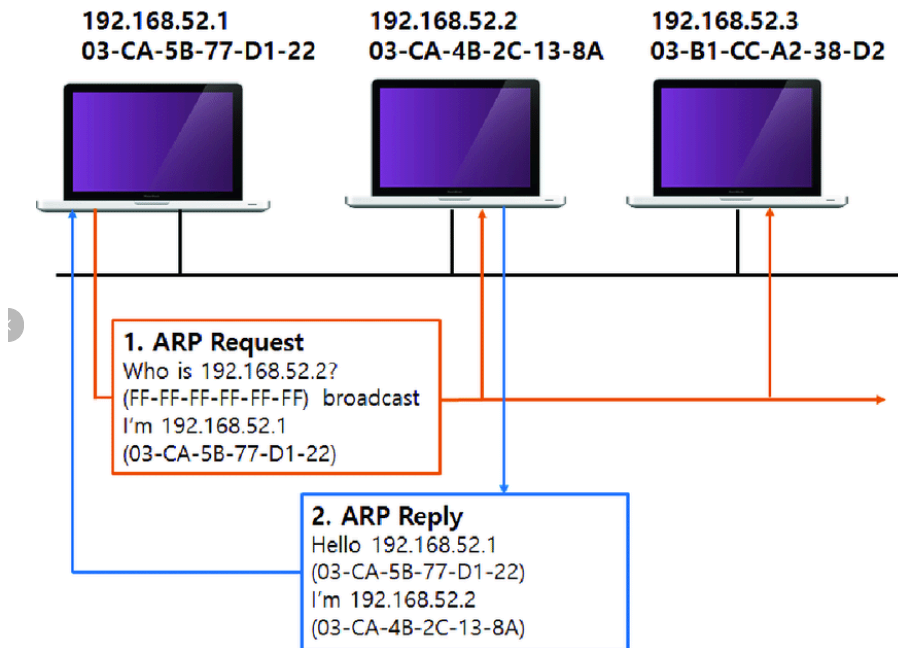


5-Man in the Middle

1-ARP (Address Resolution Protocol)



Address Resolution Protocol (ARP) attack

البروتوكول ده بيستخدم علشان جهاز يعرف Mac address بتاع جهاز تاني وده لو جهازين عاوزين يتواصلو مع بعض A and B

Host A have :

- source ip :192.168.1.3
- dist ip : 192.168.1.4
- source mac
- dist mac ?????

1- ARP request : A need to know what is the mac address for B, so A send broadcast over network?
what is the mac address for this ip (192.168.1.4) ,

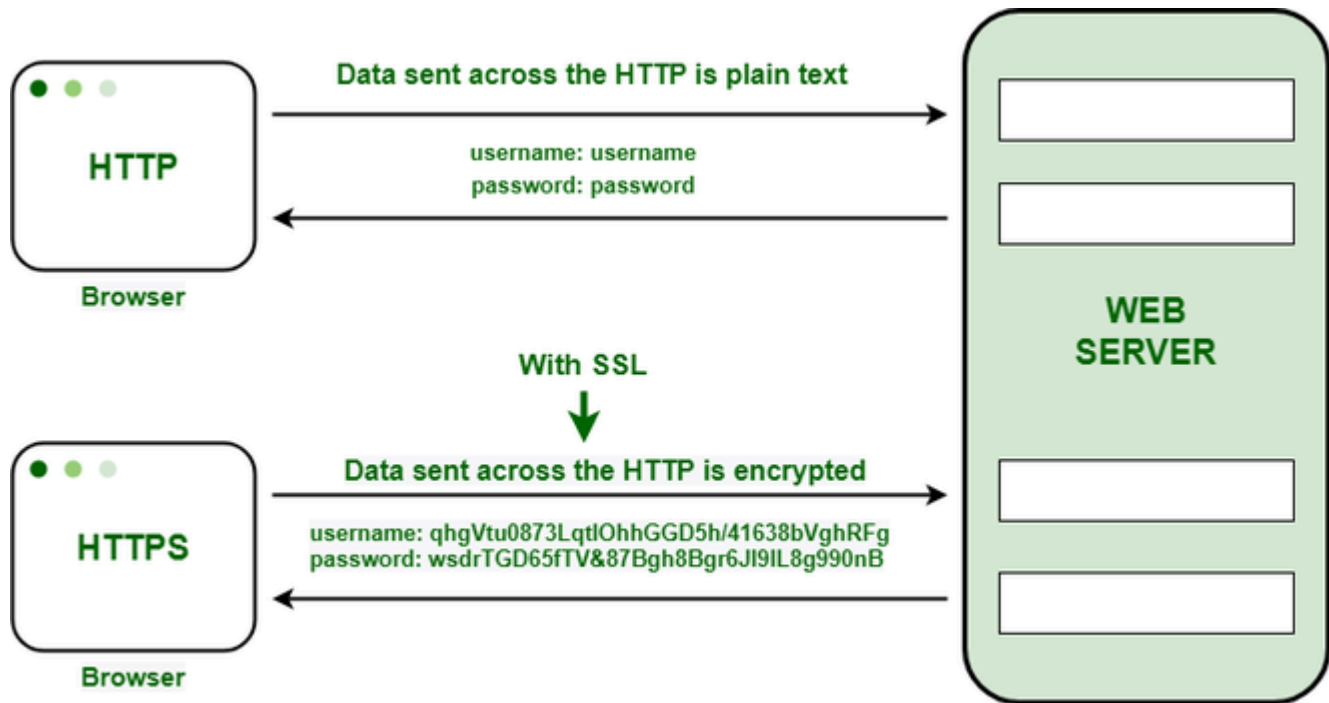
2- ARP replay : B send to A i am the host and my mac address is (mac)

3- after knew the mac address for B, A send message to B

بعد كده جهاز A بيخزن عنده المعلومات اللي حصل عليها اللي هما ip and mac for host B وده بقي بيقى اسمه ARP cache

HTTP and HTTPS

عاوزين نعرف ايع الفرق بين 2 protocol دول ونروح لاول واحد هو HTTP (Hyper Text Transfer Protocol) بيبقي علي port 80 ونجي لا هم نقطة وهي طريقة التواصل طريقة التوائل بتكون غير مشفرة يعني لو احنا في شبكة عادية وشخص عمل login باستخدام http اي حد لو عمل sniffing او MITM يقدر يشوف username and password علشان كده بقي بنستخدم https (Hyper Text Transfer Protocol Secure) on port 443 ده بيقى secure بستخدم SSL and TLS encryption



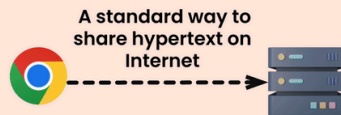
how https work

HTTPS Summarised



What is HTTP?

Hyper Text Transfer Protocol

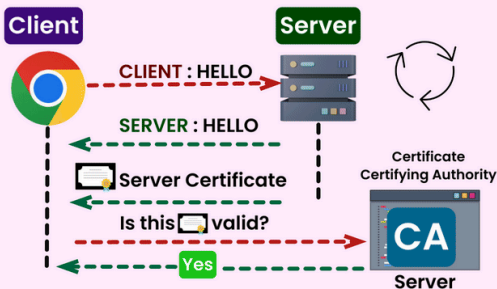


Server

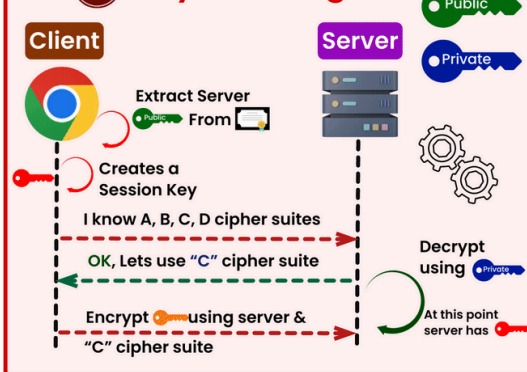
Hyper Text : Interconnected text allowing non-linear navigation, transforming how information is accessed and shared on the internet.



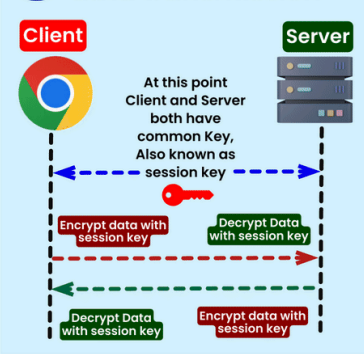
1 Server Certificate Check



2 Key Exchange



3 Encrypted Tunnel for data transmission



1. The client searches for google.com

- The user enters "google.com" in the browser.
- The browser starts a process to resolve the domain name to an IP address (DNS Resolution).

2. The browser resolves the domain name and gets the IP address of google.com

- The browser queries DNS servers to find the IP address associated with the domain name "google.com."

3. The browser sends a request to the IP address of google.com

- An HTTP request is sent over HTTPS to the Google server.

4. The Google server responds with a certificate and public key

- The server sends a digital certificate that contains:
 1. The **public key** of the server.
 2. Information about the server's identity (e.g., domain name).
 3. A digital signature from a trusted Certificate Authority (CA).

5. The client verifies the certificate via certificate servers (cert.sh)

- The browser queries a certificate server (like OCSP or CRL) to ensure the certificate is valid and hasn't been revoked.
 - The server responds confirming the certificate's validity.
-

6. The client verifies the certificate is issued by a trusted authority

- The certificate includes a digital signature created by the issuing Certificate Authority.
 - The browser checks the signature using a list of trusted root certificates stored in the operating system.
-

7. The client sends information about supported encryption algorithms (Cipher Suites)

- The browser sends a list of supported encryption methods (e.g., AES, RSA, ECDHE) to the server.
-

8. The server selects the encryption method and sends it to the client

- The server chooses the most appropriate encryption method from the client's list and responds with the selected cipher suite.
-

9. The client generates a session key

- The browser generates a random session key, which will be used to encrypt the data.
-

10. The client encrypts the session key using the public key received in the certificate

- The session key is encrypted with the public key of the server.
 - The encrypted session key is sent to the server.
-

11. The server decrypts the session key using its private key

- The server uses its private key to decrypt the session key sent by the client.
-

12. Communication between the client and the server is encrypted using the session key

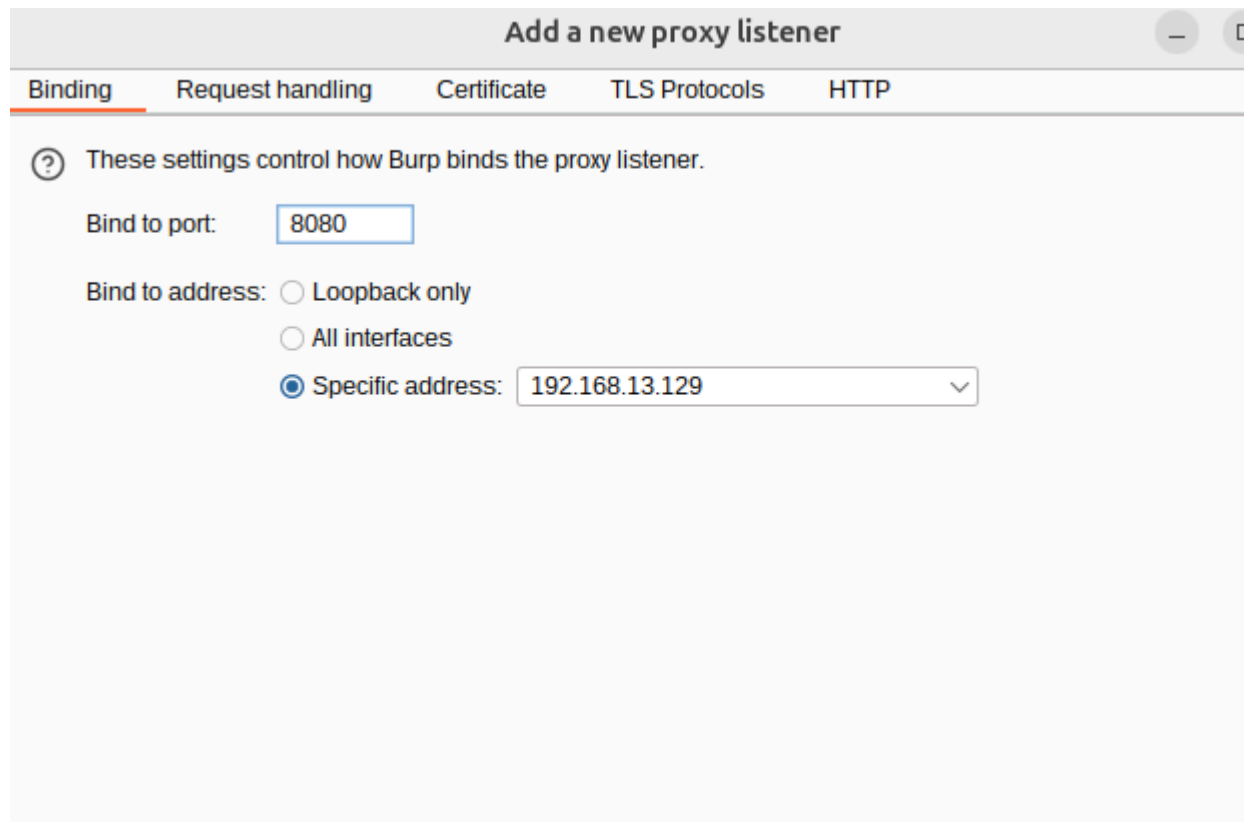
- All subsequent communication between the client and the server is encrypted using the session key.
 - Symmetric encryption (e.g., AES) is used for efficiency and security.
-
-

Install Burpsuite and connect to emulator

1- install burpsuite on ubuntu

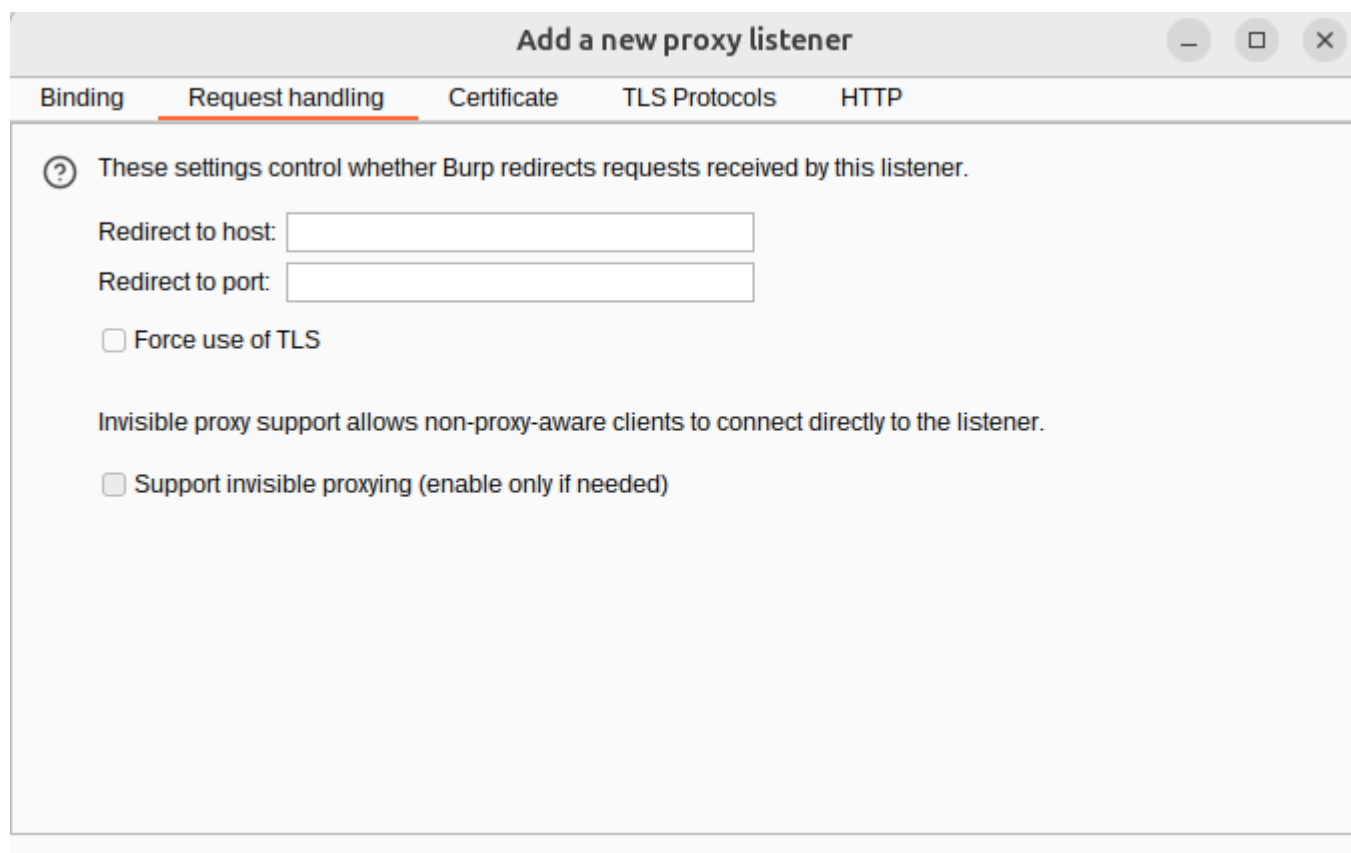
2- configer burpsutite and change the proxy setting

add ip of ubuntu on burp



The screenshot shows the 'Add a new proxy listener' dialog box in Burp Suite. The 'Binding' tab is selected. The text says: 'These settings control how Burp binds the proxy listener.' Below this, there are three fields: 'Bind to port:' with the value '8080', 'Bind to address:' with three radio button options: 'Loopback only', 'All interfaces', and 'Specific address:'. The 'Specific address:' option is selected, and its dropdown menu shows the IP address '192.168.13.129'.

enable support invisible proxying



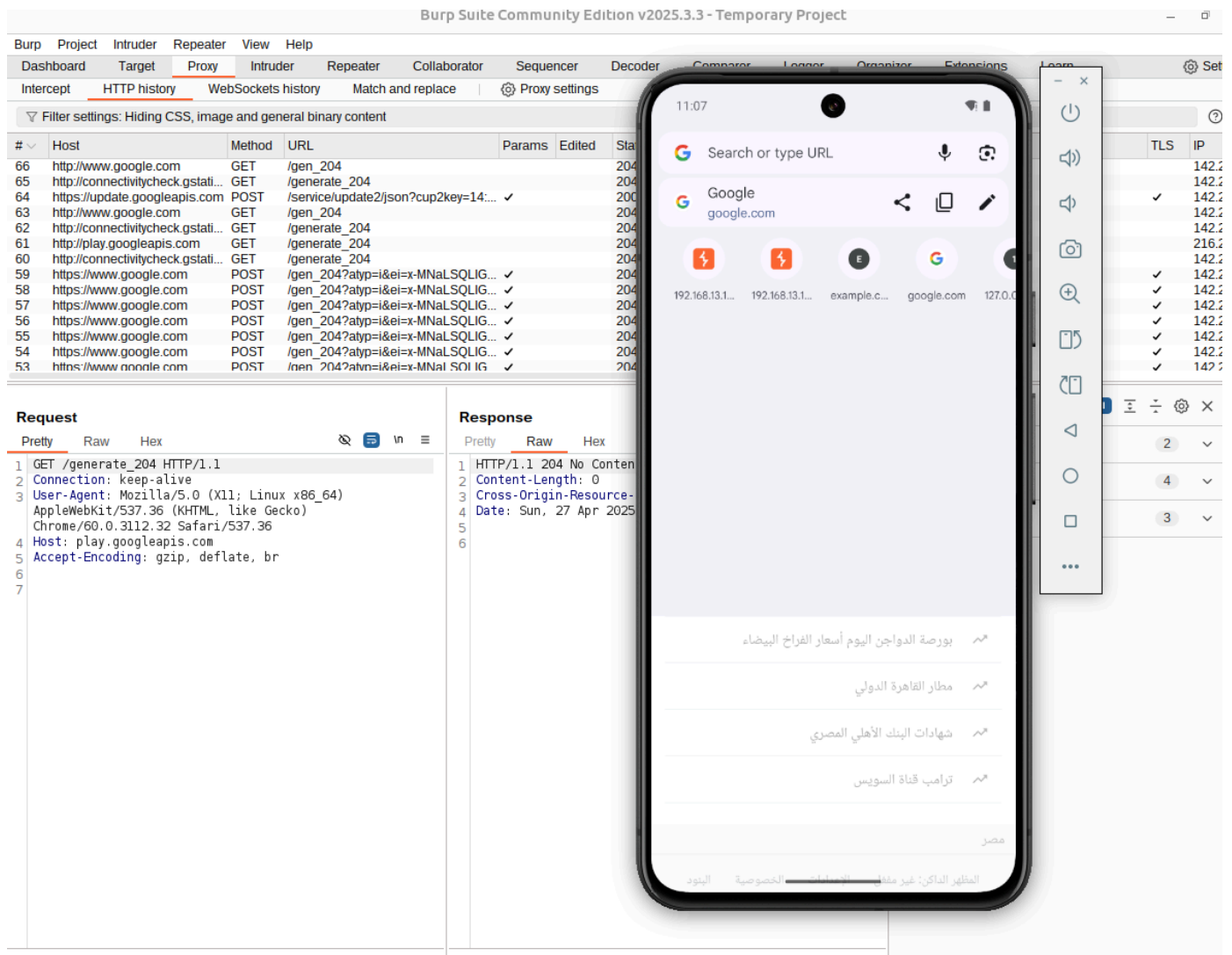
The screenshot shows the 'Add a new proxy listener' dialog box in Burp Suite, with the 'Request handling' tab selected. The text says: 'These settings control whether Burp redirects requests received by this listener.' Below this, there are two input fields: 'Redirect to host:' and 'Redirect to port:'. There is a checkbox labeled 'Force use of TLS'. Below that, there is a paragraph: 'Invisible proxy support allows non-proxy-aware clients to connect directly to the listener.' and a checkbox labeled 'Support invisible proxying (enable only if needed)'.

on emulator open google and search for this : <http://192.168.13.129:8080> and install certificate

on emulator go to -> setting -> security & privacy -> certificate and install certificate

open wifi and configer wifi and set proxy ip of ubuntu and port 8080

open burp



Certificates /Openssl (Important)

يعني إيه Certificate Pinning؟

لما أي تطبيق بيتصل بالسيرفر، الاتصال ده بيكون مشفّر (HTTPS). السيرفر بيقدّم شهادة (Certificate) عشان يثبت للتطبيق إنه سيرفر حقيقي ومش مزيف.

هي طريقة بيستخدمها التطبيق عشان يتأكد إن الشهادة اللي جاية من السيرفر هي اللي هو متوقعها بالضبط. لو الشهادة دي مش مطابقة أو لو التطبيق شاف إن فيه وسيط زي Burp Suite بيحاول يدخل في النص، التطبيق يقطع الاتصال فورًا.

ليه بنحتاج نعمل Bypass Certificate Pinning؟

لما نكون بنعمل Penetration Testing، بنحتاج نشوف البيانات اللي التطبيق بيبيعها للسيرفر أو اللي بيستقبلها منه. لكن لو التطبيق مفعّل Certificate Pinning، ما نقدرش نعرض البيانات باستخدام أدوات زي Burp Suite، لأنه هيشوف الشهادة اللي Burp Suite بيقدّمها ويعتبرها غير موثوقة.

إزاي بنتخطي Certificate Pinning؟

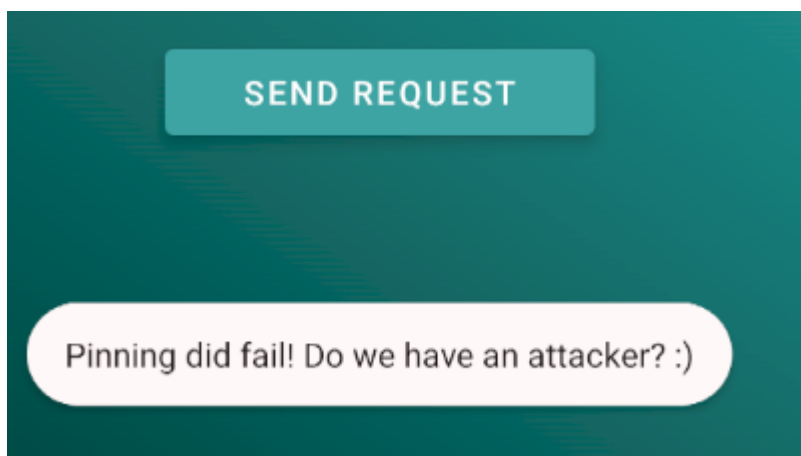
إزاي بنتخطى Certificate Pinning؟

1. تعديل التطبيق نفسه:

- بنفك التطبيق (ملف APK) باستخدام أدوات زي **APKTool**.
- بندور جوه الكود على الأجزاء اللي بتعمل تحقق للشهادات.
- بنعطل الأكواد دي.
- بنجمع التطبيق تاني ونوقعه عشان يشتغل.

2. استخدام أدوات زي **Frida**

هنا في المثال ده هو مش راضي بيعت request علشان بيقول ان في attacker في النص



فدلوقتى عاوزين ننشئ certificate ونحطها في التطبيق علشان بيعت request هنا domain : example.com:443

1-Create certificate for example.com:443

```
→ Downloads openssl s_client -showcerts -connect example.com:443
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert
Global Root G3
verify return:1
depth=1 C = US, O = DigiCert Inc, CN = DigiCert Global G3 TLS ECC SHA384
2020 CA1
verify return:1
depth=0 C = US, ST = California, L = Los Angeles, O = Internet Corporation
for Assigned Names and Numbers, CN = *.example.com
verify return:1
---
Certificate chain
 0 s:C = US, ST = California, L = Los Angeles, O = Internet Corporation for
Assigned Names and Numbers, CN = *.example.com
  i:C = US, O = DigiCert Inc, CN = DigiCert Global G3 TLS ECC SHA384 2020
```

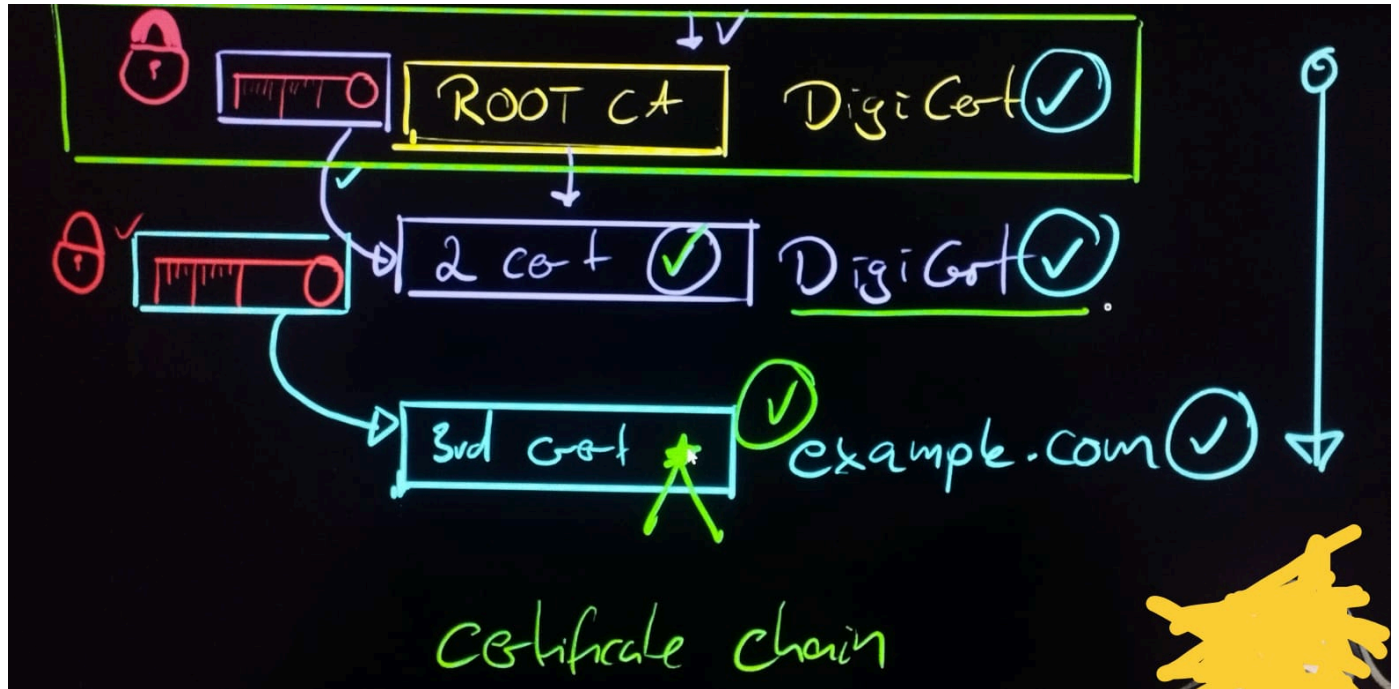

CA1

```
a:PKEY: id-ecPublicKey, 256 (bit); sigalg: ecdsa-with-SHA384
```

```
v:NotBefore: Jan 15 00:00:00 2025 GMT; NotAfter: Jan 15 23:59:59 2026 GMT
```

```
-----BEGIN CERTIFICATE-----
```

هنا بقي هو بينشئ 3 certificate اول واحدة دي خاصة ب digicert وببقي ليها private and public key بس هنا مينفعش نستخدمها ف digicert بينشئ شهادة تانية تكون وسيط بين digicert وشهادة اللي خاصة بالموقع اللي هي رقم 3 الشهادة اللي هي رقم تاني بيكون ليها برضه private and public key بسنستخدم بقي private key علشان نعمل sign للشهادة اللي هي رقم 3 اللي هي خاصة ب example.com



هنا اه دي certificate بتاعت example.com دي بتكون pem format يعني base64 and ascii

```

-----BEGIN CERTIFICATE-----
MIIFmzCCBSGgAwIBAgIQctiTuvposLf7ekBPBuyvmjAKBggqhkJOPQQDAzBZMQsw
CQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMTMwMQYDVQQDEypEaWdp
Q2VydCBHbG9iYWwgRzMGVEXtIEVDQyBTSEezODQgMjAyMCDQTEwHhcNMjUwMTE1
MDAwMDAwWhcNMjYwMTE1MjM1OTU5WjCBjjELMAkGA1UEBhMCVVMxEzARBgNVBAgT
CkNhbgGmb3JuaWExFDASBgNVBAcTC0xvcyBBbmdlbGVzMTEwOgYDVQQKEzNJbnRl
cm5ldCBDb3Jwb3JhdGlvbiBmb3IgQXNzaWduZWQgTmFtZXMGYw5kIE51bWJlcnMx
FjAUBgNVBAMMSouZXhbbXBsZS5jb20wWTATBgqhkJOPQIBBgqhkJOPQMBBwNC
AASaSJELWFsCmlqFKDIOIDmAMCH+plXDhsA4tiHklfnCPs8XrDThCg3wSQRjtMg
cXS9k490CQPOAjuw5GZzz6/uo4IDkzCCA48wHwYDVR0jBBgwFoAUiiPrnmvX+Tdd
+W0hOXaaowfeEKgwHQYDVR00BBYEFPDBajIN7NrH6o/NDW0ZElnRvnLtmCUGA1Ud
EQQeMByCDSouZXhbbXBsZS5jb22CC2V4YW1wbGUuY29tMD4GA1UdIAQ3MDUwMwYG
Z4EMAQICMCKwJwYIKwYBBQUHAgEWEg2h0dHA6Ly93d3cuZGlnaWNlcnQuY29tL0NQ
UzaOBgNVHQ8BAf8EBAMCA4gwHQYDVR0lBBYwFAYIKwYBBQUHAgEGCCsGAQUFBwMC
MIGfBgNVHR8EgZcwZQwSKBGoESGQmh0dHA6Ly9jcmwzLmRpZ2ljZXJ0LmNvbS9E
aWdpQ2VydEdsb2JhbEcZVExTRUNDU0hBMzg0MjAyMENBMS0yLmNybDBIoEagRIZC
aHR0cDovL2NybdQuZGlnaWNlcnQuY29tL0RpZ2ljZXJ0R2xvYmFsRzNUTFNFQ0NT
SEezODQyMDIwQ0ExLTiUy3JSMIGHBggrBgEFBQcBAQR7MHkwJAYIKwYBBQUHMAAGG
Gh0dHA6Ly9vY3NwLmRpZ2ljZXJ0LmNvbTBRRBggrBgEFBQcwAoZFaHR0cDovL2Nh
Y2VydHMuZGlnaWNlcnQuY29tL0RpZ2ljZXJ0R2xvYmFsRzNUTFNFQ0NTSEezODQy
MDIwQ0ExLTiUy3J0MAwGA1UdEwEB/wQCMAAwggF7BgorBgEEAdZ5AgQCBIIBawSC
AWcBZQB0AA5XLlZzrqk+MxssmQez95Dfm8I9cTIL3SGpJaxhxU4hAAABlGd6v8cA
AAQDAEUwQwIfJBcPWkx80ik7uLYW6OGvNYvJ4NmOR2RXc9uviFPH6QIgUtuuUenH
ITSUNWJffBBRq31tUGi7ZDTSrrM0f4z1Va4AdQBkEcRspBLsp4kcogIuALyrTygH
1B41J6vq/tUDyX3N8AAAAZRnesAFAAAEAwBGMEQCIHCu6NgHhV1Qvif/G7BHq7ci
MGH8jdch/xy4LzrYlesXAIByMFMvDhGg4sYm1MsrDGVedcwpE4eN0RuZcFGmWxwJ
cgB2AEmcm2neHXzs/DbezYdkprhbrwqHgBnRVVL76esp3fjDAAABlGd6wBkAAAQD
AEcwRQIgaFh67yEQ2lwgm3X16n2iWjEQFII2b2fp0NtBVibZVWwCIQD5psqjXDYs
IEb1hyh0S8bBN304u2sA9zisKILYjZg8wjAKBggqhkJOPQQDAwNoADBLAJEA+aaC
RlPbb+VY+u4avPyaG7fvUDJqN8KwlrXD4XptT7QL+D03+BA/FUEo3dD1iz37AjBk
Y3jhsuLAW7pWsDbtX/Qwxp6kNsK4jh1/RjvV/260sxQwM/GM7t0+T0uP2L+Y12U=
-----END CERTIFICATE-----

```

هنا الفرق بين certificate الذي يرجعها server وبين certificate الذي يبنئها attacker وان مينعش الهادة التي يبنئها attacker هو ستخدمها علشان يعمل بيها sign علشان الشهادة دي الصلا بيبكون معلوها لا hash ومتخزنة عند client ف attacker الشهادة التي هي عملها هتكون متغيرة فالتالي hash بتاعها هيكون تغير ف client هيرفضها

```

→ certfcate openssl x509 -in cert.pem -fingerprint -inform pem -md5 -noout
md5 Fingerprint=C3:39:79:FF:8B:C1:9A:94:82:0D:68:04:B3:68:18:81

```

how hashed has been generated

Certificate Fingerprints

```

SHA1:          31 0D B7 AF 4B 2B C9 04 0C 83 44 70 1A CA 08 D0 C6 93 81 E3
MD5:           C3 39 79 FF 8B C1 9A 94 82 0D 68 04 B3 68 18 81

```

sha1

```

→ certfcate openssl x509 -in cert.pem -fingerprint -inform pem -noout
SHA1 Fingerprint=31:0D:B7:AF:4B:2B:C9:04:0C:83:44:70:1A:CA:08:D0:C6:93:81:E3

```

md5

```
→ certifcate openssl x509 -in cert.pem -fingerprint -inform pem -md5 -noout  
md5 Fingerprint=C3:39:79:FF:8B:C1:9A:94:82:0D:68:04:B3:68:18:81
```

how to get public key from certificate

```
→ certifcate openssl x509 -in cert.pem -inform pem -pubkey -noout  
-----BEGIN PUBLIC KEY-----  
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEmkiXhC1hbAjjJahSgyDiA5gDAh/qZ  
Vw4bAOLYh5JX5wj7PF6w04QoN8EkeY7TIHF0vZOPTgkDzgI7sORmc8+v7g==  
-----END PUBLIC KEY-----  
→ certifcate
```