Day 3

Exploitation Analyst

Hacking the SSL Network protocol:

First understand what is SSL Pinning?

SSL pinning is the process of associating a host (like an API server) with a specific SSL/TLS certificate or public key. Instead of trusting any certificate that is signed by a trusted Certificate Authority (CA), the app only accepts a specific certificate (or public key). This adds an extra layer of security.

Why Use SSL Pinning?

By default, HTTPS trusts any certificate issued by a trusted CA. If an attacker can:

- Install a malicious root certificate (on the device or network),
- Or intercept traffic using a forged certificate,

...they could decrypt and modify the data in transit.

SSL pinning stops this by rejecting all certificates except the one(s) you trust.

SSL Pinning bypass by overwriting packaged CA certificate with custom CA certificate:

Some mobile applications **bundle their own CA certificates** for SSL pinning (instead of using the system trust store). This helps them verify the server’s identity and block MITM attacks — but it can be bypassed if the cert is replaced.

Bypass Concept

If the app uses a custom CA certificate (typically found in the assets or res/raw directory of the APK), you can:

1. Extract the APK (e.g., using APK Studio or apktool).
2. Locate the bundled CA certificate inside directories like /assets/.
3. Replace it with your own custom CA certificate (generated with tools like Burp Suite or mitmproxy).
4. Repack and sign the APK, then install it on the device.

Result

The app will now trust your MITM proxy’s certificate, allowing you to intercept HTTPS traffic without triggering SSL pinning errors.

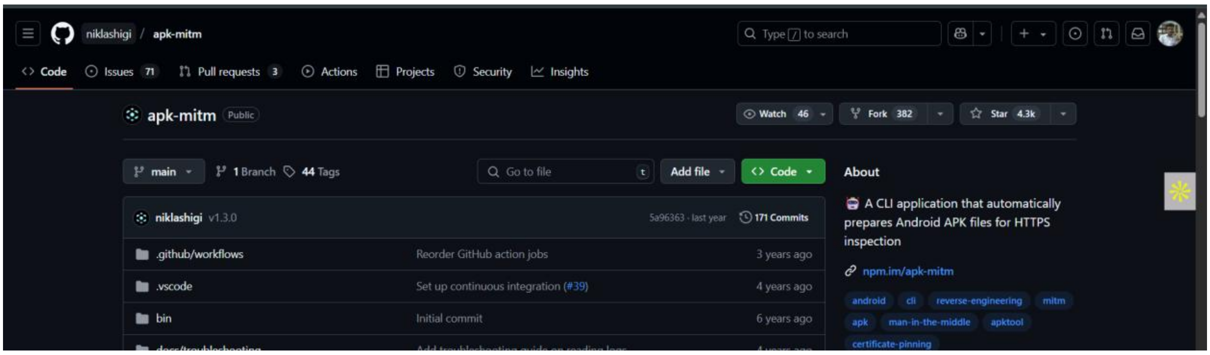
Tools Used

- apk-mitm: Automates certificate injection and patching of common pinning libraries.
- apktool/APK Studio: For manual unpacking and certificate replacement.

Bypassing SSL pinning using the tool apk-mitm:

https://youtu.be/odGnlw4MZx0?si=glwPNCqeFDP22PRv

First visit the github repo of this: https://github.com/niklashigi/apk-mitm



niklashigi v1.3.0 5a96363 · last year 171 Commits

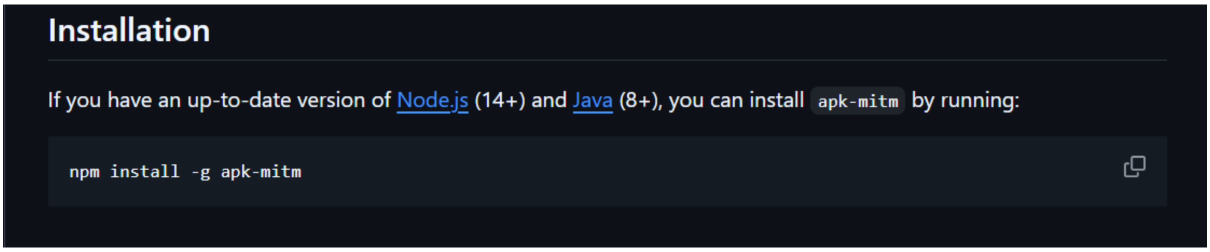
.github/workflows	Reorder GitHub action jobs	3 years ago
.vscode	Set up continuous integration (#39)	4 years ago
bin	Initial commit	6 years ago
docs/troubleshooting	Add troubleshooting guide on reading logs	4 years ago

A CLI application that automatically prepares Android APK files for HTTPS inspection

npm.im/apk-mitm

android cli reverse-engineering mitm apk man-in-the-middle apktool certificate-pinning

Install the tool using the following command: (in terminal as shown next)

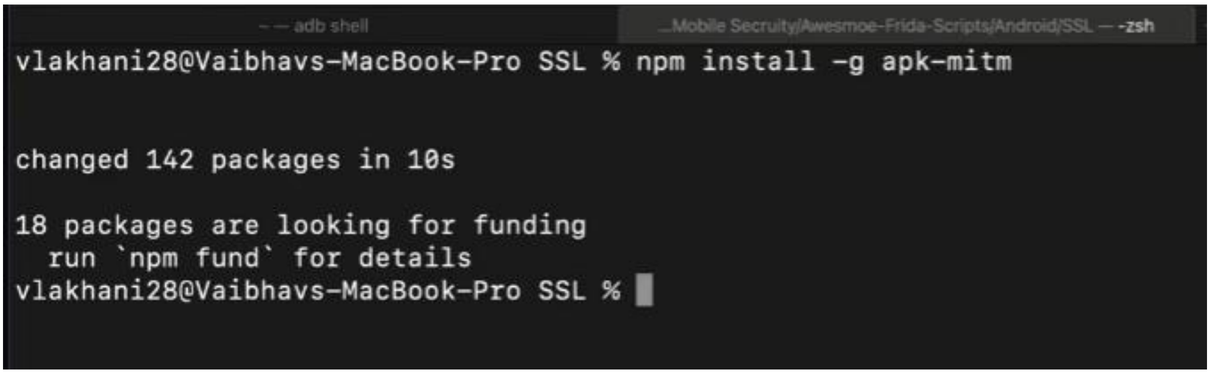


Installation

If you have an up-to-date version of Node.js (14+) and Java (8+), you can install `apk-mitm` by running:

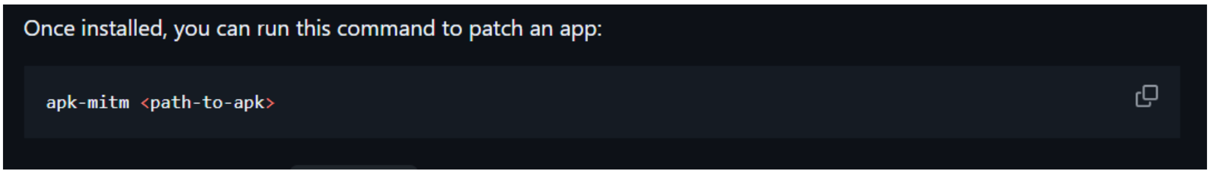
```
npm install -g apk-mitm
```

In terminal:



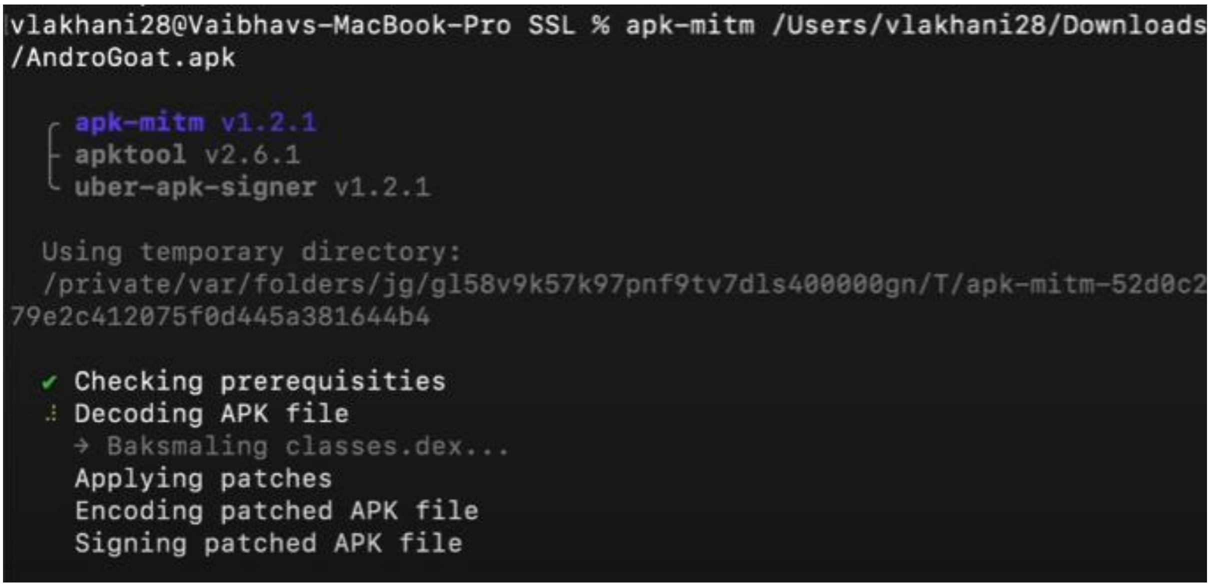
```
vlakhani28@Vaibhavs-MacBook-Pro SSL % npm install -g apk-mitm


changed 142 packages in 10s

18 packages are looking for funding
  run `npm fund` for details
vlakhani28@Vaibhavs-MacBook-Pro SSL % ▌
```

Now, you can run it on the .apk file using the following command: (shown next in terminal)



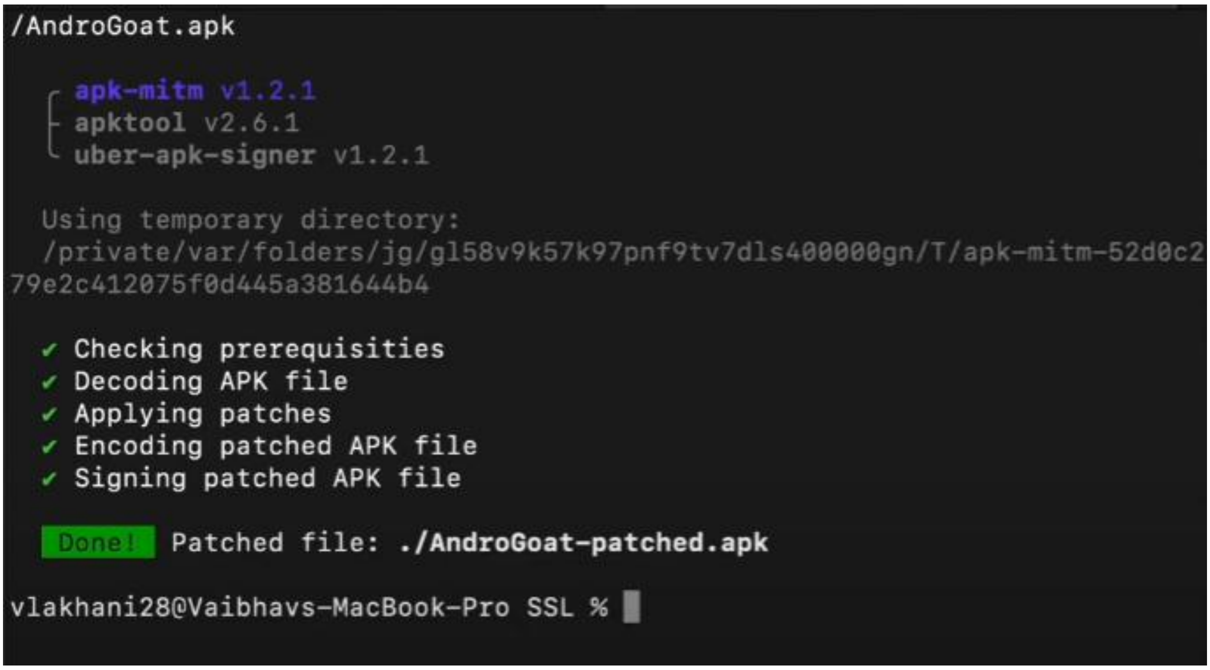
Once installed, you can run this command to patch an app:

```
apk-mitm <path-to-apk>
```



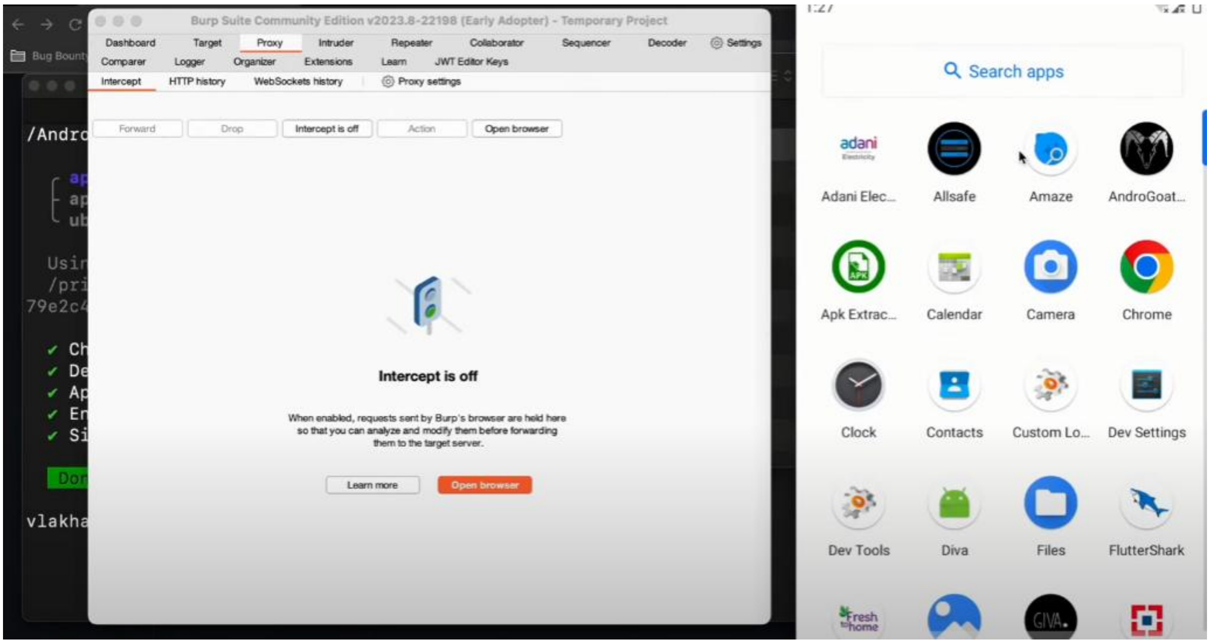
```
vlakhani28@Vaibhavs-MacBook-Pro SSL % apk-mitm /Users/vlakhani28/Downloads
/AndroGoat.apk

  ╭ apk-mitm v1.2.1
  ├ apktool v2.6.1
  ╰ uber-apk-signer v1.2.1

  Using temporary directory:
  /private/var/folders/jg/gl58v9k57k97pnf9tv7dls400000gn/T/apk-mitm-52d0c2
79e2c412075f0d445a381644b4

  ✔ Checking prerequisites
  : Decoding APK file
    → Baksmaling classes.dex...
    Applying patches
    Encoding patched APK file
    Signing patched APK file
```

Once, it is done, it will give us a patched .apk file:



```
/AndroGoat.apk

  ╭ apk-mitm v1.2.1
  ├ apktool v2.6.1
  ╰ uber-apk-signer v1.2.1

  Using temporary directory:
  /private/var/folders/jg/gl58v9k57k97pnf9tv7dls400000gn/T/apk-mitm-52d0c2
79e2c412075f0d445a381644b4

  ✔ Checking prerequisites
  ✔ Decoding APK file
  ✔ Applying patches
  ✔ Encoding patched APK file
  ✔ Signing patched APK file

  Done!  Patched file: ./AndroGoat-patched.apk

vlakhani28@Vaibhavs-MacBook-Pro SSL % █
```

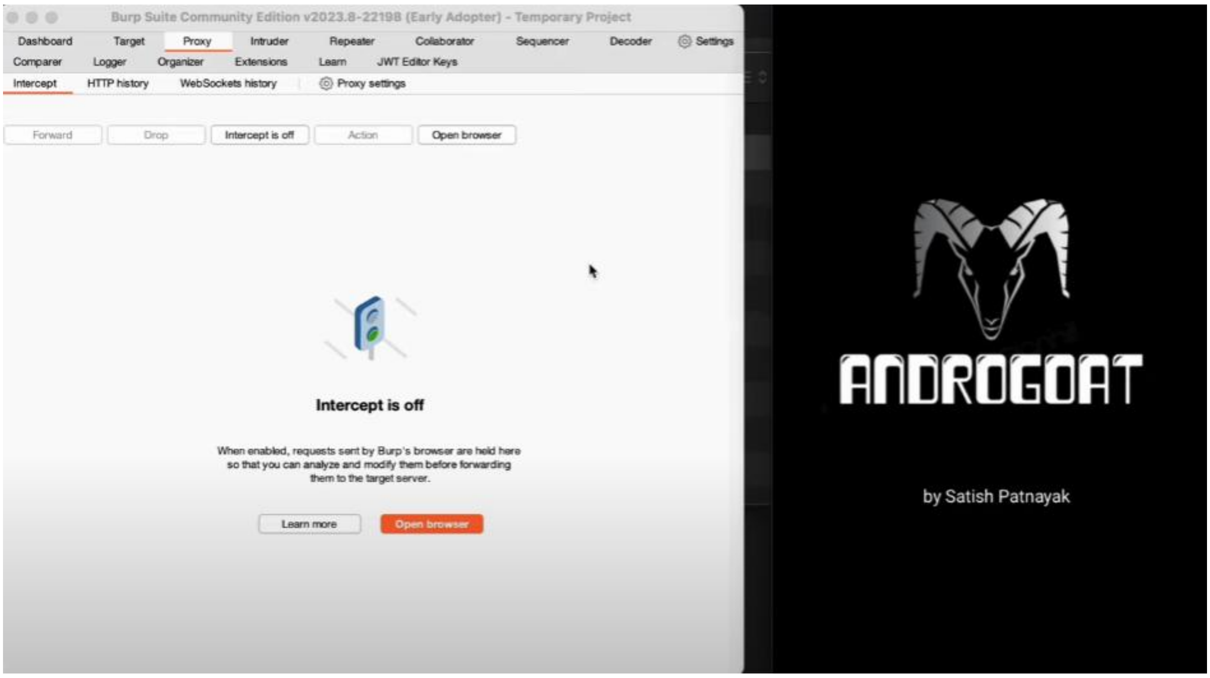
Now, we will try to un install the original .apk file from the android simulation. For this we will need the Burp Suite



Intercept is off

When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

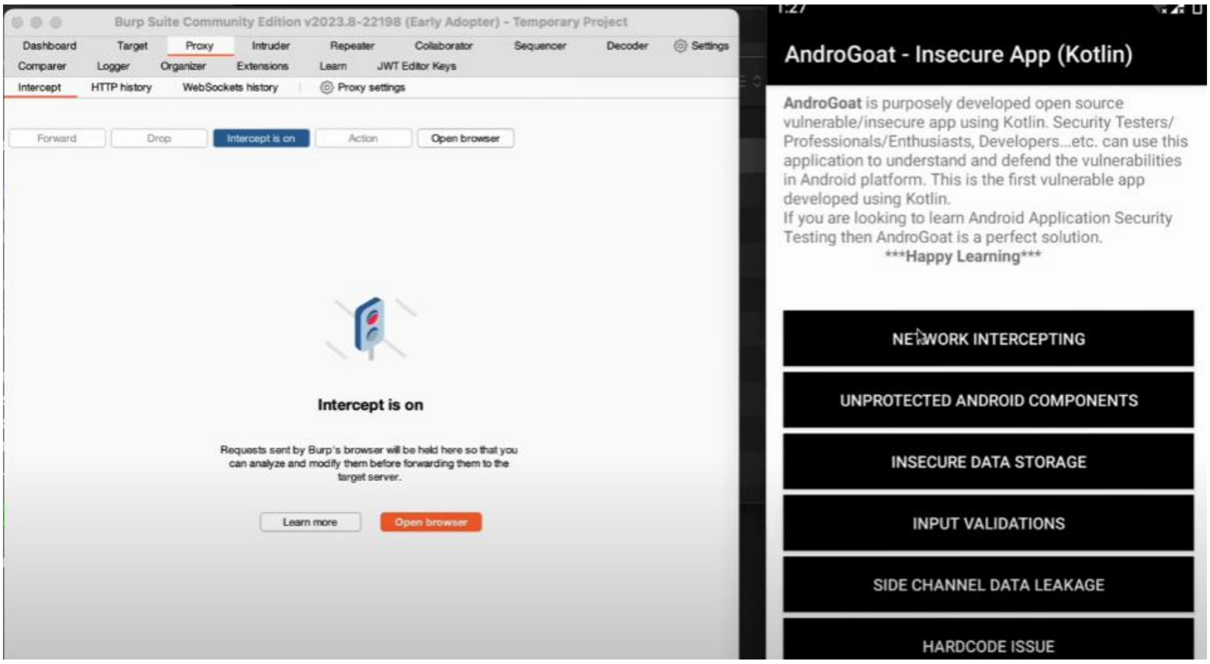
Here, open the Androgoat app:



Intercept is off

When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

Keep your intercept on, and then click on the "Networking intercept" option:



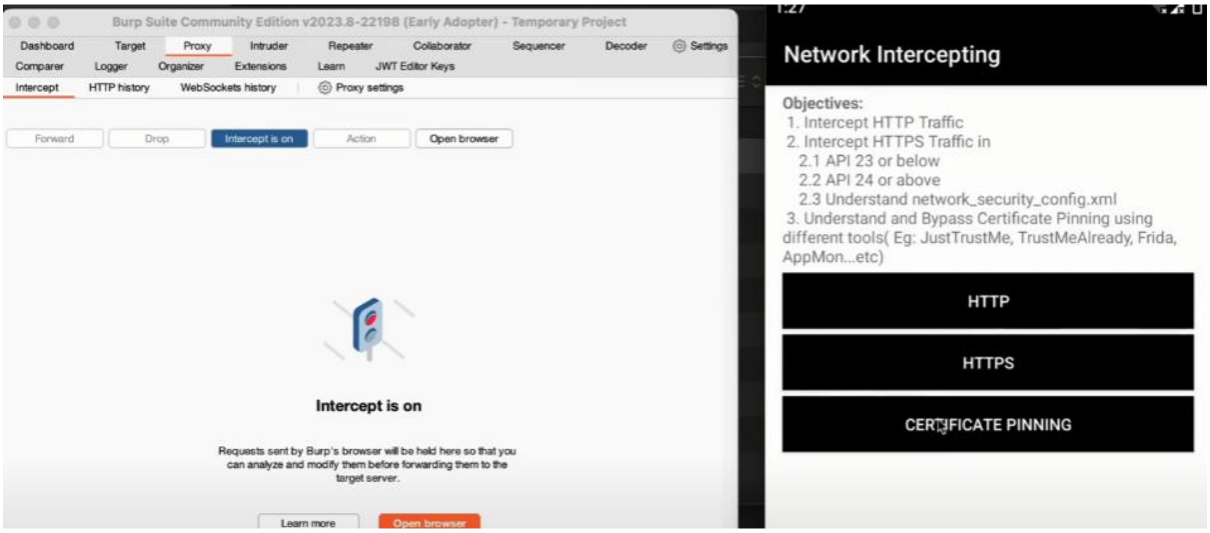
Intercept is on

Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the target server.

AndroGoat - Insecure App (Kotlin)

AndroGoat is purposely developed open source vulnerable/insecure app using Kotlin. Security Testers/ Professionals/Enthusiasts, Developers...etc. can use this application to understand and defend the vulnerabilities in Android platform. This is the first vulnerable app developed using Kotlin.
If you are looking to learn Android Application Security Testing then AndroGoat is a perfect solution.
Happy Learning

Following screen will appear:



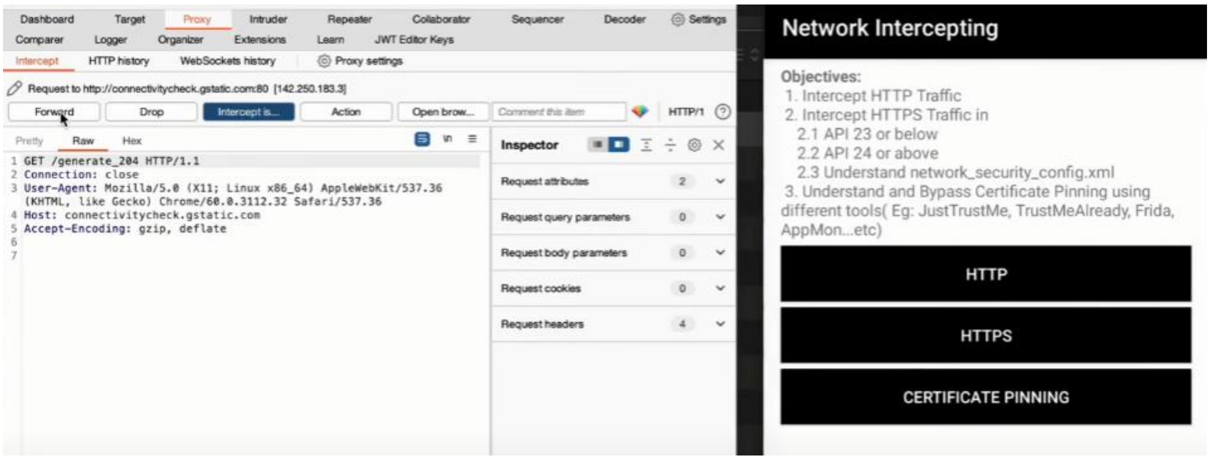
Intercept is on

Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the target server.

Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
2.1 API 23 or below
2.2 API 24 or above
2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

Click on the "Certificate pinning", and then check if intercept is captured or not using the Burp Suite. As, we see below we are not able to intercept that:



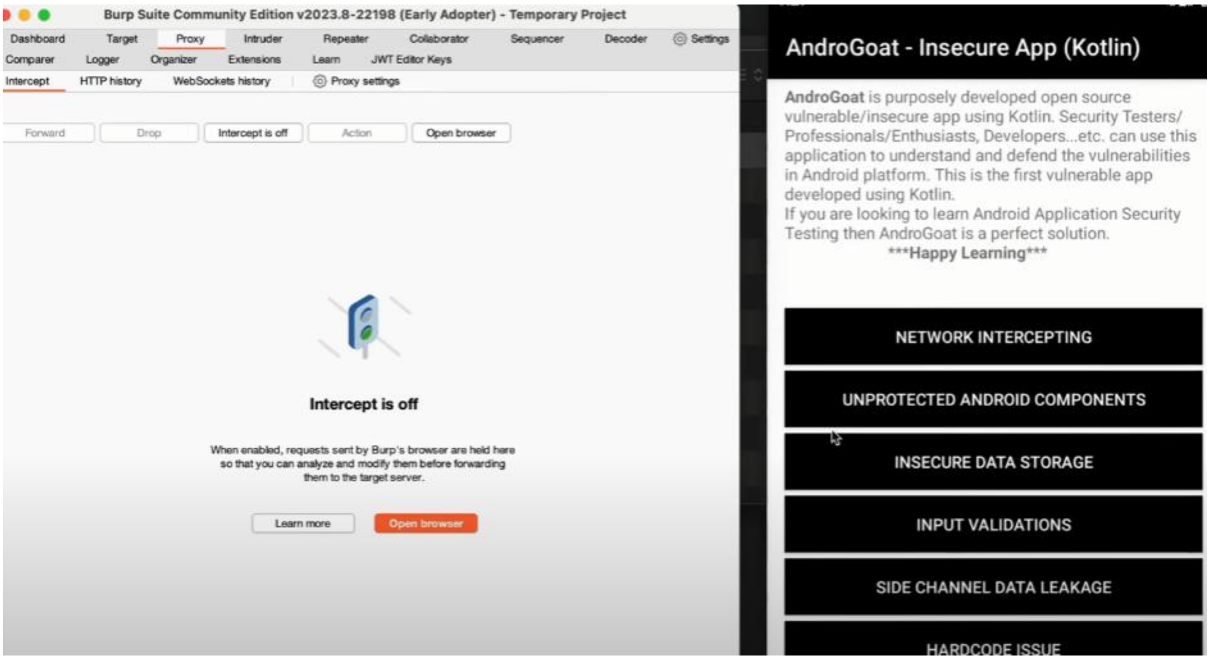
Request to http://connectivitycheck.gstatic.com:80 [142.250.183.3]

```
GET /generate_204 HTTP/1.1
Connection: close
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/60.0.3112.32 Safari/537.36
Host: connectivitycheck.gstatic.com
Accept-Encoding: gzip, deflate
```

Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
2.1 API 23 or below
2.2 API 24 or above
2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

So, now, we can uninstall this old .apk file and install the new, manually:

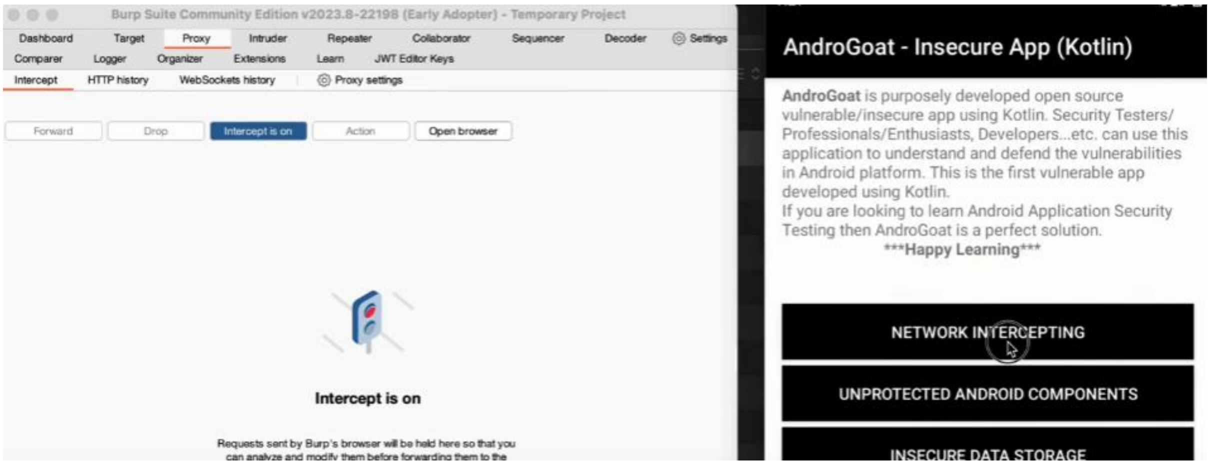


When enabled, requests sent by Burp's browser are held here so that you can analyze and modify them before forwarding them to the target server.

AndroGoat - Insecure App (Kotlin)

AndroGoat is purposely developed open source vulnerable/insecure app using Kotlin. Security Testers/ Professionals/Enthusiasts, Developers...etc. can use this application to understand and defend the vulnerabilities in Android platform. This is the first vulnerable app developed using Kotlin.
If you are looking to learn Android Application Security Testing then AndroGoat is a perfect solution.
Happy Learning

Keep intercept on and then click on the network intercepting option:

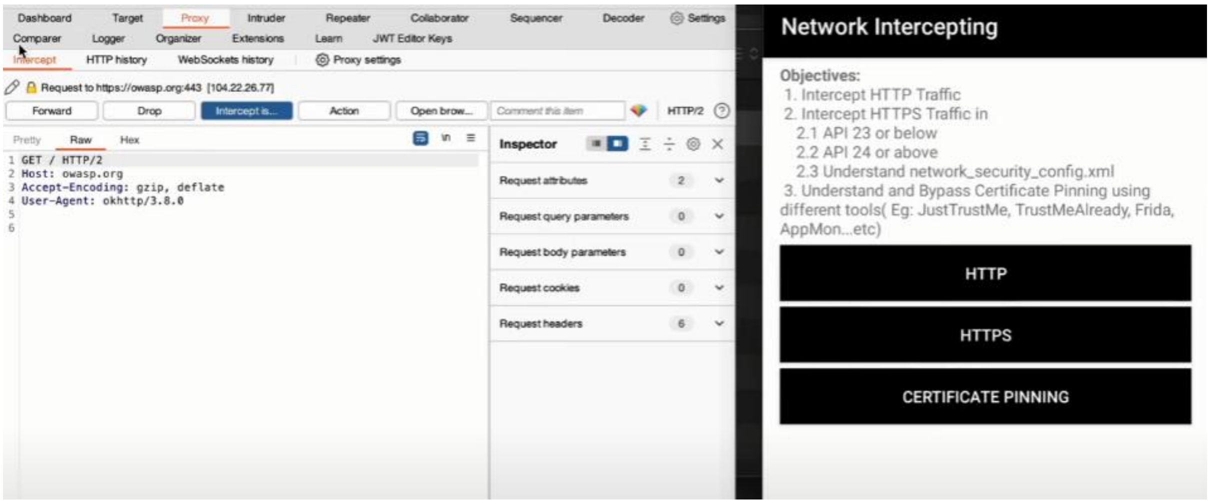


Requests sent by Burp's browser will be held here so that you can analyze and modify them before forwarding them to the

AndroGoat - Insecure App (Kotlin)

AndroGoat is purposely developed open source vulnerable/insecure app using Kotlin. Security Testers/ Professionals/Enthusiasts, Developers...etc. can use this application to understand and defend the vulnerabilities in Android platform. This is the first vulnerable app developed using Kotlin.
If you are looking to learn Android Application Security Testing then AndroGoat is a perfect solution.
Happy Learning

Then, click on the certificate pinning option as shown: Clearly, intercept get captured and thus we managed to by pass the SSL pinning.



Request to https://owasp.org:443 [104.22.26.77]

```
GET / HTTP/2
Host: owasp.org
Accept-Encoding: gzip, deflate
User-Agent: okhttp/3.8.0
```

Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 0
Request headers 6

Network Intercepting

Objectives:
1. Intercept HTTP Traffic
2. Intercept HTTPS Traffic in
2.1 API 23 or below
2.2 API 24 or above
2.3 Understand network_security_config.xml
3. Understand and Bypass Certificate Pinning using different tools(Eg: JustTrustMe, TrustMeAlready, Frida, AppMon...etc)

--The End--