

SSL Pinning with Different Variations

Ahmethan Gültekin



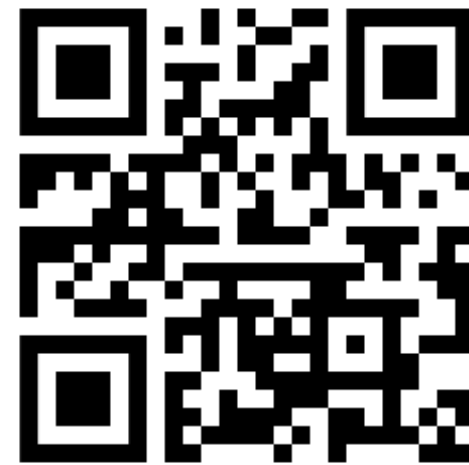
About

reverse engineer, mobile analyzer, former web security researcher

working on Byteria



ahmeth4n.github.io



byterialab.com

Context

General Context: Understanding SSL, How does SSL pinning work?

Java Stuff: SSL unpinning with different perspectives

Flutter Environment: DartVM, Object Pools and finally SSL Unpinning

Examples similar to real-life applications: Example lab apps analysing
a little bit of **radare2** and **frida scripting**...

Helpers

- radare2
- frida
- ghidra
- blutter
- reflutter

SSL Pinning

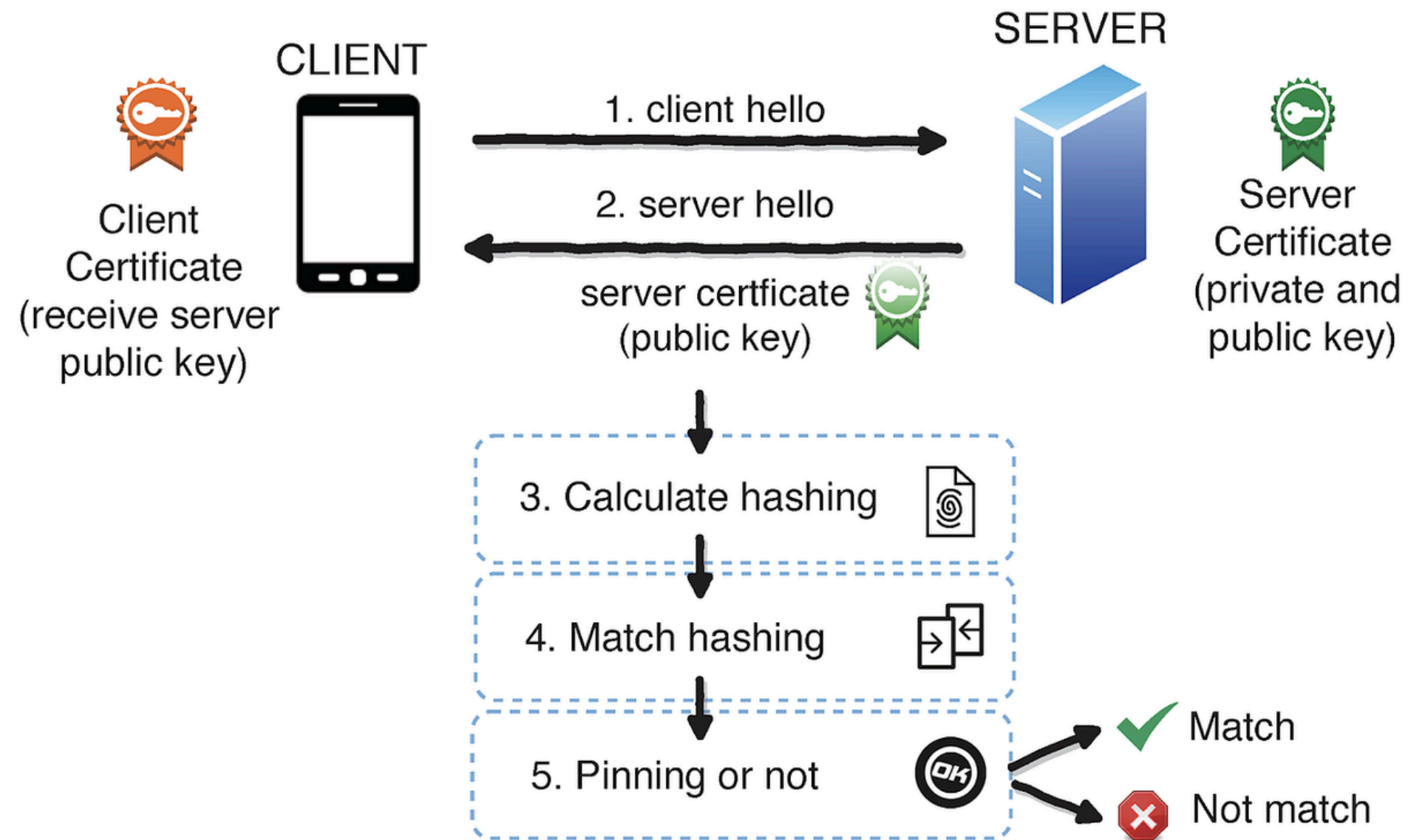
(what is the SSL? how to works pinning?)

SSL Pinning

SSL Pinning is a security technique that ensures an app communicates only with a trusted server by hardcoding the server's certificate or public key. This helps prevent man-in-the-middle (**MITM**) attacks, even if a device's root certificates are compromised.



SSL Pinning



Hash Calculation

```
openssl x509 -in badssl.com.pem -pubkey -noout | openssl pkey -pubin -outform der |  
openssl dgst -sha256 -binary | openssl enc -base64
```

- extract public key from X.509 certificate.
- public key converting to DER format
- SHA-256 hashing - binary output
- binary base64 encoding

A Different Perspective on SSL Pinning

(there is always an alternative way.)

Okhttp3 Basic Example

```
CertificatePinner certificatePinner = new CertificatePinner.Builder()
    .add( pattern: "*.badssl.com", ...pins: "sha256/0hhQVWECWs2VYm1Flexz7U0IeMs6UUT6G6o6Rs9StLU=")
    .build();

return new OkHttpClient.Builder()
    .sslSocketFactory(sslContext.getSocketFactory(), (X509TrustManager) tmf.getTrustManagers()[0])
    .certificatePinner(certificatePinner)
    .build();
```

(Example OkHttpClient Certificate Pinning)

Okhttp3 SSL Pinning

Okhttp3 have a CertificatePinner class for basic pinning stuffs. This class get an alias domain and calculated hash of main certificate file from parameter. The check() method calculates the hash of the certificate and compares to calculated hash vs parameter hash.

Okhttp3 Basic Example

As expected, this fails with a certificate pinning exception:

```
javax.net.ssl.SSLPeerUnverifiedException: Certificate pinning failure!
Peer certificate chain:
  sha256/afwiKY3RxoMmLkuRW1l7QsPZTJPwDS2pdDR0QjXw8ig=: CN=publicobject.com, OU=PositiveSSL
  sha256/kl023nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY=: CN=COMODO RSA Secure Server CA
  sha256/grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME=: CN=COMODO RSA Certification Authority
  sha256/lCppFqbkr1J3EcVFAkeip0+44VaoJUymbnOaEUk7tEU=: CN=AddTrust External CA Root
Pinned certificates for publicobject.com:
  sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
at okhttp3.CertificatePinner.check(CertificatePinner.java)
at okhttp3.Connection.upgradeToTls(Connection.java)
at okhttp3.Connection.connect(Connection.java)
at okhttp3.Connection.connectAndSetOwner(Connection.java)
```

Follow up by pasting the public key hashes from the exception into the certificate pinner's configuration:

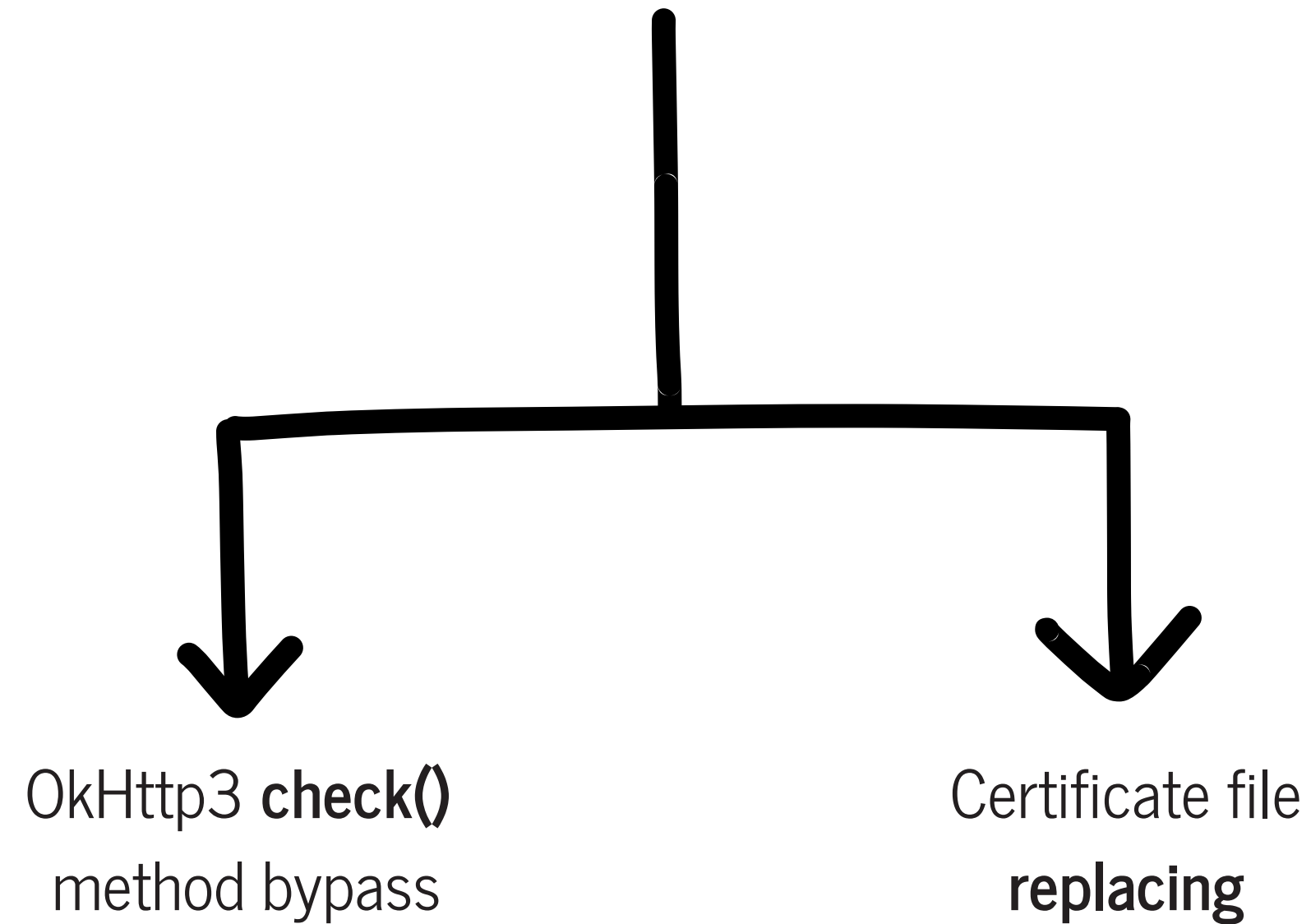
```
CertificatePinner certificatePinner = new CertificatePinner.Builder()
    .add("publicobject.com", "sha256/afwiKY3RxoMmLkuRW1l7QsPZTJPwDS2pdDR0QjXw8ig=")
    .add("publicobject.com", "sha256/kl023nT2ehFDXCfx3eHTDRESMz3asj1muO+4aIdjiuY=")
    .add("publicobject.com", "sha256/grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME=")
    .add("publicobject.com", "sha256/lCppFqbkr1J3EcVFAkeip0+44VaoJUymbnOaEUk7tEU=")
    .build();
```

Pinning is per-hostname and/or per-wildcard pattern. To pin both `publicobject.com` and `www.publicobject.com`, you must configure both hostnames.

(Certificate pinning failure error message)

OkHttp3 Pinning Bypass

there are 2 ways here;



Okhttp3 SSLPinning Class

As expected, this fails with a certificate pinning exception:

```
javax.net.ssl.SSLPeerUnverifiedException: Certificate pinning failure!
Peer certificate chain:
  sha256/afwiKY3RxoMmLkuRW1l7QsPZTJPwDS2pdDR0QjXw8ig=: CN=publicobject.com, OU=PositiveSSL
  sha256/kl023nT2ehFDXCfx3eHTDRESMz3asj1mu0+4aIdjiuY=: CN=COMODO RSA Secure Server CA
  sha256/grX4Ta9HpZx6tSHkmCrvpApTQGo67CYDnvprLg5yRME=: CN=COMODO RSA Certification Authority
  sha256/lCpPfqbkrlJ3EcVFAkeip0+44VaoJUymbn0aEUk7tEU=: CN=AddTrust External CA Root
Pinned certificates for publicobject.com:
  sha256/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA=
at okhttp3.CertificatePinner.check(CertificatePinner.java)
at okhttp3.Connection.upgradeToTls(Connection.java)
at okhttp3.Connection.connect(Connection.java)
at okhttp3.Connection.connectAndSetOwner(Connection.java)
```

CertificatePinner.**check()** method compares the calculated hash of the certificate with the hash given as a parameter to its own method and verifies it.

Certificate Replacing

```
public static OkHttpClient getPinnedHttpClient(Context context, String hostname) {  
    try {  
        CertificatePinner certificatePinner = new CertificatePinner.Builder()  
            .add( pattern: "*.badssl.com", ...pins: "sha256/0hhQVWECWs2VYm1F1exz7U0IeMs6UUT6G6o6RsqsStLU=" )  
            .build();  
  
        return new OkHttpClient.Builder()  
            .certificatePinner(certificatePinner)  
            .build();  
    } catch (Exception e) {  
        throw new RuntimeException(e);  
    }  
}
```

we already have BurpSuite's certificate and calculated hash. why don't we change this?

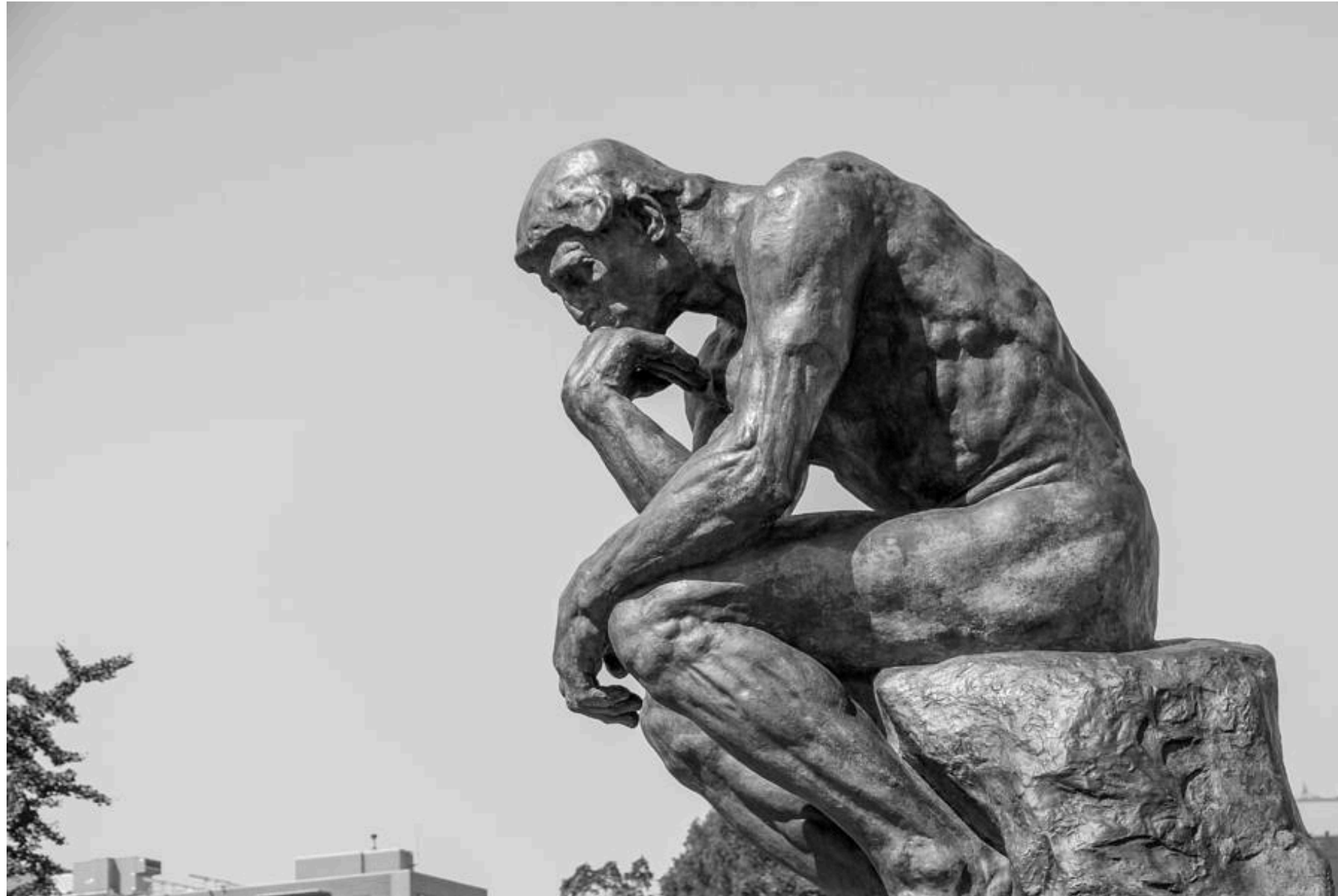
Practice

(let's practice and learn.)


**are we really need SSL Pinning Bypass
for request intercepting?**


(nope, but it's not short like that.)

Brainstorm



Alternative Ways

If we have any using methods in request methods, we can hook this method instead of main methods. 

Also, if application has a custom request library (like okhttp3 wrapper class) for http requesting, we still have one other way instead of SSL pinning. 

We have many ways like this. You can choose whichever way works best for you and your applications.

Let's See

(let's look at alternative ways.)

Flutter App Analysing

(let's take a closer look Flutter environment.)

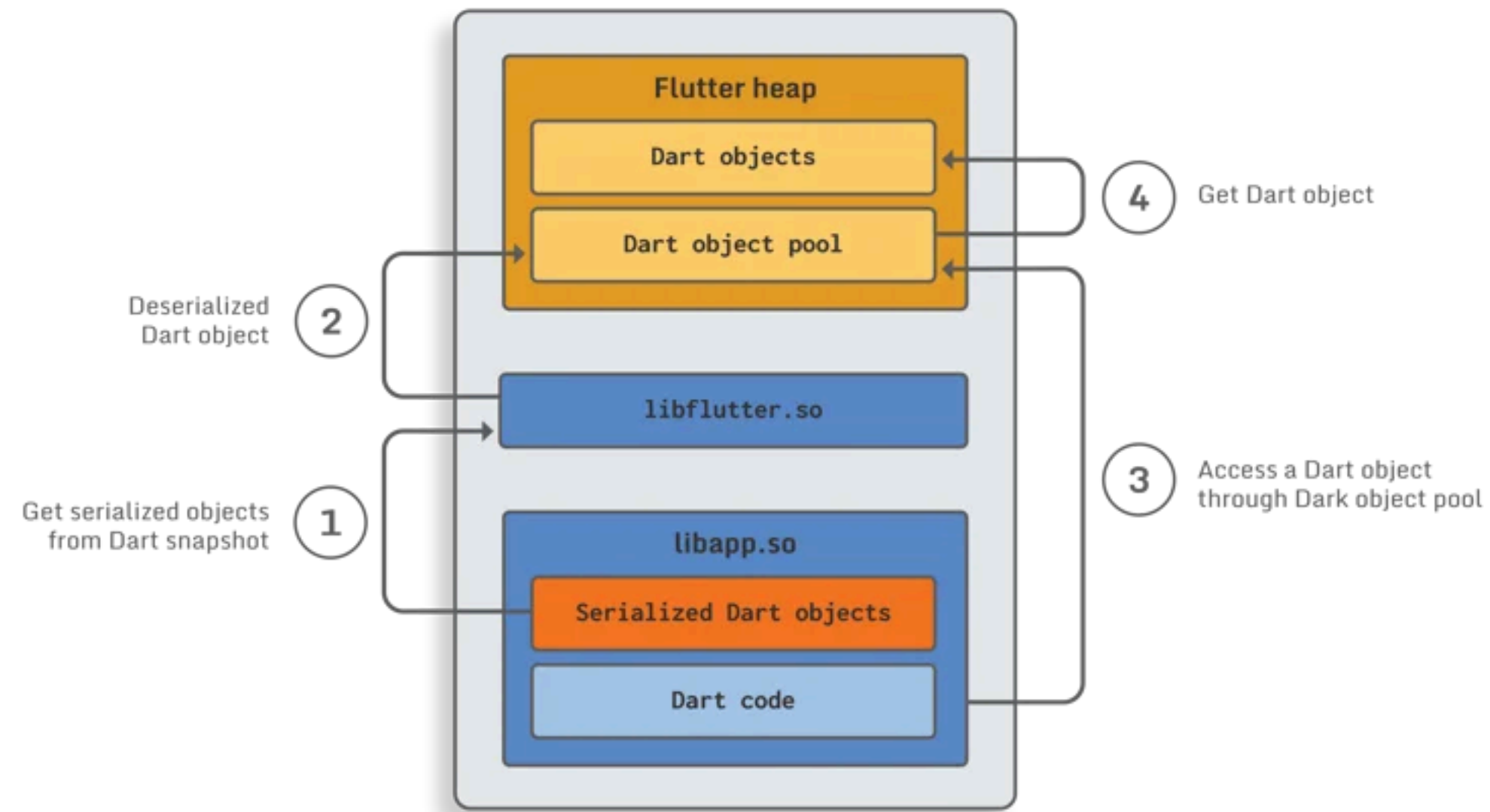
Flutter Basics

libapp.so: main file. Dart codes compiled version. includes business logic and all things of the main code.

libflutter.so: Flutter Engine, not includes business logic.

snapshot (in libapp.so): compiled Dart object data. includes object pool (strings, types), object table, and header info. used by Dart runtime to restore app state and logic.

Flutter Structure



Critical Points

- **libflutter.so** has all VM structure we need.
- **libapp.so** cannot access memory to directly. all heap processes managed by **libflutter.so**.
- **libapp.so** contains the compiled Dart app snapshot. but it's managing/executing by **libflutter.so**

Flutter Proxy Rules

```
emu64a:/ # settings put global global_http_proxy_host 192.168.1.211
emu64a:/ # settings put global global_http_proxy_port 8083
UTPUT -p tcp --dport 443 -j DNAT --to-destination 192.168.1.211:8083
UTPUT -p tcp --dport 443 -j DNAT --to-destination 192.168.1.211:8083
emu64a:/ #
emu64a:/ # █
```

global **HTTP** proxy settings and configuring **iptables** rules.

```
emu64a:/ # iptables -t nat -F
emu64a:/ # settings put global global_http_proxy_host null
emu64a:/ # settings put global global_http_proxy_port 0
emu64a:/ # █
```

reset all stuffs.

Useful Radare2 Commands

aaaa → analyze (advanced).

pd → print disassemble code

pdg → print disassembled **ghidra**
generated code (extension needed)

pdC → print disassembled **C**
psuedo code

afn → **rename** function

iz → **search** strings

s → jump the address (seek).

V → view mode (so cool)

axf → x-refs list

Practice Reversing Flutter apps with Radare2

(let's do this..)

Last Words

- **radare2** so cool, ~~flutter~~ is sucks for **RE**.

Useful Links

https://scoding.de/uploads/r2_cs.pdf

https://github.com/fatalSec/flutter_reversing

<https://github.com/worawit/blutter>

<https://github.com/Impact-I/reFlutter>

<https://blog.byterialab.com/>