FIELD NOTE / ANDROID

# Easy Way to Bypass SSL Pinning with Objection & Frida [Beginner Friendly]

Petruknisme · 29 Aug, 2022 · 5 min to read

After creating an article "Let's Bypass SSL Pinning By Manual Modification (https://petruknisme.medium.com/lets-bypass-ssl-pinning-by-manual-modification-617afc08a079)", I've thought of making an easy way for beginner in learning how to bypass ssl pinning with **uncomplicated targets** and using automatic method. In this article I will talk about objection for bypassing ssl pinning.

**Objection** is a runtime mobile exploration toolkit, powered by Frida (https://www.frida.re/), built to help you assess the security posture of your mobile applications, without needing a jailbreak.

- Supports both iOS and Android.
- Inspect and interact with container file systems.
- Bypass SSL pinning.
- Dump keychains.
- Perform memory related tasks, such as dumping & patching.
- Explore and manipulate objects on the heap.
- And much, much more (https://github.com/sensepost/objection/wiki/Features) ...

We need to install and setup Frida because objection uses Frida as the instrumentation. For more information about Frida, you can read at ***https://frida.re*** (https://frida.re/) **/**

**Install frida**

I'm using virtualenv in MacOS, but you can follow as your environment. Install frida-tools from pip

```
❯ pip -V
pip 22.1.2 from /Users/petruknisme/.venv/lib/python3.9/site-packages/pip (
❯ python -V
Python 3.9.13❯ pip install frida-tools
Collecting frida-tools
  Using cached frida_tools-11.0.0-py3-none-any.whl
Requirement already satisfied: prompt-toolkit<4.0.0,>=2.0.0 in /Users/petr
Requirement already satisfied: colorama<1.0.0,>=0.2.7 in /Users/petruknism
Collecting frida<16.0.0,>=15.2.0
  Using cached frida-15.2.2-cp39-cp39-macosx_10_12_x86_64.whl
```

Licensed under CC BY-SA 4.0

```
Requirement already satisfied: pygments<3.0.0,>=2.0.2 in /Users/petruknism
Requirement already satisfied: setuptools in /Users/petruknisme/.venv/lib/
Requirement already satisfied: wcwidth in /Users/petruknisme/.venv/lib/pyt
Installing collected packages: frida, frida-tools
Successfully installed frida-15.2.2 frida-tools-11.0.0
```

We've successfully installed frida- **15.2.2** and frida-tools-11.0.0. For frida to work in our android/ios, we need to setup the frida-server.

**Setup frida-server in Android**

Go to https://github.com/frida/frida/releases (https://github.com/frida/frida/releases) and choose the frida-server-version-android-arm.xz (https://github.com/frida/frida/releases/download/15.2.2/frida-server-15.2.2-android-arm.xz), change the version with your installed frida version like I bold above. Frida client and server must have the **same version**.

Download the frida to host machine

```
❯ wget https://github.com/frida/frida/releases/download/15.2.2/frida-serve

Length: 6824388 (6.5M) [application/octet-stream]
Saving to: 'frida-server-15.2.2-android-arm64.xz'frida-server-15.2.2-andr
```

Extract xz file

```
❯ xz -d frida-server-15.2.2-android-arm64.xz
❯ ls | grep frida
frida-server-15.2.2-android-arm64
❯ mv frida-server-15.2.2-android-arm64 frida-server
```

After downloading, rename the file to frida-server, the next step is to copy the file to android. In my device, I copy to /data/local/tmp/

```
❯ adb push frida-server /data/local/tmp
```

```
frida-server: 1 file pushed, 0 skipped. 23.1 MB/s (19972988 bytes in 0.824
```

Run the frida-server

```
❯ adb shell
jasmine_sprout:/ $ su
jasmine_sprout:/ # cd /data/local/tmp/
jasmine_sprout:/data/local/tmp # chmod +x frida-server
jasmine_sprout:/data/local/tmp # ./frida-server &
```

If running sucessfully, we can check with ps and grep

```
1|jasmine_sprout:/data/local/tmp # ps | grep frida
root          23299 23280   39280  19640 poll_schedule_timeout f10486e0 S f
```

We've done with setuping frida in android. Now, we will setup the objection.

**Install Objection**

```
❯ pip install -U objection
Collecting objection
  Downloading objection-1.11.0.tar.gz (327 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 327.2/327.2 kB
Successfully built objection
Installing collected packages: objection
Successfully installed objection-1.11.0
```

Testing if Objection is working properly with frida-server from our android

```
❯ objection --gadget "com.android.settings" device-type
```

Licensed under CC BY-SA 4.0

```
Using USB device `Mi A2`
Agent injected and responds ok!
Connection: USB
Name: com.android.settings
System: jasmine_sprout
Model: xiaomi
Version: 10
Asking jobs to stop...
Unloading objection agent...
```

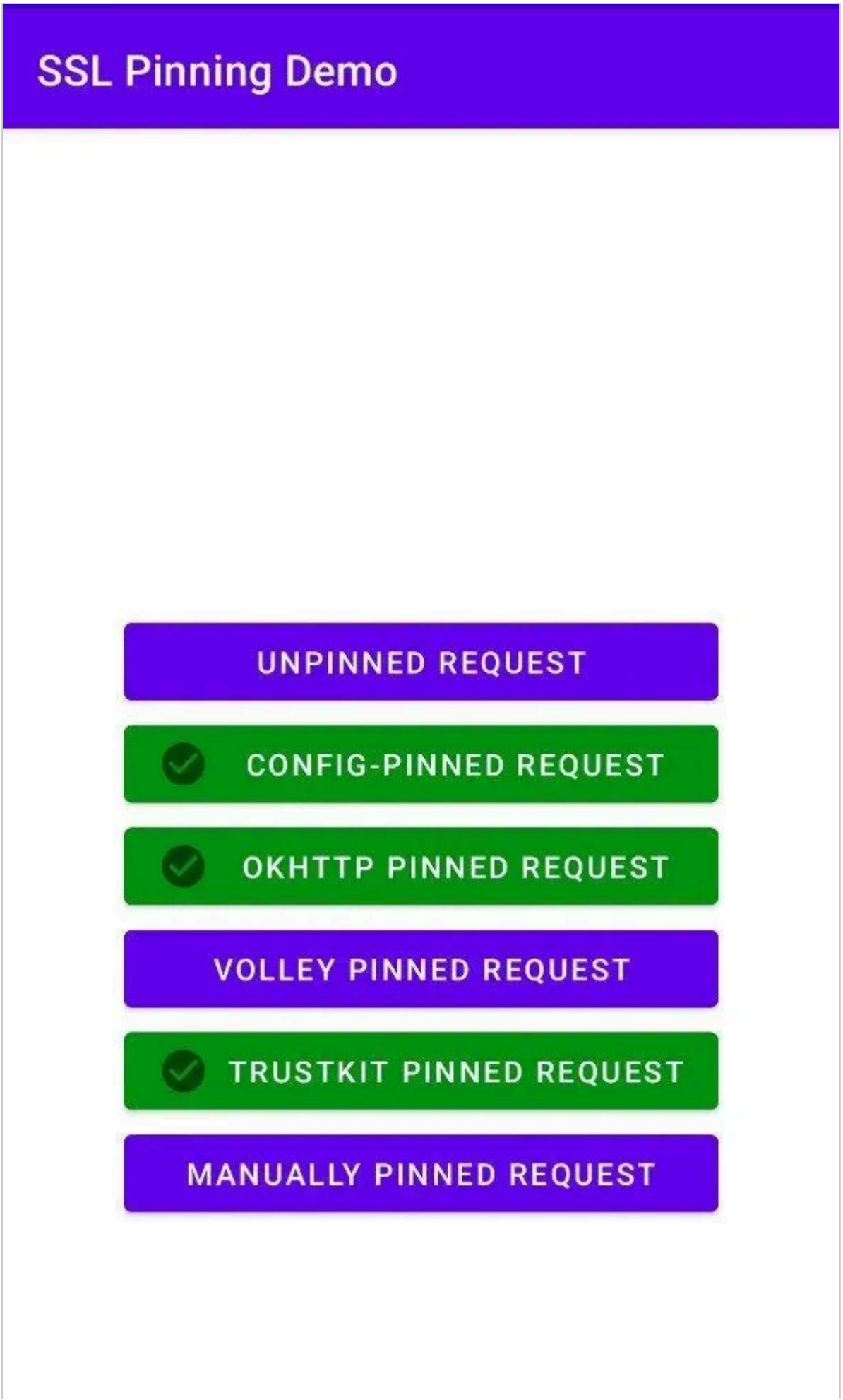
From the above message, it tells us that it's successfully injected the agent and showing device information.

In this article, we will usea sample application to learn how objection bypass the ssl pinning. Download and install the application from https://github.com/aancw/android-ssl-pinning-signed-demo (https://github.com/aancw/android-ssl-pinning-signed-demo)

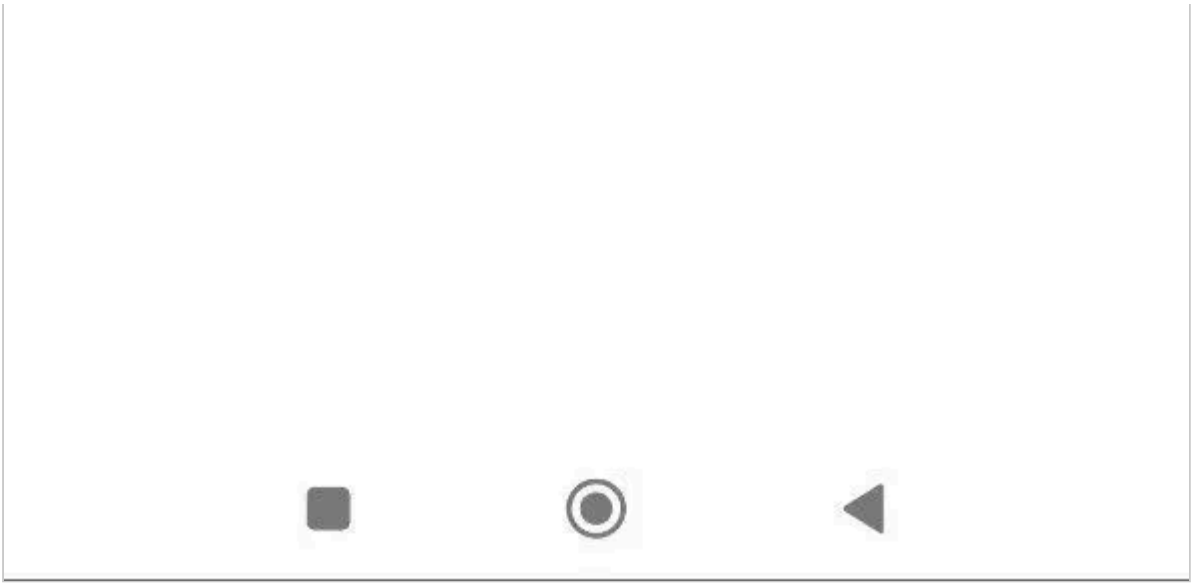
```
❯ wget https://github.com/aancw/android-ssl-pinning-signed-demo/releases/d
Performing Streamed Install
Success
```

When we open the application, the button color will be purple. But when we click the button, it will change the color to red(fail)/green(success) when doing https request to the pinned server.

Licensed under CC BY-SA 4.0



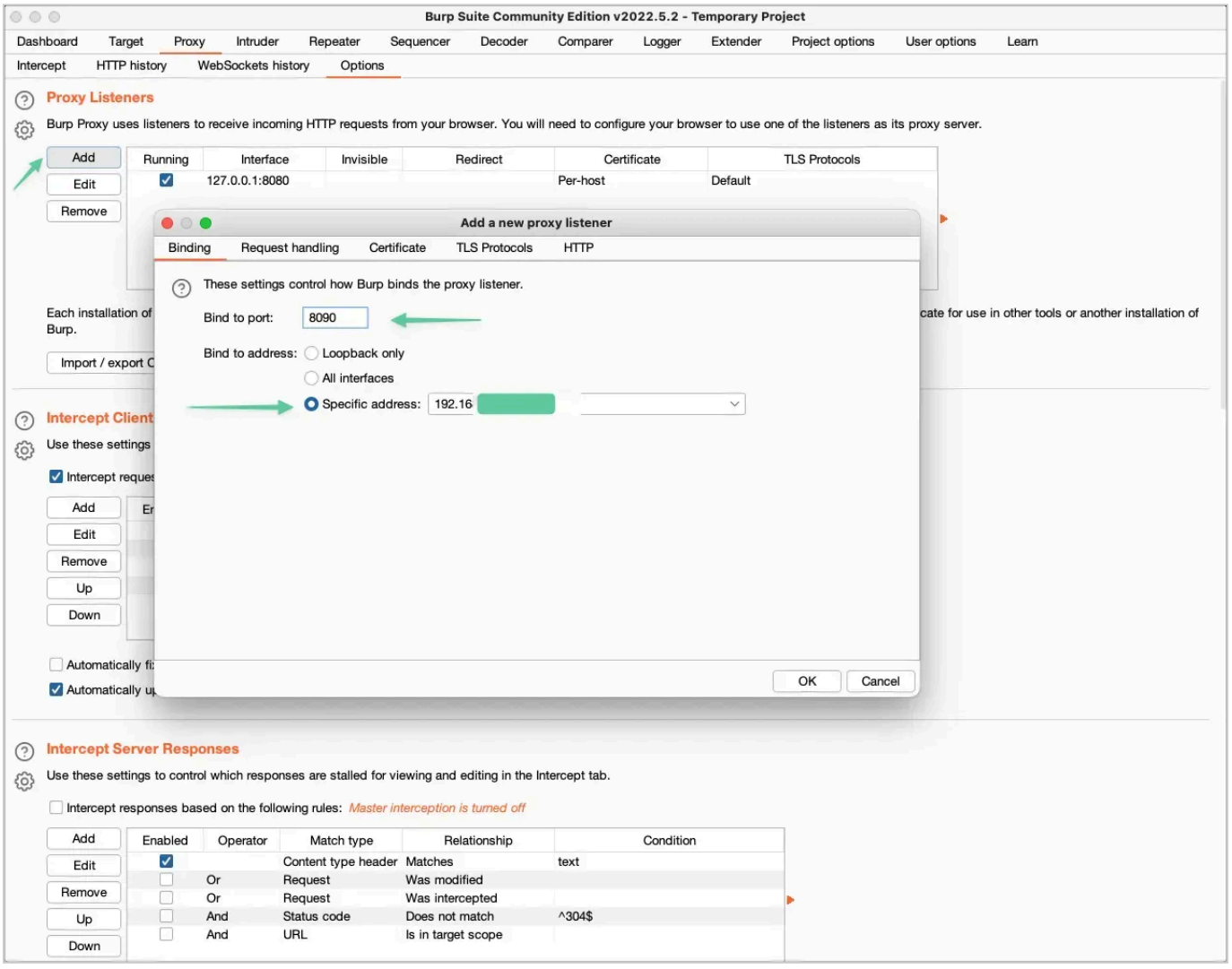

Licensed under CC BY-SA 4.0 



SSL Pinning Demo app

### Proxy Setup

We need to test the app using a proxy to know the app is fail or success when doing http reque st. Run burp suite app and se t listen address




Licensed under CC BY-SA 4.0

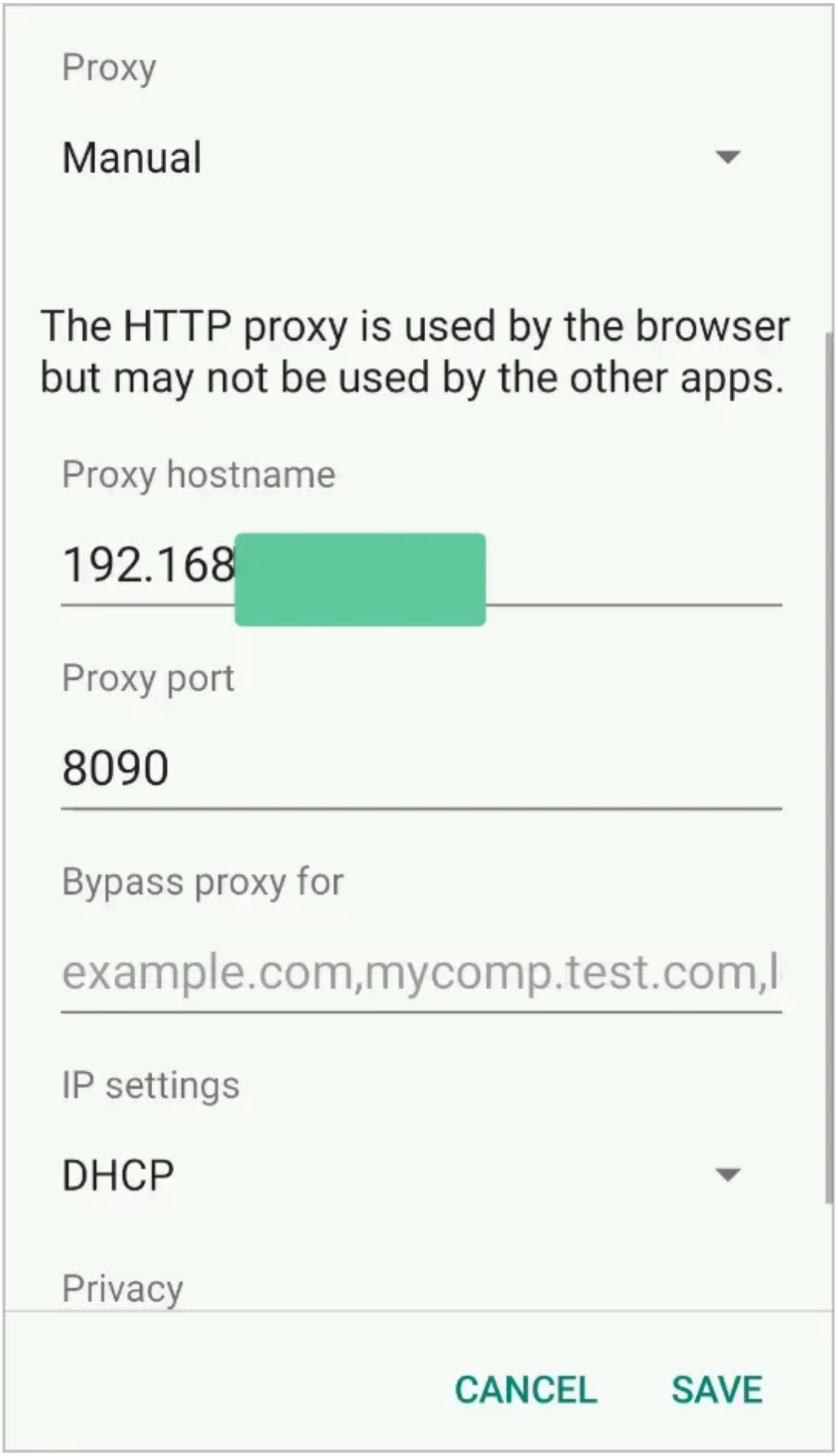
Setting Burp Proxy

Explanation:

- Setting bind port to 8090, this port will be used in android setting
- Setting listen address to our LAN IP

Setting the android wifi with manual proxy setting

Licensed under CC BY-SA 4.0 

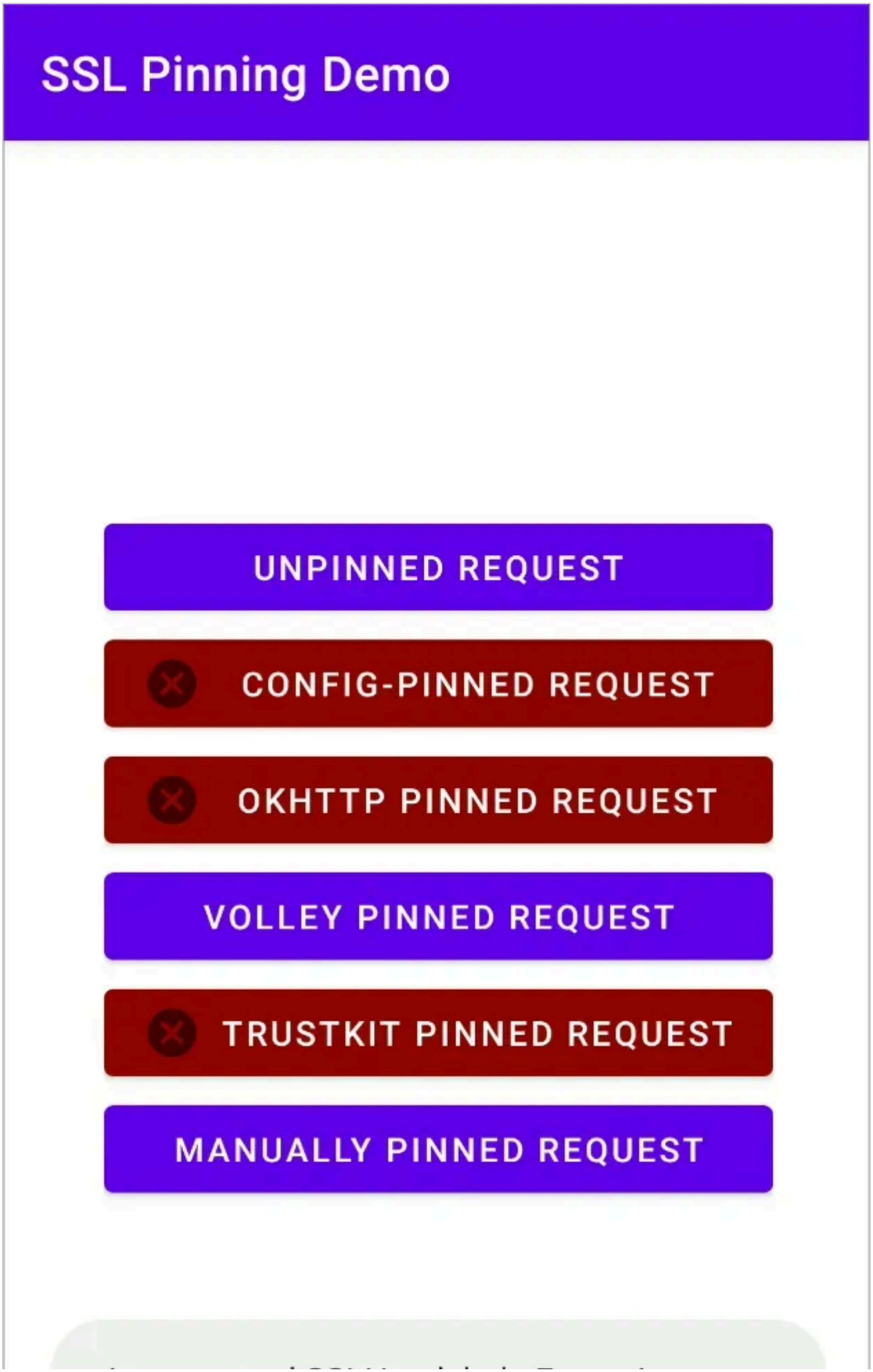



Android proxy setting

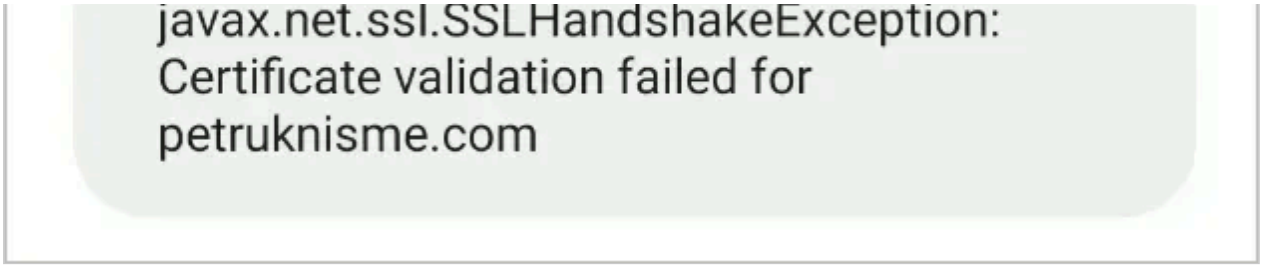
Licensed under CC BY-SA 4.0

Click save and run the application again. When we click the button, it will be showing an error message telling us the pinning process failed

Licensed under CC BY-SA 4.0 




Licensed under CC BY-SA 4.0 




Failed Pinning using Proxy

The main topic for this article is bypassing ssl pinning with different pinning method/library like okhttp, trustkit, volley and much more.

**Bypass SSL Pinning**

Check the target running app package name

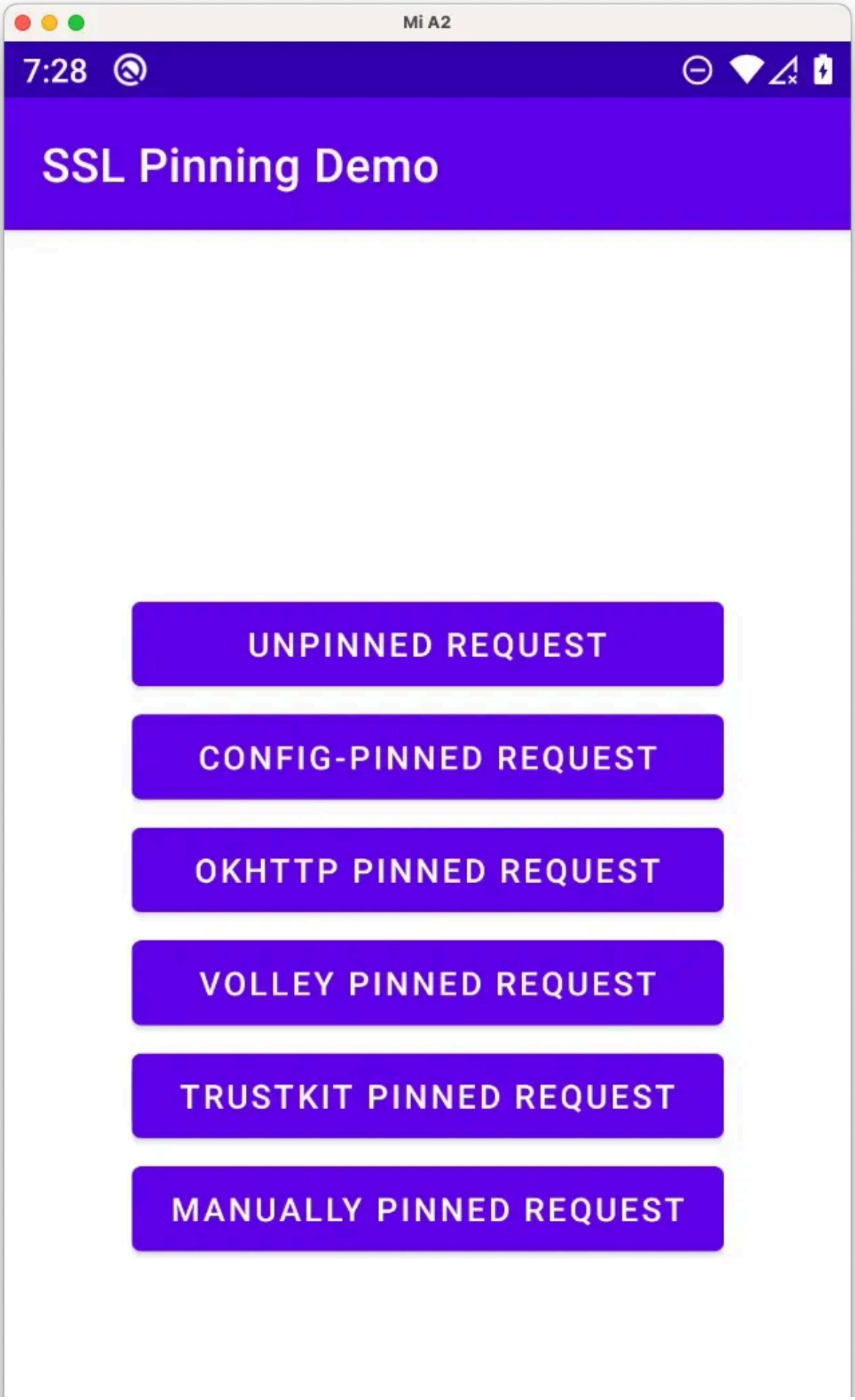
```
❯ frida-ps -Uia | grep pinning
21332  SSL Pinning Demo     tech.httptoolkit.pinning_demo
```

Running objection and injecting the agent. Our target application will be running again with the new session. When we click the button, it will still be showing us an error message when doing pinning request.

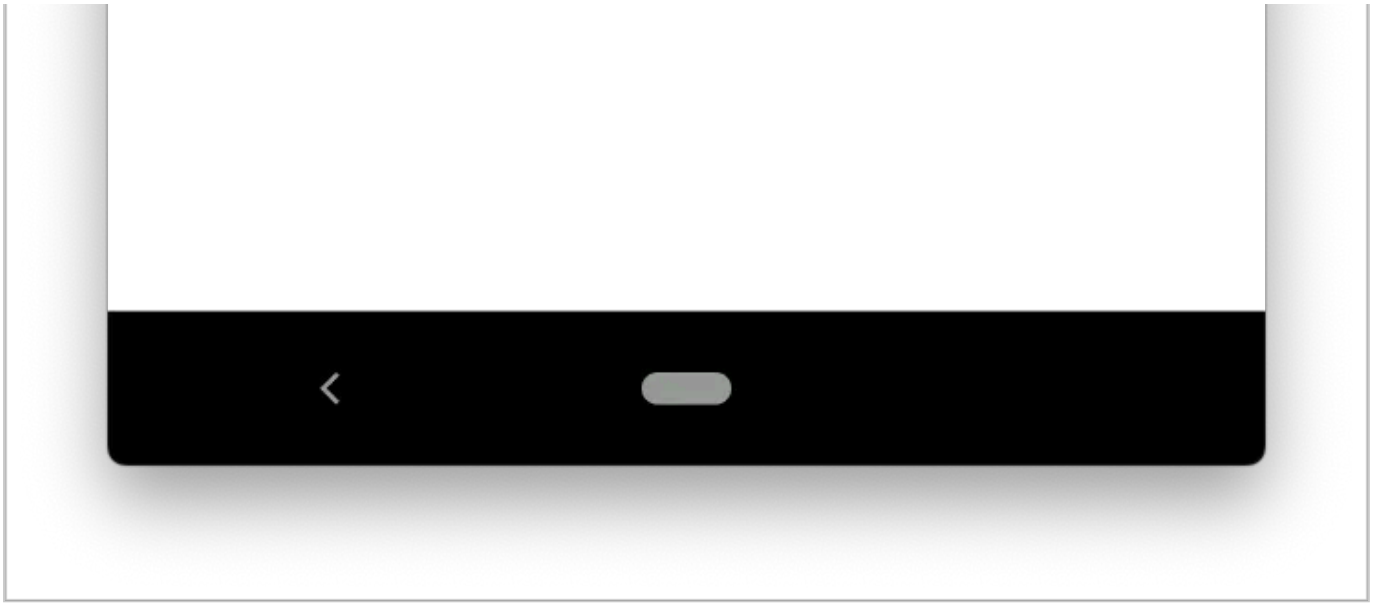
Objection

```
❯ objection --gadget "tech.httptoolkit.pinning_demo" explore
Using USB device `Mi A2`
Agent injected and responds ok!_   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | | -_|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.11.0Runtime Mobile Exploration
        by: @leonjza from @sensepost[tab] for command suggestions
tech.httptoolkit.pinning_demo on (xiaomi: 10) [usb] #
```

Licensed under CC BY-SA 4.0




Licensed under CC BY-SA 4.0 

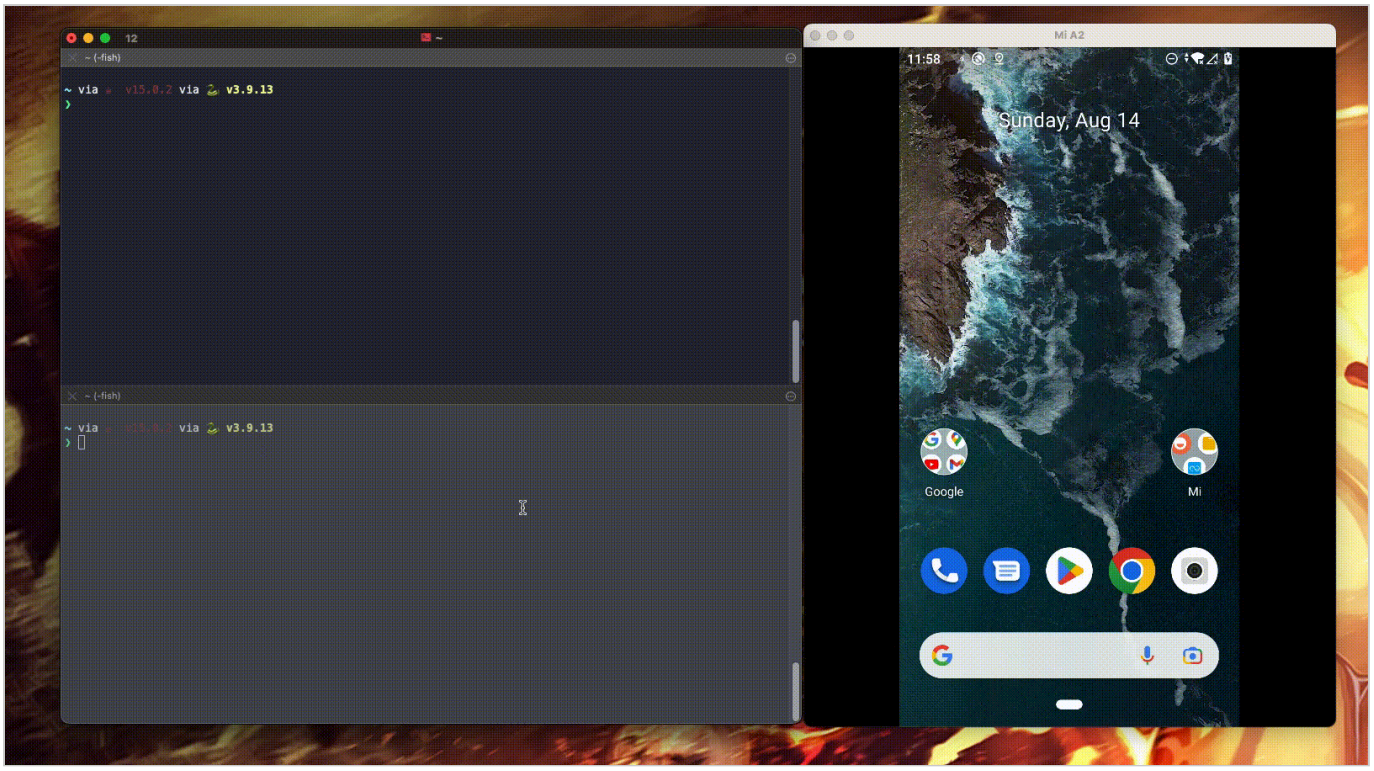


SSL Pinning Application running with objection

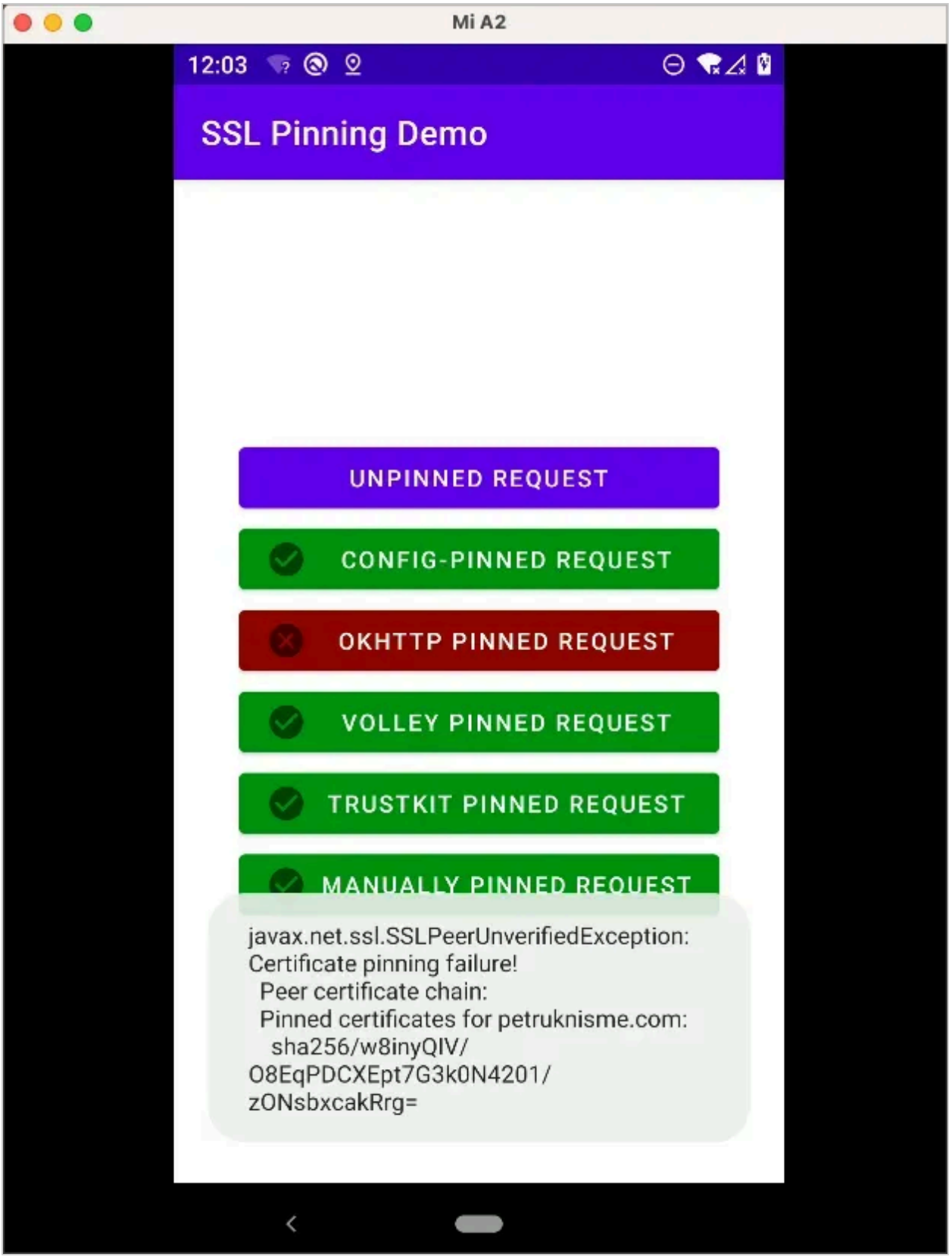
For doing ssl pinning bypass, we can use the command `android sslpinning disable`

```
tech.httptoolkit.pinning_demo on (xiaomi: 10) [usb] # android sslpinning d
(agent) Custom TrustManager ready, overriding SSLContext.init()
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.chec
(agent) Found okhttp3.CertificatePinner, overriding CertificatePinner.chec
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding Trust
(agent) Found com.android.org.conscrypt.TrustManagerImpl, overriding Trust
(agent) Registering job 751373. Type: android-sslpinning-disable
```

Licensed under CC BY-SA 4.0




Licensed under CC BY-SA 4.0 



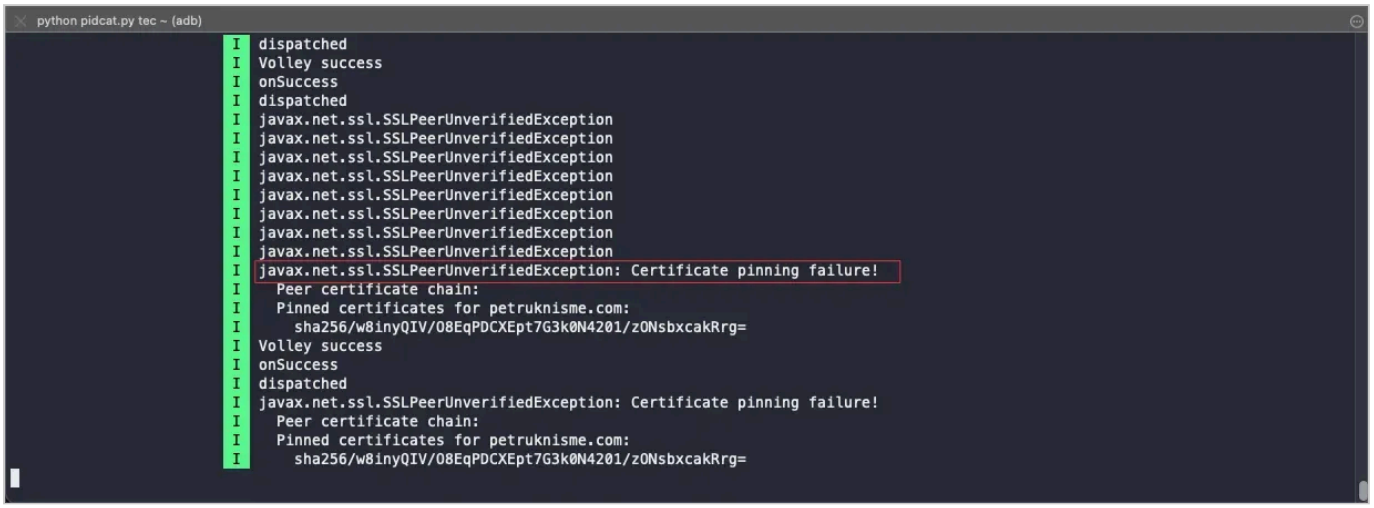

As you can see in the video above, we successfully bypassed the ssl pinning in our demo app. Only the Okhttp that still fail when bypassing ssl pinning using Objection.

As far as I know, the objection method can bypass ssl pinning for Okhttp. But for this target, it seems missing some function to hook. Let's start to dig into the problem.

Licensed under CC BY-SA 4.0

### Debugging with Frida

We will use Frida to debug our demo application to know what is causing objection so can't bypass the ssl pinning. From our last experiment above, it tells us that application throwing an error at `javax.net.ssl.SSLPeerUnverifiedException`.




Result from pidcat.py

Frida supports Javascript API (https://frida.re/docs/javascript-api/) for instrumenting the binary. So we can patch and modify the binary at runtime with javascript. In this case, we will create a javascript file to debug our application and try to bypass the ssl pinning that was missed by objection.

Because we are already spawned the application before, so we will use attach mode for frida to launch the script. More information about spawning vs attaching for frida: https://summit-labs.frida.ninja/frida-tool-reference/frida (https://summit-labs.frida.ninja/frida-tool-reference/frida)

Get PID of application that running with objection in previous section

```
❯ frida-ps -Uia | grep pinning_demo
8988  SSL Pinning Demo     tech.httptoolkit.pinning_demo
```

Our application PID is 8988. Note this number for later.

### Creating our Frida Javascript

Create an instance of javax.net.ssl.SSLPeerUnverifiedException and initialize it. This instance is from our previous error.

Licensed under CC BY-SA 4.0

```
Java.perform(function () {const UnverifiedCertError = Java.use('javax.net.
});
```

We will hook the constructor of a class using `$init` as the method name and hook Method in a class and set the implementation to the custom code.

```
UnverifiedCertError.$init.implementation = function (str) {
    };
```

In the above function, we will add java stack trace. So we know which function/method causing the previous error

```
UnverifiedCertError.$init.implementation = function (str) {const stackTrac
    };
```

After that we will filter the stack trace to only index the `javax.net.ssl.SSLPeerUnverifiedException`

```
UnverifiedCertError.$init.implementation = function (str) {const stackTrac
        const exceptionStackIndex = stackTrace.findIndex(stack =>
            stack.getClassName() === "javax.net.ssl.SSLPeerUnverifiedExcep
        );
        const callingFunctionStack = stackTrace[exceptionStackIndex + 1];
    };
```

Then, we will map the className and methodName that causing the problem

```
UnverifiedCertError.$init.implementation = function (str) {const stackTrac
        const exceptionStackIndex = stackTrace.findIndex(stack =>
            stack.getClassName() === "javax.net.ssl.SSLPeerUnverifiedExcep
        );
        const callingFunctionStack = stackTrace[exceptionStackIndex + 1];c
        const methodName = callingFunctionStack.getMethodName();
    };
```

Licensed under CC BY-SA 4.0

We are done for debugging the stack trace. The final code for debugging the stack trace:

Save the file as ssl-error.js or whatever you name it. Run frida with our javascript

```
❯ frida -U -l <js_file> --no-pause -p <PID_Number>
```

Frida will attach the script to our running process

Frida

```
____
 / _  |   Frida 15.1.17 - A world-class dynamic instrumentoolkit
| (_| |
 > _  |   Commands:
/_/ |_|       help      -> Displays the help system
. . . .       object?   -> Display information about 'object'
. . . .       exit/quit -> Exit
. . . .
. . . .   More info at https://frida.re/docs/home/[Mi A2::PID::8988]->
```

Switch to android ssl pinning demo application and then click Okhttp Pinning Request. This is what I get from frida

```
❯ frida -U -l ssl-error.js --no-pause -p 8988
    ____
   / _  |   Frida 15.1.17 - A world-class dynamic instrumentation toolkit
  | (_| |
   > _  |   Commands:
  /_/ |_|       help      -> Displays the help system
  . . . .       object?   -> Display information about 'object'
  . . . .       exit/quit -> Exit
  . . . .
  . . . .   More info at https://frida.re/docs/home/

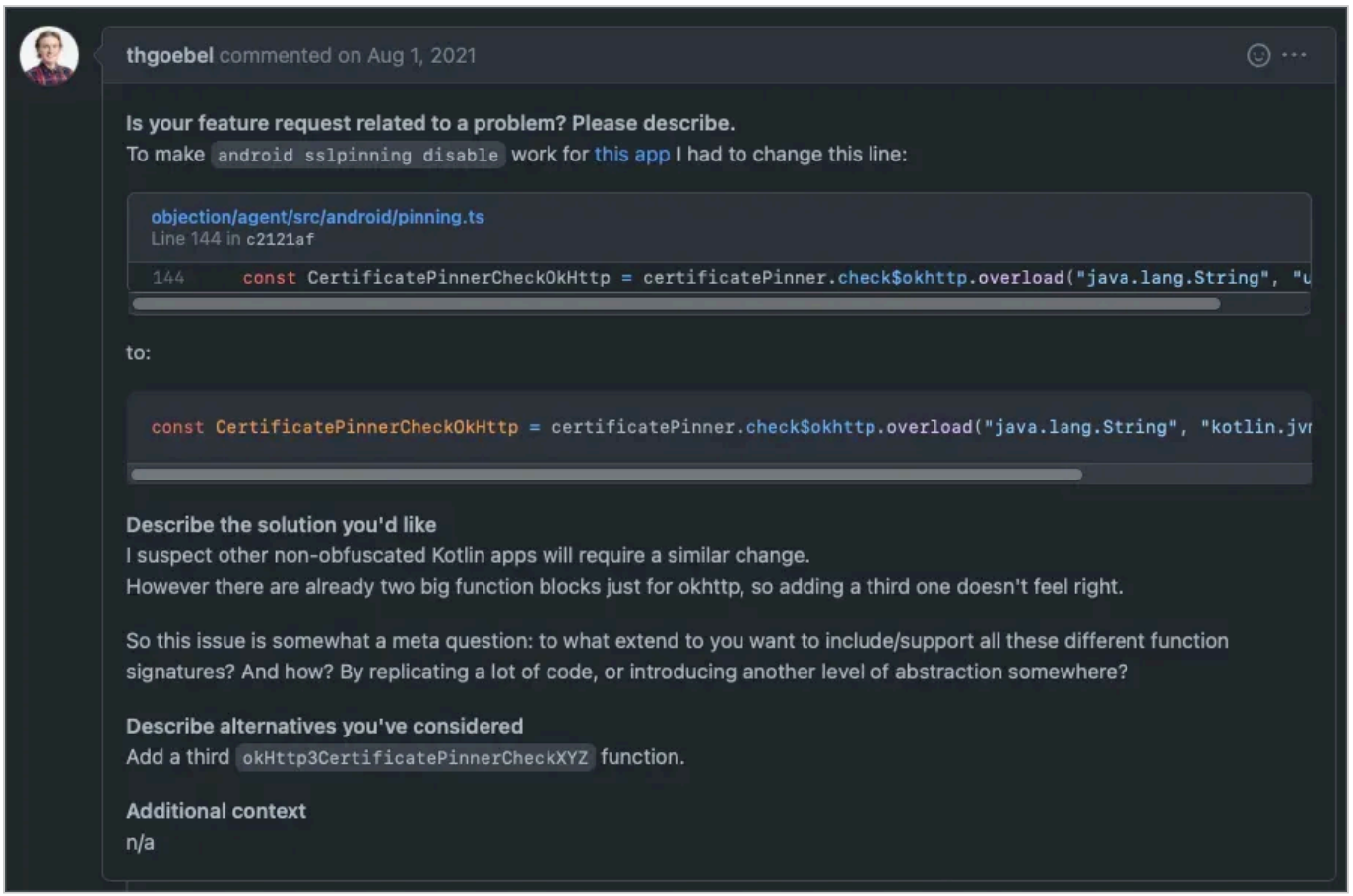
[Mi A2::PID::8988]->       ==================
     Debugging Stack Trace for SSLPeerUnverifiedException
     Thrown by okhttp3.CertificatePinner->check$okhttp
```

Frida attach

Licensed under CC BY-SA 4.0

From the message above, it is clear that error message is throwing by okhttp3.CertificatePinner->check$okhttp.

After hours googling, I found that someone experience this issue too when using objection https://github.com/sensepost/objection/issues/475 (https://github.com/sensepost/objection/issues/475)




From the above message, he suspect non-obfuscated Kotlin apps need little change for function hook in the script. Our app is non-obfuscated kotlin app too, so it’s relevant issue.

Licensed under CC BY-SA 4.0 

```
    return wrapJavaPerform(() => {
      try {
        const certificatePinner: CertificatePinner = Java.use("okhttp3.CertificatePinner");
        send(c.blackBright(`Found okhttp3.CertificatePinner, overriding CertificatePinner.check$okhttp()`));

        const CertificatePinnerCheckOkHttp = certificatePinner.check$okhttp.overload("java.lang.String", "u15");

        // tslint:disable-next-line:only-arrow-functions
        CertificatePinnerCheckOkHttp.implementation = function() {
          qsend(quiet,
            c.blackBright(`[${ident}] `) + `Called check$okhttp ` +
            c.green(`OkHTTP 3.x CertificatePinner.check$okhttp()`) +
            `, not throwing an exception.`,
          );
        };

        return CertificatePinnerCheckOkHttp;

      } catch (err) {
        if (err.message.indexOf("ClassNotFoundException") === 0) {
          throw new Error(err);
        }
      }
    });
  };
```

**Creating Bypass Script for non-obfuscated Kotlin**

We will add the script from objection but change the function as **thgoebel** (https://github.com/thgoebel) mentioned before.

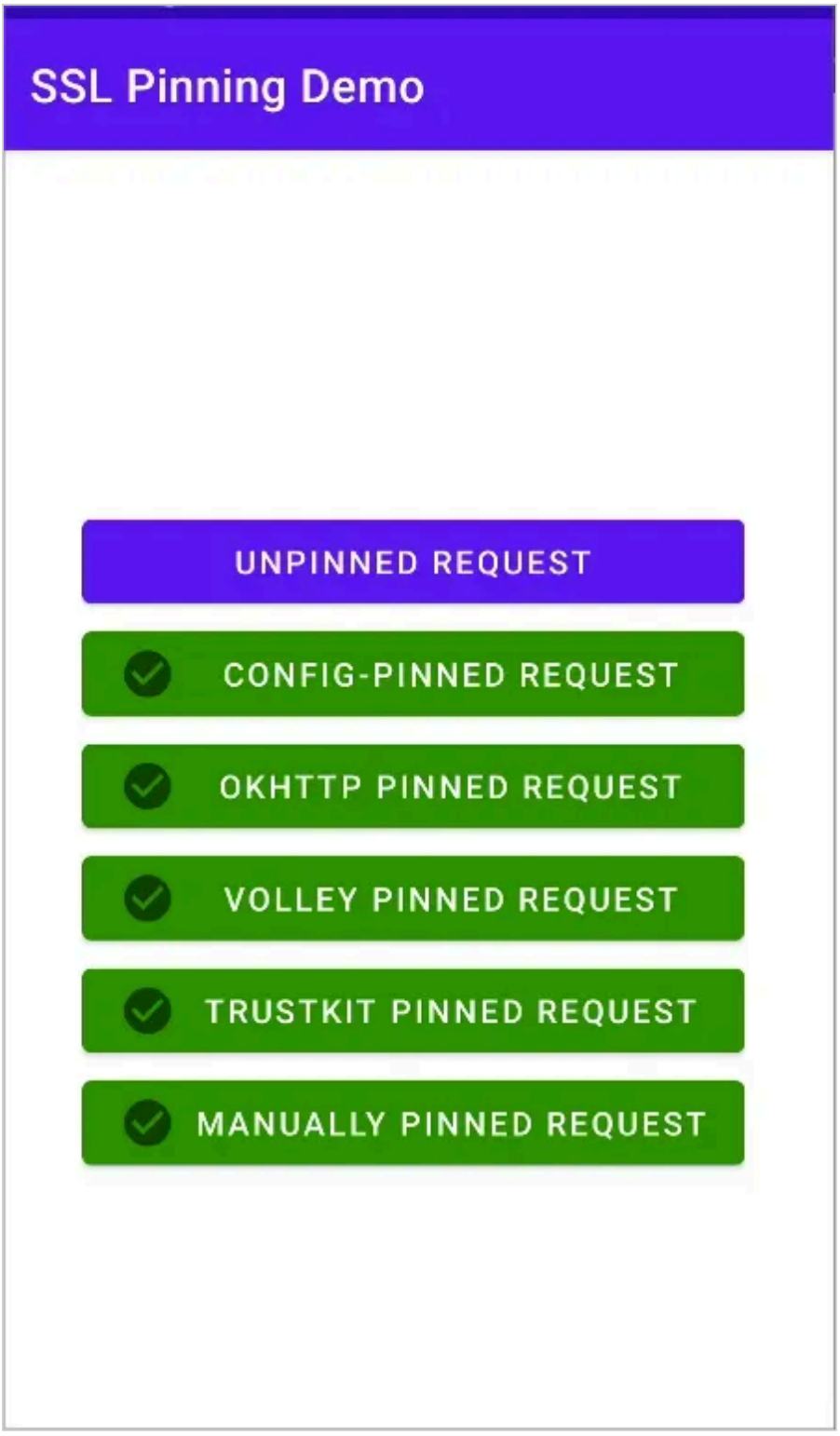
Save and click the Okhttp Pinned Request again. No need to exit and relaunch frida, because it will automatically reload the script.

```
     ____
    / _  |   Frida 15.1.17 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/

[Mi A2::PID::8988]->       ==================
      Debugging Stack Trace for SSLPeerUnverifiedException
      Thrown by okhttp3.CertificatePinner->check$okhttp
[Mi A2::PID::8988]-> [+] OkHTTPv3 ($okhttp)
[+] OkHTTPv3 ($okhttp)
  --> Bypassing OkHTTPv3 ($okhttp): petruknisme.com
```

Success okhttp bypass pinning

Our final script will success to bypass the ssl pinning from okhttp as below




Sadly, we reach the end of the article. I hope this article will be guidance for newcomer when doing mobile application pentest but you don't know what to do to bypass the ssl pinning. If you like what you read, you can share this article to everyone you think need to know.

Feedback are always welcome because it's valuable thing and there will be always room for improvement. Thank you!

Licensed under CC BY-SA 4.0 

Originally published on **Medium** (https://medium.com/@petruknisme/easy-way-to-bypass-ssl-pinning-with-objection-frida-beginner-friendly-58118be4094).

# Android # Security # Tutorial

Licensed under CC BY-SA 4.0