FIELD NOTE / ANDROID

Let’s Bypass SSL Pinning By Manual Modification

Petruknisme · 17 Mar, 2021 · 2 min to read

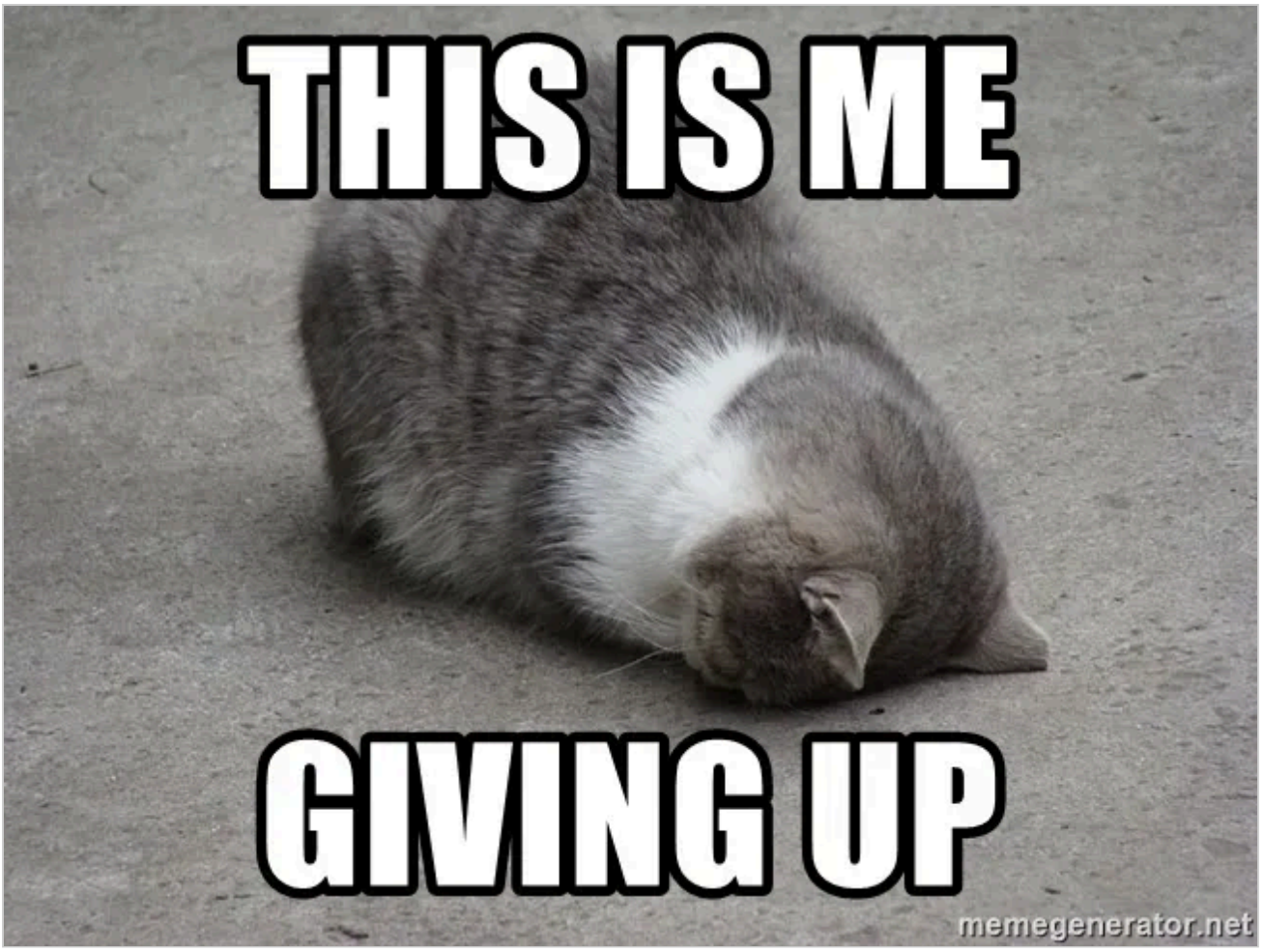
SSL Pinning is the trickiest part when doing penetration testing for android application target. You may hear of frida, objection, and other methods doing the great job for bypassing android SSL pinning. But, what if all methods is not suitable for your target? ssl pinning system function is not covered by common tools? Let’s assume that.

So, what’s next? Throwing your phone? Tell your manager that the application is super duper hard to bypass? HAHAHAHA



Source: memegenerator.net

Licensed under CC BY-SA 4.0



Source: memegenerator.net

Relax, there are still many roads to mars. Did you ever do reverse engineer/unpacking android applications? We still can use this technique to bypass SSL pinning, for some reason doing manual modification is better than relying on tools. But how?

Debugging Android APP

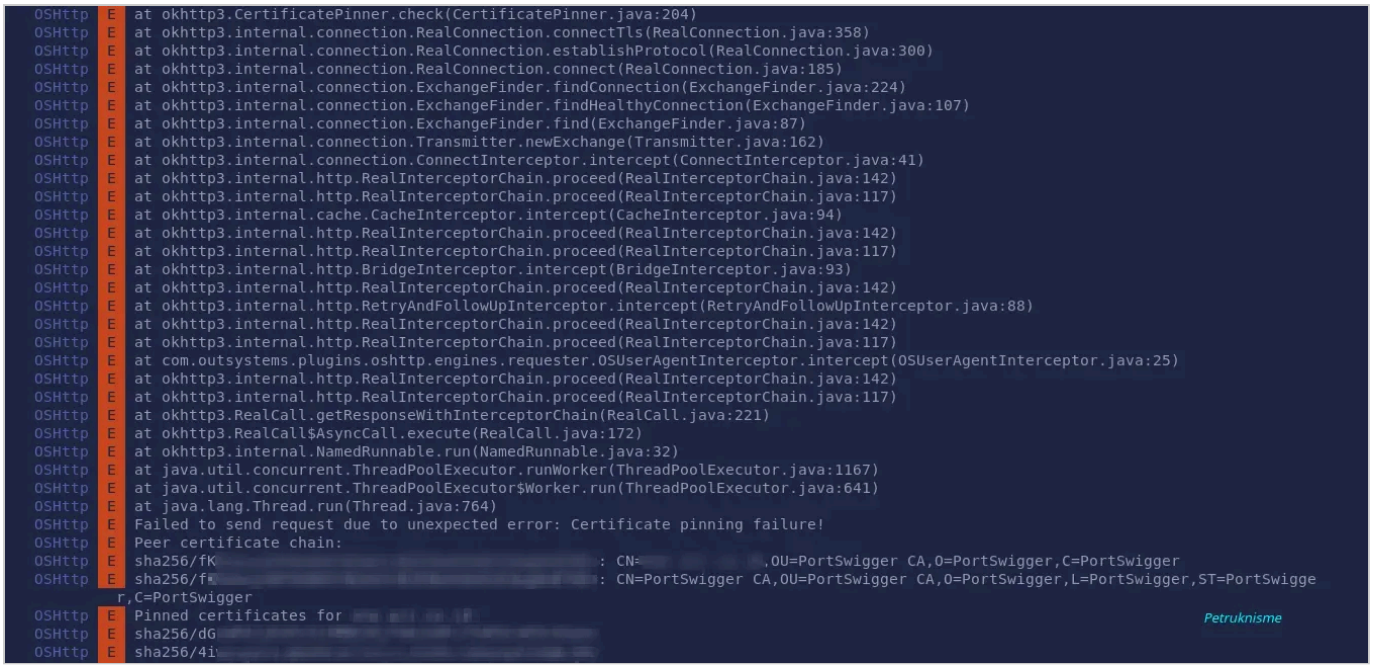
We need to debug our android app so we can know what's going on. Plug your android and run

```
$ [sudo] pip install logcat-color # install logcat-color
$ adb logcat | logcat-color
```

BOOOM! You will see android debug log that coming so fast HAHAHA

To know what's going on, try to run your application and make an HTTP request. After that, Ctrl+C the logcat and scroll up to see the logs. Search for android certificate error like mine(using okhttp3)

Licensed under CC BY-SA 4.0



Android Certificate Error

As you can see, android certificate pinning is failing and complaining about mismatch certificate hash. Our peer certificate chain is for PortSwigger(BurpSuite), but the Pinned Certificate for a domain(masked) is defined in the android resource file.

Unpacking Android APK

The android developer will store the certificate pin on the resource file in the android apk. So we need to unpack it first. I'm using apktool for doing this task.

```
java -jar apktool.jar d base.apk -o decompiled
I: Using Apktool 2.5.0-dirty on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /home/user/.local/share/apktool/frame
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

Licensed under CC BY-SA 4.0

Look at the resource folder and search for the pinning file.

```
grep -r "sha256/dGxxxxxxxxxxx" decompiled/* #adjust with your folder
```

For me, it's at resource/ assets /www/pinning/ because I'm pentesting cordova application based, please adjust with your environment.

Edit the file and inject with your burp suite sha256 hash from the output log, mine is `sha256/fKR2xxxxxxxxxxx`

That's it. Now we need to repack/recompile again so we can run this on the android.

Recompile APK

```
$ java -jar apktool.jar b decompiled -o app-certpin-bypass.apk
I: Using Apktool 2.5.0-dirty
I: Checking whether sources has changed...
I: Checking whether resources has changed...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
$ ls
app-certpin-bypass.apk  base.apk  decompiled
```

We can't just install it after recompiled, the apk needs to be re-signed. We can use uber-apk-signer (https://github.com/patrickfav/uber-apk-signer)

```
java -jar uber-apk-signer.jar -a app-certpin-bypass.apk
```

Or if you want to sign all apk in specific directory

Licensed under CC BY-SA 4.0

```
java -jar uber-apk-signer.jar --apks /path/to/apks
```

Output:

Output

```
uber-apk-signer -a app-certpin-bypass.apk
source:
        /home/user/apk
zipalign location: PATH
        /bin/zipalign
keystore:
        [0] 161a0018 /tmp/temp_10038107399149177087_debug.keystore (DEBUG_
        file: /home/user/apk/app-certpin-bypass.apk (24.53 MiB)
        checksum: 79489a567c31bcc1f82bf45b5d9d6ffa1de2ca71487a538505b31a8b
        - zipalign success
        - sign successVERIFY
        file: /home/user/apk/app-certpin-bypass-aligned-debugSigned.apk (2
        checksum: 5ade3620151dd81029508faa40fa1a63240cf1e55150702aa79b3928
        - zipalign verified
        - signature verified [v1, v2, v3]
                Subject: CN=Android Debug, OU=Android, O=US, L=US, ST=US,
                SHA256: 1e08a903aef9c3a721541b54ec764e01d3d094eb954161b625
                Expires: Fri Mar 11 03:10:05 WIB 2044[Wed Mar 17 16:36:55
Successfully processed 1 APKs and 0 errors in 3.08 seconds.
```

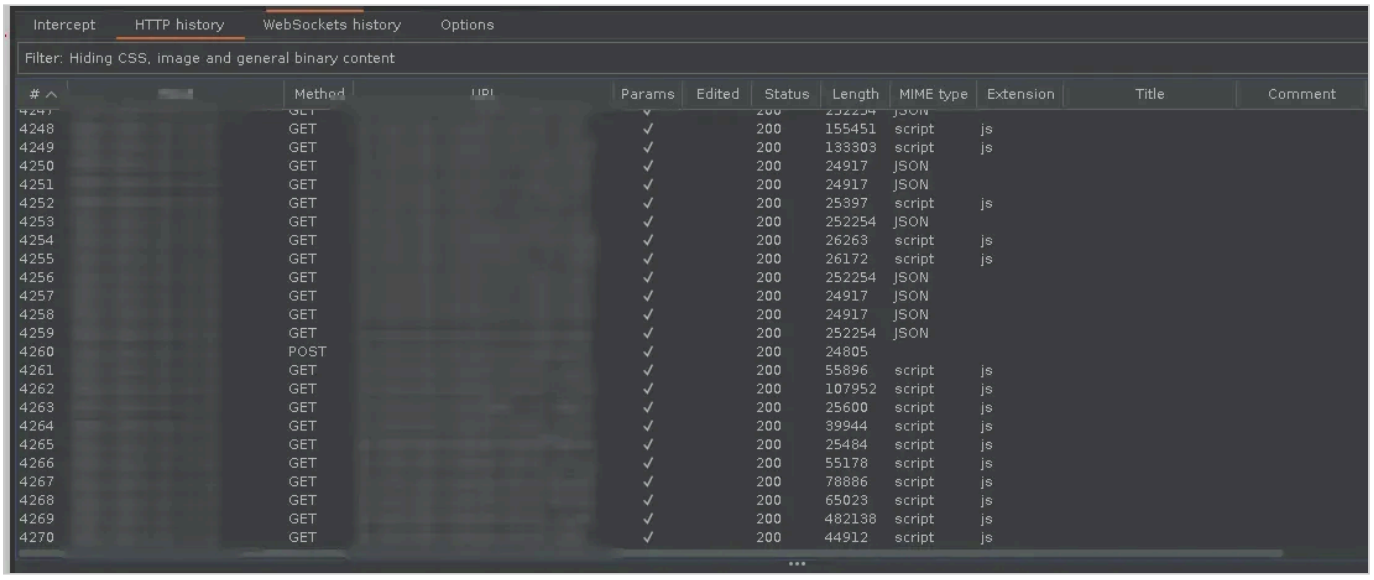
It will use android debug certificate, so it can be installed on our phone.

```
adb install app-certpin-bypass-aligned-debugSigned.apk
```

Run the application and check BurpSuite interceptor. VOILAAAAAAAAAA

Licensed under CC BY-SA 4.0





#	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment
4248	GET		✓		200	155451	script	js		
4249	GET		✓		200	133303	script	js		
4250	GET		✓		200	24917	JSON			
4251	GET		✓		200	24917	JSON			
4252	GET		✓		200	25397	script	js		
4253	GET		✓		200	252254	JSON			
4254	GET		✓		200	26263	script	js		
4255	GET		✓		200	26172	script	js		
4256	GET		✓		200	252254	JSON			
4257	GET		✓		200	24917	JSON			
4258	GET		✓		200	24917	JSON			
4259	GET		✓		200	252254	JSON			
4260	POST		✓		200	24805				
4261	GET		✓		200	55896	script	js		
4262	GET		✓		200	107952	script	js		
4263	GET		✓		200	25600	script	js		
4264	GET		✓		200	39944	script	js		
4265	GET		✓		200	25484	script	js		
4266	GET		✓		200	55178	script	js		
4267	GET		✓		200	78886	script	js		
4268	GET		✓		200	65023	script	js		
4269	GET		✓		200	482138	script	js		
4270	GET		✓		200	44912	script	js		

Burpsuite success intercept and bypass ssl pinning

That’s all, if you have any question just tell me in the comment section and feedback is welcome ❤

Originally published on **Medium** (https://medium.com/@petruknisme/lets-bypass-ssl-pinning-by-manual-modification-617afc08a079).

Licensed under CC BY-SA 4.0

Android # Security # Tutorial

Licensed under CC BY-SA 4.0