoded strings in the binary,especially the ones from the string resource.If there is any hard-coded sensitive api keys or any other secrets in the string section that will be dumbed and shown over here.

ACTIVITIES

com.facebook.lite.MainActivity
com.google.android.gms.auth.api.signin.internal.SignInHubActivity
com.facebook.lite.ShortcutLauncherActivity
com.facebook.lite.ShortcutActivity
com.facebook.lite.rtc.RTCActivity
com.facebook.lite.webviewrtc.RTCIncomingCallActivity
com.facebook.lite.media.AlbumGalleryActivity
com.facebook.lite.photo.PreviewActivity
com.facebook.lite.storagemanager.ManageStorageActivity
com.facebook.lite.bugreporter.screencast.ScreencastActivity
com.facebook.lite.inappbrowser.common.BrowserLiteProxyActivity
com.facebook.browser.lite.BrowserLiteActivity

SERVICES

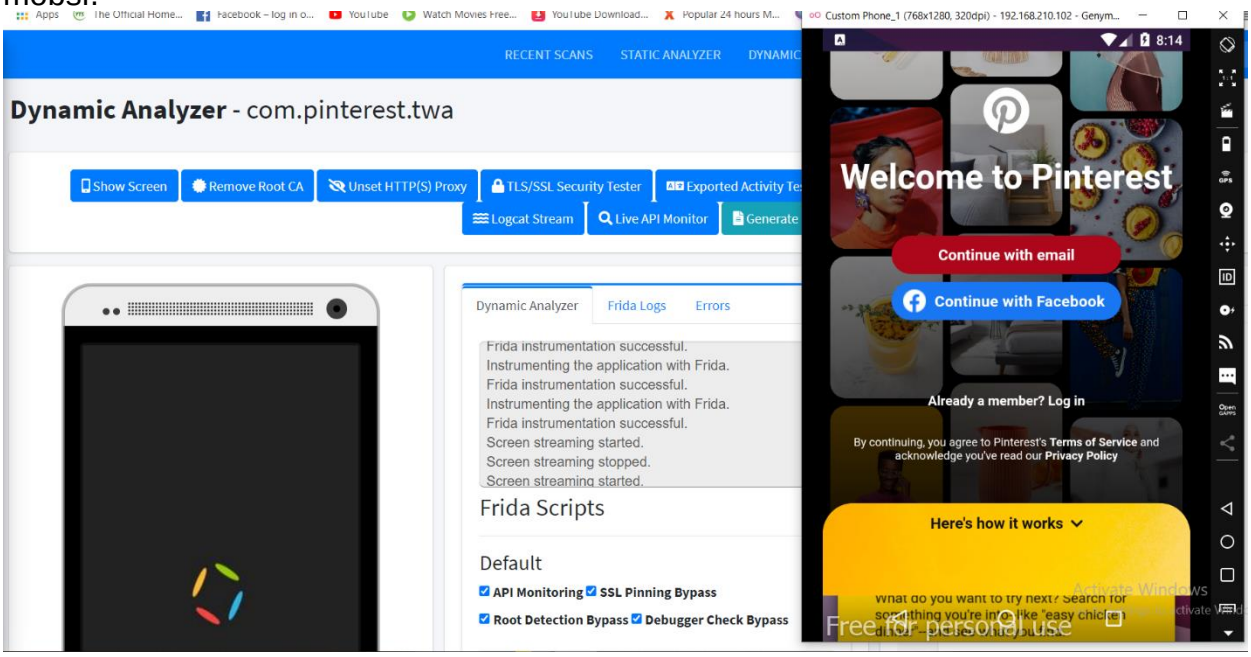
com.google.firebase.iid.FirebaseInstanceIdService
com.google.firebase.messaging.FirebaseMessagingService
com.google.android.gms.auth.api.signin.RevocationBoundService
com.facebook.lite.ForegroundService
com.facebook.lite.webviewrtc.RTCService
com.facebook.lite.FbnsIntentService
com.facebook.lite.FbnsForegroundService
com.facebook.analyticslite.memory.MemoryDumpUploadService
com.facebook.rti.push.service.FbnsService
com.facebook.lite.notification.LiteFirebaseMessagingService
com.facebook.lite.intent.WakefulIntentService
com.facebook.lite.service.SnoozeNotificationService
com.facebook.lite.service.NotificationLoggingService
com.facebook.lite.service.AppInitService
com.facebook.lite.service.TaskLifeDetectingService
com.facebook.lite.messagingapps.FirstPartyMessagingAppsDetectionService
com.facebook.lite.bugreporter.screencast.ScreencastService
com.facebook.lite.service.MediaUploadService
com.facebook.browser.lite.BrowserLiteIntentService
com.facebook.lite.browser.BrowserLiteCallbackService
com.facebook.analytics.appstatelogger.AppStateIntentService
com.facebook.appcomponentmanager.AppComponentManagerService
com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppCacheService
com.facebook.oxygen.preloads.sdk.firstparty.managedappcache.IsManagedAppCacheJobService
com.facebook.video.heroplayer.service.HeroService
com.facebook.video.heroplayer.service.MainProcHeroService
com.facebook.videolite.api.VideoUploadForegroundService

- after that we have the components which is nothing but the different components of the application.it will list out all the activities ,services, receivers ,providers and

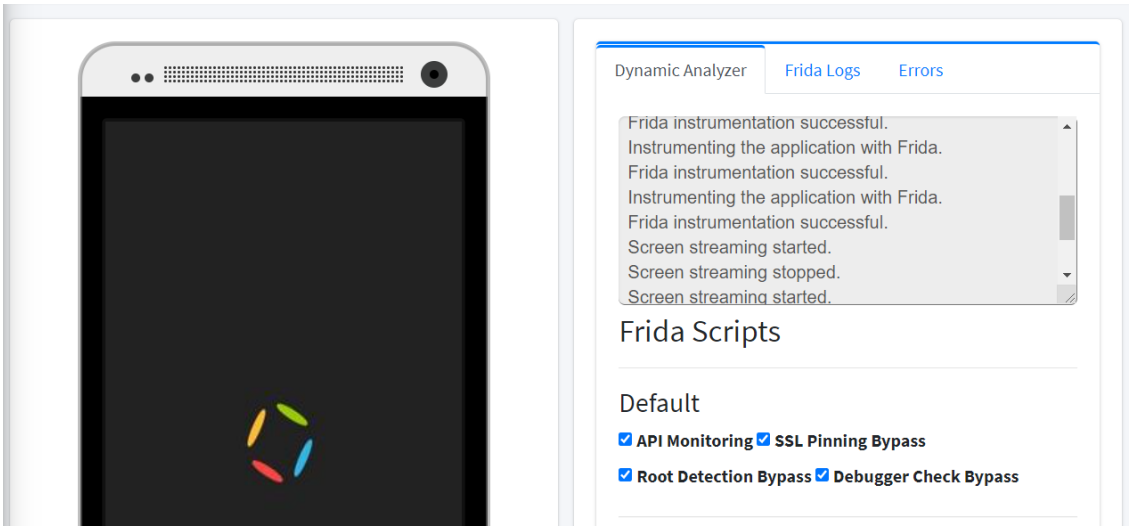
any libraries that's used by this particular application.also it will list out all the files inside the application binary

Dynamic Analysis

- For dynamic analysis, I use pinterest-lite.apk and the tool is same as again mobsf.



- Since I have use android 7.0,there are lot of options which are related to Frida.
- In order to get most out of mobsf dynamic analysis,we need to manually go through business logics and flaws.



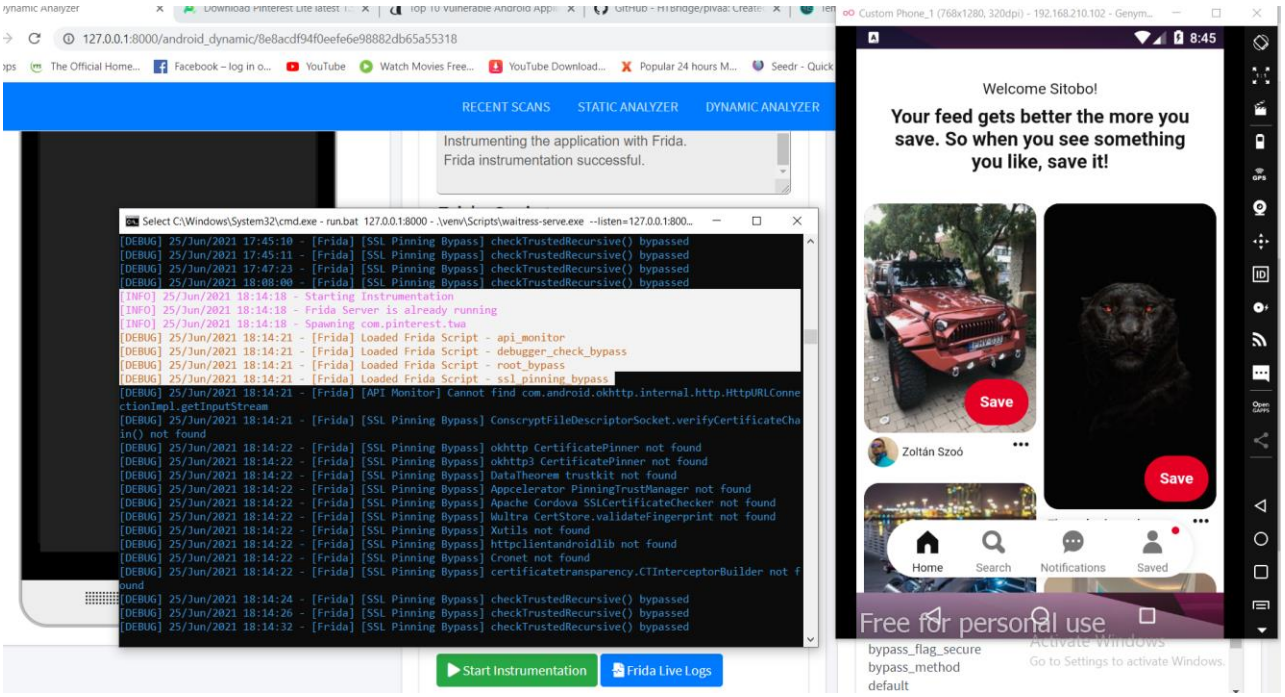
- In above figure there's an activity log in ,also if there is any errors that will be listed under the errors tab.Now below that there is some Frida scripts related option you will see a default section which will actually show you the default frida scripts that are loaded with mobsf.

- We have api monitoring script this script will monitor all the critical apis at runtime

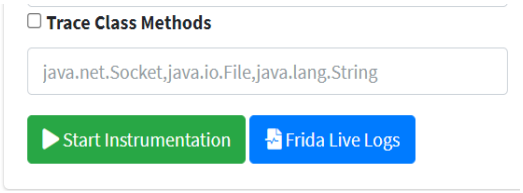
- We have the ssl pending bypass script which is responsible for bypassing ssl pinning.

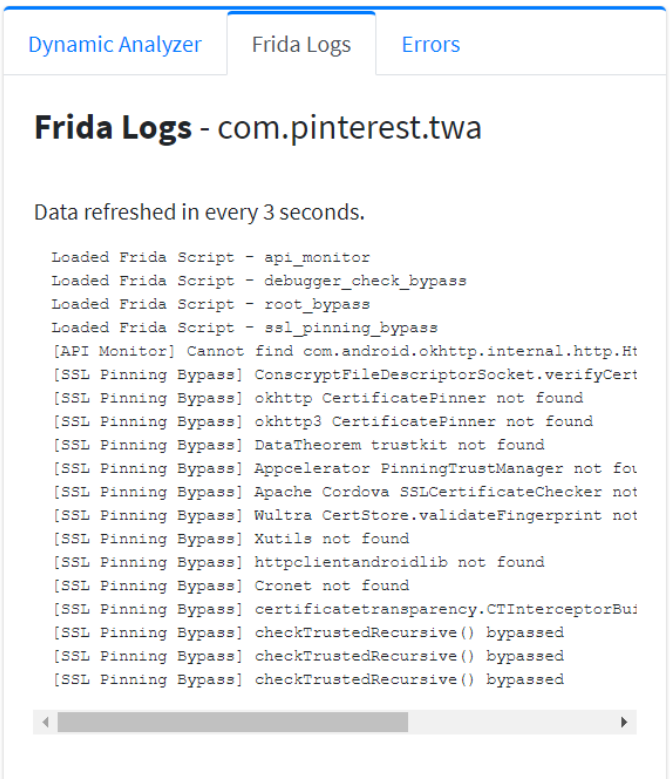
- Then there is a root detection bypass script.If your application is performing root detection it will bypass that.

- Similarly it will also bypass debugger checks using the debugger check bypass script.

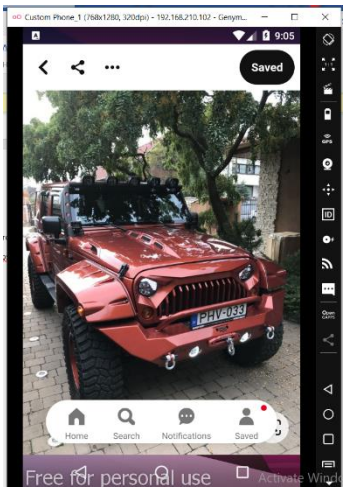


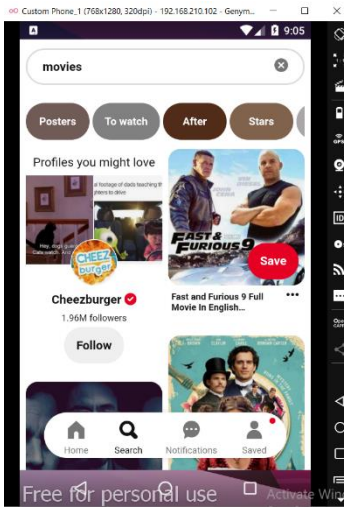
- In order to start dynamic analysis the first thing that you need to do is start instrumentation.If you click on start instrumentation you can see that your application will get loaded and it will be instrumented by mobsf.
- If you look into the console you can see the application has spawned.You can see couple of frida scripts being loaded and some messages that are coming from the frida scripts.

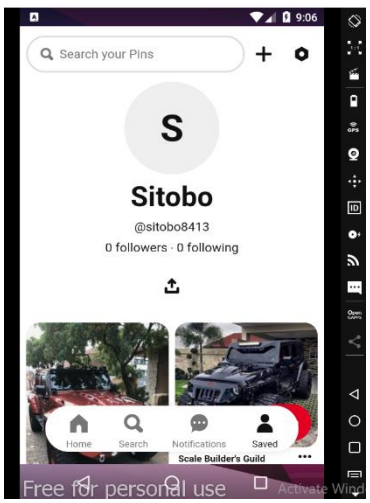


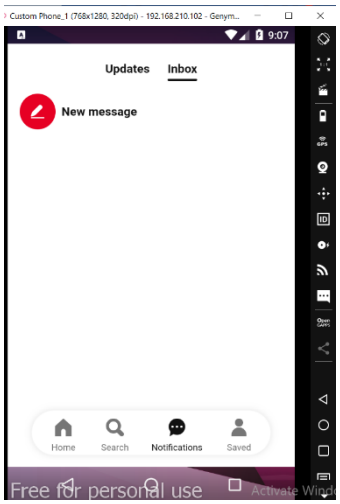


- Once you instrument the app with Frida,in order to see the output generated by these scripts you can look into the console or the best place to look into is Frida live logs.
- Frida live logs will show all the output from the different frida scripts.
- If you are running a custom frida script or if you are writing a custom frida script and if you want to see the output this is the place that you need to look in to.

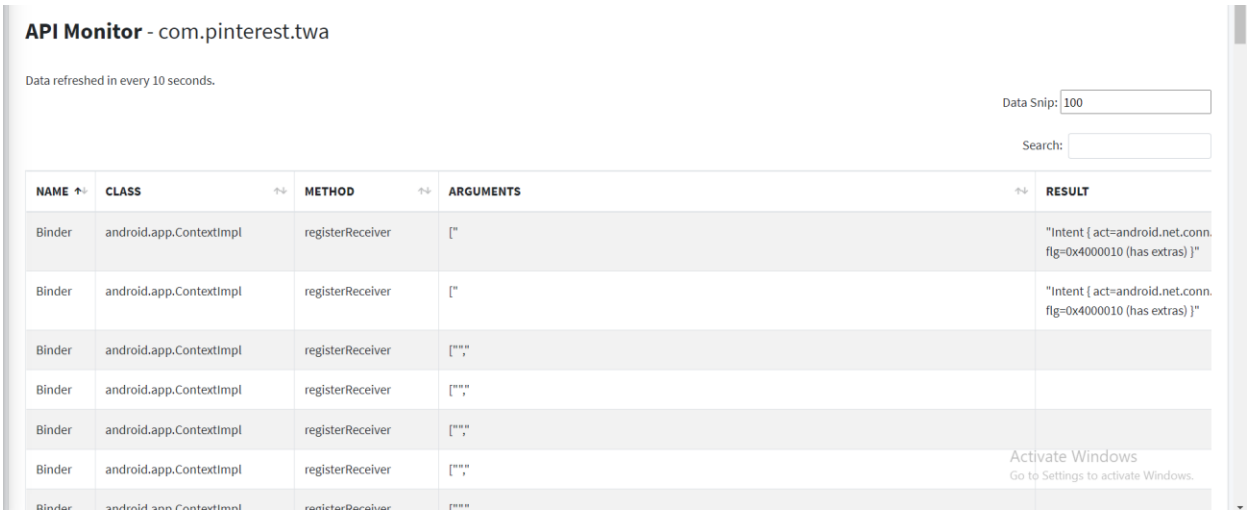


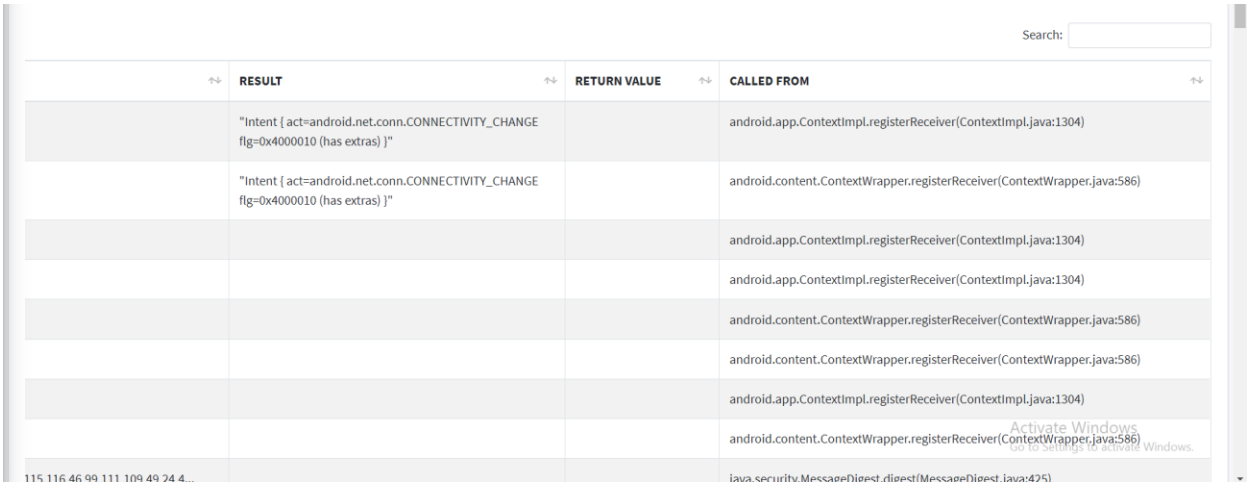




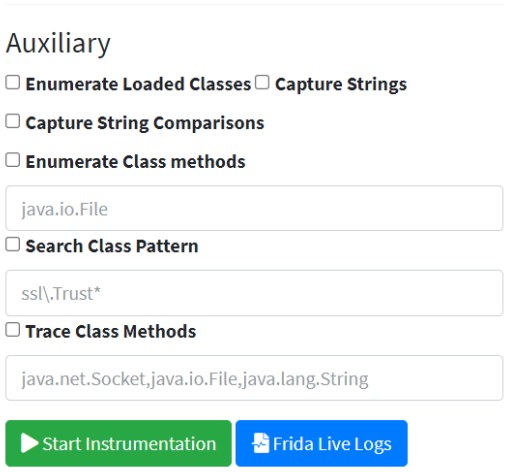


- Once you click on start instrumentation,you should be navigating through the different flaws of the app.
- Im actually going through the different flaws of the app so that mobsf can actually perform security analysis in the background.
- Now you can see that i've done some operations.





- If you scroll up you'll see a new field called live api monitor. this button will be enabled only if you instrument the app.once you instrument the app you can see the live api monitor which will capture all the api calls that are happening at runtime.
- Once inside the api monitor you can see the api name ,the class ,the method that was called at runtime, the arguments passed to the method ,the result if there was any, the return value if the method returns a value and then call from which actually shows from where that method was actually called.
- As an example if we get 1st method in above figure,this was called from contextImpl.java at line number 1304.



- Now we will talk about auxiliary scripts so these are some very useful script that you can use while you are performing dynamic analysis of an android application.

```
[*] android.text.style.AlignmentSpan
[*] sun.security.jca.ServiceId
[*] android.system.StructPollfd
[*] android.content.pm.ActivityInfo
[*] java.lang.reflect.Field
[*] android.icu.impl.StandardPlural
[*] java.lang.Enum
[*] android.content.pm.ConfigurationInfo
[*] com.android.okhttp.HttpUrl$Builder$ParseResult
[*] java.util.concurrent.RunnableScheduledFuture
[*] java.security.CryptoPrimitive
[*] android.text.style.SpellCheckSpan
[*] java.beans.PropertyChangeListener
[*] android.icu.lang.UScript$ScriptUsage
[*] java.util.Locale
[*] android.icu.impl.UCharacterProperty$IntProperty
[*] android.graphics.drawable.GradientDrawable$Orientation
[*] com.android.internal.policy.PhoneWindow$PanelFeatureState
[*] android.content.pm.ServiceInfo
[*] sun.security.util.DisabledAlgorithmConstraints$KeySizeCor
[*] android.text.style.CharacterStyle
[*] android.animation.Keyframe$FloatKeyframe
```

- let's start with Enumerate Loaded Classes as the name suggests it will enumerate all the classes loaded at runtime.Let's click on start instrumentation,our app is loaded.now if you look into frida live logs you can see the classes that are loaded at runtime.

```
[SSL Pinning Bypass] certificatetransparency.CTInterceptorBui
[*] [String Catch] [0] Failed to init handler: Attempt to inv
[*] [String Catch] [0] cr_SBApiBridge
[*] [String Catch] [0] en-US
[*] [String Catch] [0] en-US
[*] [String Catch] [0] /system/etc/security/cacerts
[*] [String Catch] [0] /data/misc
[*] [String Catch] [0] /data/misc/user
[*] [String Catch] [0] /data/misc/user/0
[*] [String Catch] [0] /data/misc/user/0/cacerts-removed
[*] [String Catch] [0] /system/etc/security/cacerts
[*] [String Catch] [0] android.app.ContextImpl.registerReceiv
[*] [String Catch] [0] (ContextImpl.java:1304)
[*] [String Catch] [0] android.app.ContextImpl.registerReceiv
[*] [String Catch] [0] android.app.ContextImpl.registerReceiv
[*] [String Catch] [0] (ContextWrapper.java:586)
[*] [String Catch] [0] android.content.ContextWrapper.registe
[*] [String Catch] [0] (ContextWrapper.java:586)
[*] [String Catch] [0] android.content.ContextWrapper.registe
[*] [String Catch] [0] android.content.ContextWrapper.registe
[*] [String Catch] [0] (PG:42)
[*] [String Catch] [0] org.chromium.net.X509Util.d(PG:42)
[*] [String Catch] [0] (PG:3)
[*] [String Catch] [0] org.chromium.net.X509Util.a(PG:3)
[*] [String Catch] [0] (PG:85)
[*] [String Catch] [0] org.chromium.net.X509Util.a(PG:85)
```

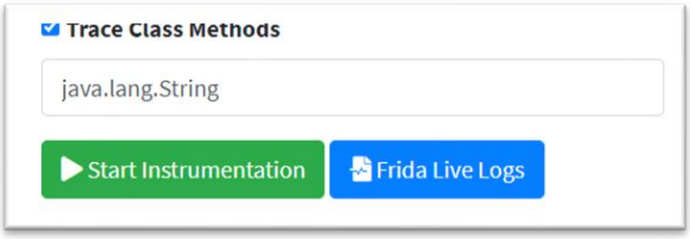
- Similarly there is another auxiliary script called capture strings.what it does is it try to capture all the strings at runtime and you'll see that there are a couple of strings being called by this auxiliary script in Frida live logs.

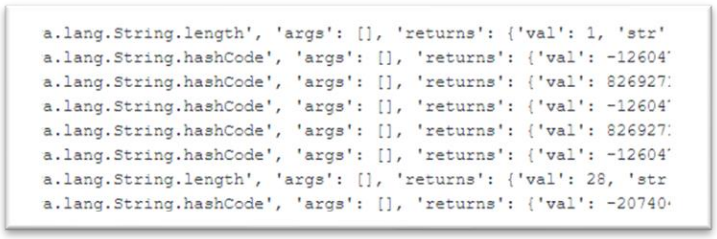
```
checkTrustedRecursive() bypassed
https://www.pinterest.com/search/ == https://www.pinterest.com
https://www.pinterest.com/search/pins/?q=sliit&rs=rs == https:
https://www.pinterest.com/search/pins/?q=sliit&rs=rs == https:
https://www.pinterest.com/search/pins/?q=sliit&rs=rs == https:
https://www.pinterest.com/search/pins/?q=sliit&rs=rs == https:
https://www.pinterest.com/search/?rs=typed&q=sliit&scope=pins
https://www.pinterest.com/search/pins/?rs=typed&q=cyber securi
https://www.pinterest.com/search/ == https://www.pinterest.com
https://www.pinterest.com/notifications/ == https://www.pinter
https://www.pinterest.com/inbox/ == https://www.pinterest.com/
https://www.pinterest.com/inbox/ == https://www.pinterest.com/
https://www.pinterest.com/inbox/ == https://www.pinterest.com/
Lumindu == Lumindu ? true
```

- Now let's see capture string comparison.This is a useful auxiliary script that will capture all the string equal comparisons .On frida live logs you can see all the string comparisons that happened at runtime. The last line in above figure the two names are equal and result it as a true statement.(Lumindu=Lumindu)

```
*] Getting Methods and Implementations of Class: java.io.File
*] public static java.io.File java.io.File.createTempFile(java
*] public static java.io.File java.io.File.createTempFile(java
*] private static java.io.File java.io.File.generateTempFile(j
*] public static java.io.File[] java.io.File.listRoots()
*] private synchronized void java.io.File.readObject(java.io.O
*] private static java.lang.String java.io.File.slashify(java.
*] private synchronized void java.io.File.writeObject(java.io.
*] public boolean java.io.File.canExecute()
```

- Now we have an auxiliary script called enumerate class methods .like the name suggest it will enumerate all the methods of a class .for example if you have a class and you want to know what are the methods supported you can give that.As an example I typed java.io.File and click on start instrumentation.If you look at frida live logs you can see the different methods supported by the class. It will get all the methods and implementations of the class java.io.file





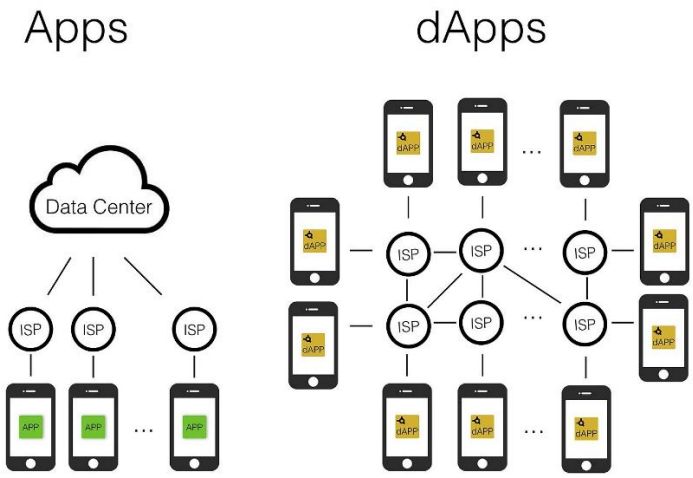
- Finally we have something called trace class methods .this is another useful and handy feature that allows you to trace class methods .Like for example if you have a class and if you want to see what are the methods of the class that are invoked at runtime, what are the arguments passed, what's return value ,similar to that of api monitor but if you want to do that on a custom method you can actually provide the class name.So that it will trace all the class methods.As an example I use java.lang.string and instrument the class. let's look into the frida logs and you can see that class tracing all the method implementation of java.lang.string .There are methods called hashcode,arguments and return value.This helps you to isolate monitoring to a particular class like what are the methods of a particular class that are being called at runtime , what are the arguments being passed, what's the return value.

- **You can see the generated report from here.**

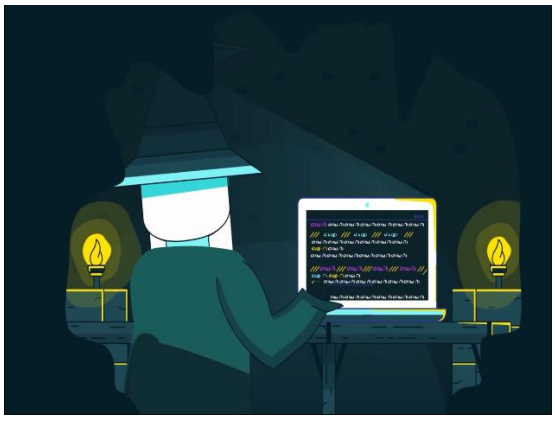
Part 2

1. What are Decentralized Applications?
   A Dapp, or decentralized application, is a software application that runs on a distributed network. It's not hosted on a centralized server, but instead on a peer-to-peer decentralized network.

2. Clear explanation of this concept

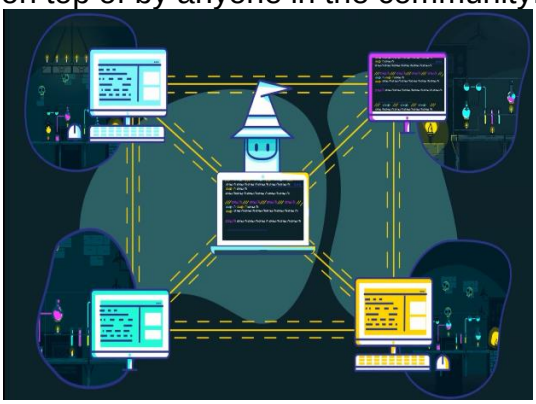


Dapps is the abbreviated term for decentralized application. just as any developer can build apps for the App Store on Apple's iOS operating system,developers can also build on top of Ethereum blockchain infrastructure.(Ethereum is the community-run technology powering the cryptocurrency).To the end-user a dapp might not look and feel any different than other apps you use today.However dapps are powered by the blockchain and this makes them different and perhaps far superior.Here's what you need to know ,a dapps front-end code and user interface can be written in any language that can make calls to its back-end.

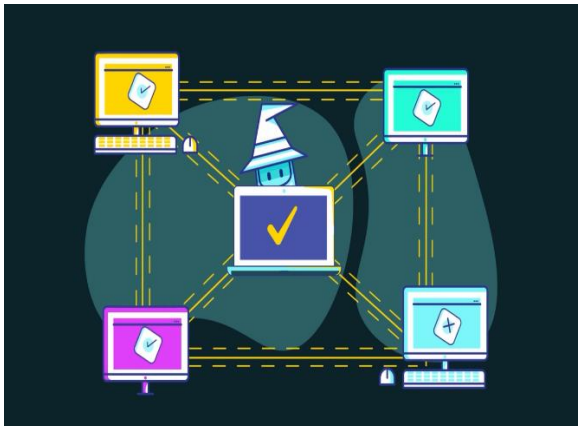


Its back-end code runs on a decentralized peer-to-peer network like ethereal and all records of the applications operations are stored on a blockchain.In most cases the

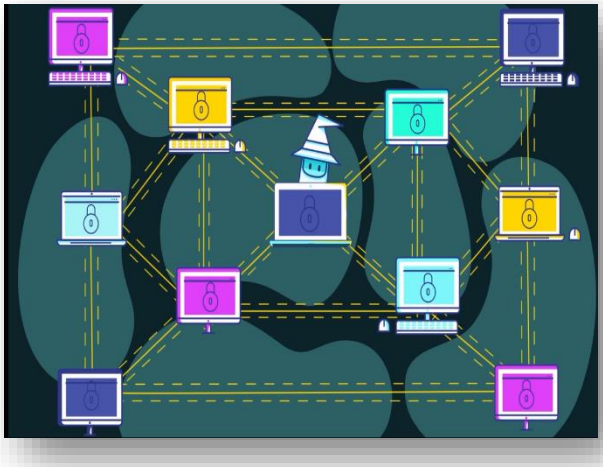
entire code base is open source, this means other people can access the code and build on prime of it.However nobody owns the application ,meaning they are free to be used, improved and built on top of by anyone in the community.



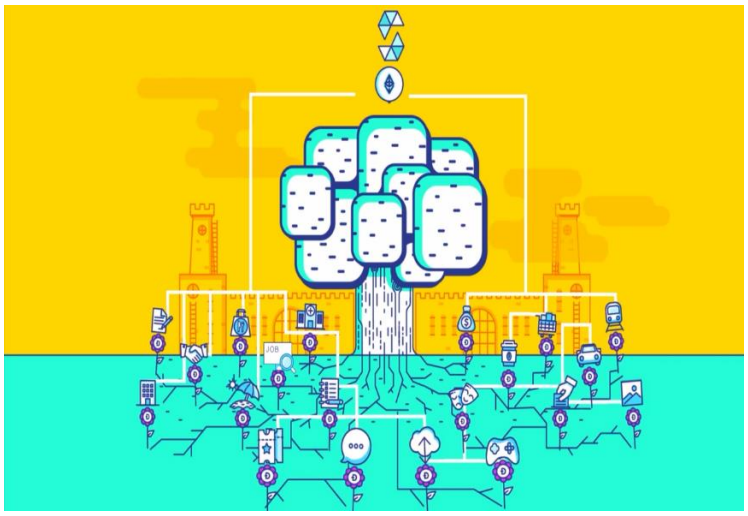
Finally the collection of users of the dapps are free to govern themselves. This concept is called autonomy.



Running dapps on the blockchain also offers added security benefits.Since the transactions are distributed and encrypted across the Ethereum blockchain a hacker has no primary location to breach and gain access.



The Ethereum is perhaps the best platform for building dapps thanks to its very own language solidity.This enables developers to form smart contracts using the Ethereum virtual machine.Using these tools developers have created dapps with use cases ranging from prediction markets to resource planning.



One example of a Dapp already up and running is 'Ethlance'(https://ethlance.com/). 'Ethlance' is a completely decentralized job market where by freelancers can find work and employers can find workers.Since it is powered by the Ethereum blockchain apart from gas fees it is completely free to use.

3. Build a suitable case scenario to explain this concept.

Car Renting Dapp

You must present your driver's license and complete certain paperwork while renting a car.You paid for the automobile and received the key.The procedure of renting an automobile is, in truth, rather difficult.It's simply that automobile manufacturers make leasing appear to be so simple.The primary issue that automobile leasing firms confront is that data is not kept up to date.

Everyone in the vehicle rental supply chain may use blockchain to monitor, share, analyze, and update the most up-to-date information.Best of all, regardless of where the automobile is in its lifespan, you get the most up-to-date information.

The vehicle rental industry as a whole benefited from faster processing times, more reliable data, and cheaper overhead costs because to blockchain.

1.Discover the Service

How would you go about finding an automobile to hire, say, at the Bandaranayake Airport?Take out your phone and use the location filter to search vehicles near you on Uber,Pickme.In terms of the blockchain technology, we'll also require a (global) discovery service to keep track of the automobiles' whereabouts.As a result, a smart contract for managing automobile registration and maintaining linkages to the car profile on decentralized storage, such as pricing, location, and photographs, is created.

2.Customer Service

What if I require assistance on the road or have a disagreement with the renter? Is there someone I can contact on a decentralized platform to help me fix this issue?On the surface, it appears impossible.
However, the “DAO,” or Decentralized Autonomous Organization, is well-known. We could create a customer service DAO and integrate it into the blockchain-based customer service market.

3.Payment

Payment is made on the blockchain using tokens or current crypto currencies like ether and bitcoin.

4.Insurance

On the blockchain, there are already several insurance use cases.A decentralized car-sharing software that supports blockchain-based plug-and-play

insurance can reduce premiums while also ensuring quick payment in the event of an accident.
To report an accident, you'll either need a DAO entity or sensors on the automobile to automate the report and damage assessment.