2. Bypass SSL Pinning và Root Checker

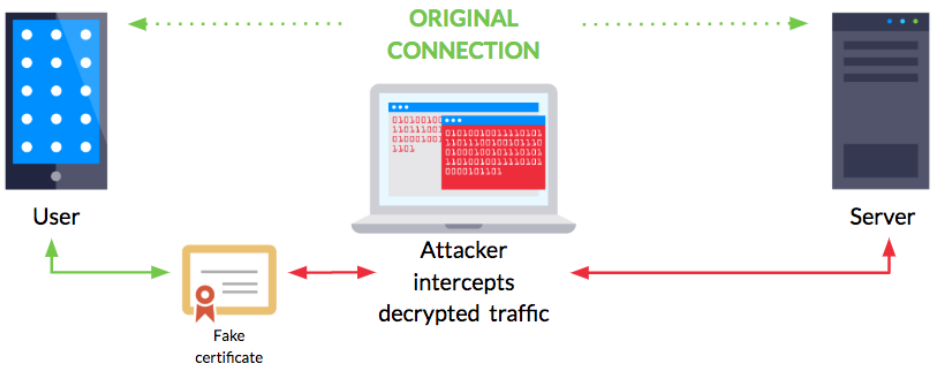
Harry

Bypass SSL Pinning và Root Checker

Sau khi chúng ta đã có máy ảo Android thông qua Genymotion và thực hiện được việc bắt các traffic bằng Burpsuite, trò chơi lúc này mới bắt đầu. Bạn bắt đầu hào hứng tải ngay một ứng dụng trên Google Play về để phân tích.

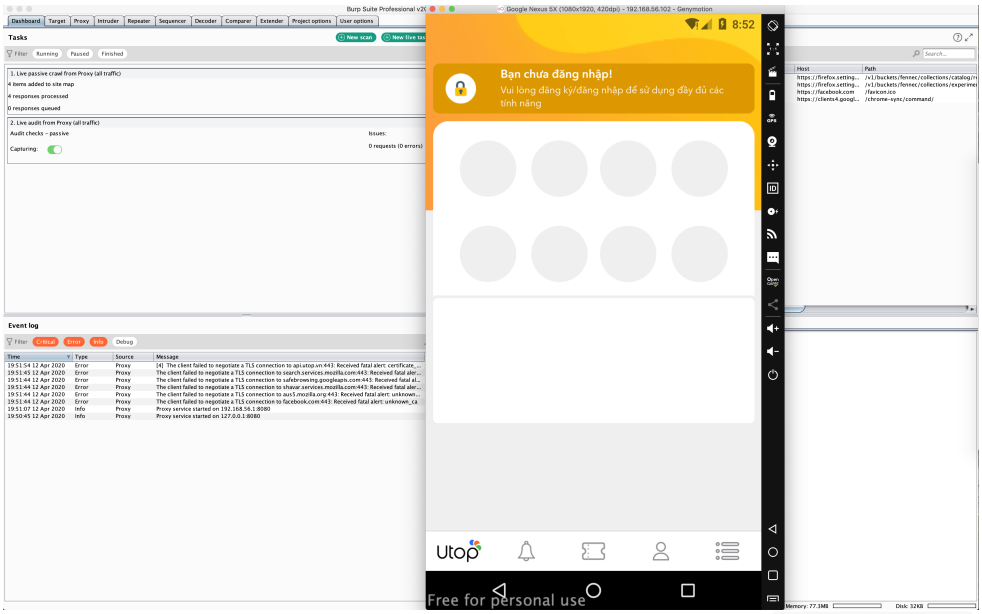
Mở ứng dụng lên và trên màn hình HTTP History của Burpsuite vẫn im bặt như tờ, không có một HTTPS Request / Response nào được ghi nhận, hoặc ứng dụng của bạn vẫn nằm yên như một chú mèo lười, các Button, Activity đều không phản hồi cho dù các ngón tay cực lực nhấn và lướt trên màn hình.

Lúc này khả năng cao ứng dụng của bạn đã thiết lập một cơ chế gọi là SSL Pinning. SSL Pining là một cách mà lập trình viên đã triển khai trong mã nguồn, và mong muốn chỉ định ứng dụng mà bạn vừa tải về chỉ được phép gửi và nhận Request tới một Server nhất định mà thôi (Ở đây là API của ứng dụng).



Nhưng do chúng ta sử dụng một Burpsuite Proxy đứng giữa, lúc này ứng dụng đã nhận thấy có một cái gì đó sai sai ở đây và ngưng việc gửi và nhận dữ liệu.

Trên Burpsuite Proxy, bạn sẽ nhận thấy dòng lỗi `1586695914853 Error Proxy [4] The client failed to negotiate a TLS connection to api.xx.xx:443: Received fatal alert: certificate_unknown` Nếu thấy lỗi này xuất hiện trong Burpsuite, thì ứng dụng này có SSL Pinning rồi.



Như hình trên bạn thấy, ứng dụng gửi các Request lên máy chủ `api.xx.xx:443` để lấy dữ liệu hiển thị, nhưng do chúng ta đang sử dụng một fake certificate của Burpsuite mà không được khai báo trong ứng dụng nên các Request này sẽ không thể gửi đi.

Vậy làm thế nào để xử lý việc này một cách gọn nhẹ nhất? Cách mà mình hay làm đó là sử dụng một vài Frida-scripts để Patch hoặc Overwrite lại các phương pháp mà lập trình viên họ hay sử dụng trong việc kiểm tra SSL Pinning.

Cài đặt Frida

Frida là gì thì chắc nói tới ở bài sau đi ha, còn giờ bắt tay vào cài đặt cho nhanh để xử lý cho gọn.

```
$ pip install frida
$ pip install objection
$ pip install frida-tools
```

Đầu tiên cần xác định kiến trúc chip của điện thoại đang sử dụng, sau đó lựa chọn Frida-Server phiên bản cho phù hợp.

```
$ adb shell getprop ro.product.cpu.abi
x86
```

Đối với máy ảo Android chạy trên Genymotion, thì cần tải bản frida-server `x86` ở đường link sau https://github.com/frida/frida/releases/download/12.8.20/frida-server-12.8.20-android-x86.xz

```
$ wget
https://github.com/frida/frida/releases/download/12.8.20/frida-
server-12.8.20-android-x86.xz
$ unxz frida-server-12.8.20-android-x86.xz
$ mv frida-server-12.8.20-android-x86 frida-server
$ adb shell "mount -o rw,remount /system"
$ adb push frida-server /system/bin/frida-server
$ adb shell "chmod 755 /system/bin/frida-server"
```

Tập lệnh trên dùng để download `frida-server-12.8.20-android-x86.xz`, sau đó giải nén và đổi tên thành `frida-server`. Remount lại `/system` của thiết bị Android cho phép quyền đọc và ghi, sau đó sử dụng `adb push` để chép `frida-server` lên thiết bị Android của bạn tại đường dẫn `/system/bin/frida-server`. Để thực thi được `frida-server` bạn cần phải cấp quyền execute bằng lệnh `chmod 755`

Sau khi đã cài đặt được `frida-server` lên máy Android, bạn cần chạy nó để lắng nghe và chờ đợi những điều tốt đẹp từ `frida` client.

Trên Android bạn thực thi lệnh dưới đây để `frida-server` chạy ngầm.

```
$ adb shell
vbox86p:/ # frida-server &
[1] 3226
```

Frida-server đã chạy và có PID là 3226

Trên máy tính, chạy `frida-ps` client để thực hiện kiểm tra các kết nối xem hoạt động tốt không, tham số `-U` (kết nối tới thiết bị Android thông qua giao thức USB), hoặc `-D <id>` (kết nối tới thiết bị thông qua deviceid). Danh sách các thiết bị đang kết nối tới máy tính thông qua `adb` được kiểm tra bằng `adb devices` hoặc `frida-ls-devices`

```
$ frida-ls-devices
Id                  Type    Name
------------------  ------  ------------
```

```
local                 local   Local System
192.168.56.102:5555   usb     Google
tcp                   remote  Local TCP
```

Ở đây mình sẽ thực hiện việc kiểm tra các tiến trình đang hoạt động trên máy Android.

```
$ frida-ps -U
 PID  Name
----  -------------------------------------------------
 138  adbd
 989  android.process.acore
 259  audioserver
 249  batteryd
 260  cameraserver
2963  com.android.cellbroadcastreceiver
2623  com.android.chrome
2802  com.android.chrome:webview_service
2302  com.android.deskclock
 615  com.android.inputmethod.latin
1156  com.android.launcher3
2903  com.android.packageinstaller
 760  com.android.phone
2522  com.android.printspooler
1414  com.android.providers.calendar
 635  com.android.systemui
1795  com.android.vending
2503  com.android.vending:download_service
1066  com.genymotion.genyd
1052  com.genymotion.systempatcher
 941  com.google.android.ext.services
1205  com.google.android.gms
 741  com.google.android.gms.persistent
2380  com.google.android.gms.ui
2173  com.google.android.gms.unstable
1036  com.google.android.googlequicksearchbox:interactor
1004  com.google.android.googlequicksearchbox:search
 967  com.google.process.gapps
2015  com.utop
 117  debuggerd
 127  debuggerd:signaller
 250  diskiod
```

OK vậy chúng ta đã cài đặt xong `frida-server` và dễ dàng thao tác được với thiết bị thông qua `frida` client

Bypass SSL Pining

Sau khi xác định ứng dụng bị SSL Pining, chúng ta cần thực hiện việc Bypass nó. Ở đây mình sử dụng các Script sau dùng để xác định các phương pháp SSL Pining mà lập trình viên họ sử dụng và tự động Patch hoặc Overwrite lại.

Với ứng dụng ở đây là `com.utop`

```
frida -U --codeshare akabe1/frida-multiple-unpinning -f com.utop -
-no-pause
```

Hoặc

```
frida -U --codeshare pcipolloni/universal-android-ssl-pinning-
bypass-with-frida -f com.utop --no-pause
```

Lệnh trên sẽ sử dụng các script được chia sẻ trên codeshare (một repo lưu trữ Script được chia sẻ với cộng đồng của frida). Đồng thời ứng dụng `com.utop` sẽ được khởi tạo và chạy bên trong môi trường của `frida`, `--no-pause` là tham số để `frida` tự động khởi tạo main thread.

- https://codeshare.frida.re/@akabe1/frida-multiple-unpinning/
- https://codeshare.frida.re/@pcipolloni/universal-android-ssl-pinning-bypass-with-frida/

Nếu các kết nối mạng bị hạn chế, bạn nên Download các script này về để sử dụng Offline, và sử dụng tham số `-l`, ví dụ `frida -l frida-multiple-unpinning.js`

```
$ frida -U --codeshare akabe1/frida-multiple-unpinning -f com.utop
--no-pause
     ____
    / _  |   Frida 12.8.20 - A world-class dynamic instrumentation
toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://www.frida.re/docs/home/
```

```
Spawned com.utop. Resuming main thread!
[Google::com.utop]->
======
[#] Android Bypass for various Certificate Pinning methods [#]
======
[-] OkHTTPv3 pinner not found
[-] Trustkit pinner not found
[-] Appcelerator PinningTrustManager pinner not found
[-] OpenSSLEngineSocketImpl Conscrypt pinner not found
[-] OpenSSLSocketImpl Apache Harmony pinner not found
[-] PhoneGap sslCertificateChecker pinner not found
[-] IBM MobileFirst pinTrustedCertificatePublicKey pinner not
found
[-] IBM WorkLight HostNameVerifierWithCertificatePinning pinner
not found
[-] CWAC-Netsecurity CertPinManager pinner not found
[-] Worklight Androidgap WLCertificatePinningPlugin pinner not
found
[-] Netty FingerprintTrustManagerFactory pinner not found
[-] Squareup CertificatePinner pinner not found
[-] Squareup OkHostnameVerifier pinner not found
[-] Apache Cordova WebViewClient pinner not found
[-] Boye AbstractVerifier pinner not found
[+] Bypassing OpenSSLSocketImpl Conscrypt
[+] Bypassing Trustmanager (Android < 7) request
[+] Bypassing Trustmanager (Android < 7) request
[+] Bypassing OpenSSLSocketImpl Conscrypt
[+] Bypassing OpenSSLSocketImpl Conscrypt
[+] Bypassing OpenSSLSocketImpl Conscrypt
[Google::com.utop]-> [+] Bypassing Trustmanager (Android < 7)
request
[+] Bypassing OpenSSLSocketImpl Conscrypt
[+] Bypassing OpenSSLSocketImpl Conscrypt
[+] Bypassing OkHTTPv3 {1}: api.xxx.xxx
```

Script đã phát hiện ra ứng dụng này đang sử dụng ra phương pháp SSL Pining cho `Trustmanager`, `OkHTTPv3` và áp dụng việc re-implement lại các Method này và luôn `return True` để bypass các cơ chế kiểm tra này.

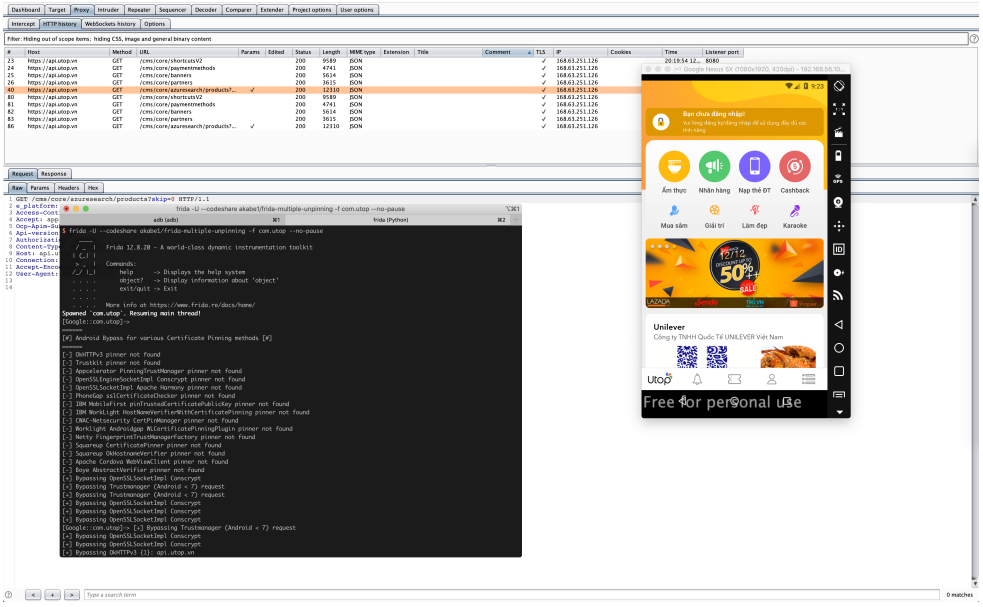
```
...
      // OkHTTPv3 (double bypass)
        try {
            var okhttp3_Activity =
Java.use('okhttp3.CertificatePinner');
            okhttp3_Activity.check.overload('java.lang.String',
'java.util.List').implementation = function (str) {
```

```
                console.log('[+] Bypassing OkHTTPv3 {1}: ' + str);
                return true;
            };
            // This method of CertificatePinner.check could be
found in some old Android app
            okhttp3_Activity.check.overload('java.lang.String',
'java.security.cert.Certificate').implementation = function (str)
{
                console.log('[+] Bypassing OkHTTPv3 {2}: ' + str);
                return true;
            };

        } catch (err) {
            console.log('[-] OkHTTPv3 pinner not found');
            //console.log(err);
        }
....
```

```
...
     var X509TrustManager =
Java.use('javax.net.ssl.X509TrustManager');
        var SSLContext = Java.use('javax.net.ssl.SSLContext');


        // TrustManager (Android < 7)
        var TrustManager = Java.registerClass({
            // Implement a custom TrustManager
            name: 'dev.asd.test.TrustManager',
            implements: [X509TrustManager],
            methods: {
                checkClientTrusted: function (chain, authType) {},
                checkServerTrusted: function (chain, authType) {},
                getAcceptedIssuers: function () {return []; }
            }
        });

        // Prepare the TrustManager array to pass to
SSLContext.init()
        var TrustManagers = [TrustManager.$new()];
        // Get a handle on the init() on the SSLContext class
        var SSLContext_init = SSLContext.init.overload(
            '[Ljavax.net.ssl.KeyManager;',
'[Ljavax.net.ssl.TrustManager;', 'java.security.SecureRandom');
        try {
            // Override the init method, specifying the custom
TrustManager
            SSLContext_init.implementation = function(keyManager,
```

```
trustManager, secureRandom) {
                console.log('[+] Bypassing Trustmanager (Android <
7) request');
                SSLContext_init.call(this, keyManager,
TrustManagers, secureRandom);
            };

        } catch (err) {
            console.log('[-] TrustManager (Android < 7) pinner not
found');
            //console.log(err);
        }
```

Vậy chúng ta đã thực hiện Bypass cơ chế SSL Pinning thành công, ứng dụng hoạt động ngon lành và các HTTPS Request đã được can thiệp.



Bypass Root Checker

Một số ứng dụng hoạt động trong các ngành tài chính, ngân hàng thì có thể sẽ áp dụng các phương pháp kiểm tra xem máy của người dùng có bị Jailbreak hoặc Rooted không. Nếu phát hiện ra, thì ứng dụng này sẽ không được khởi chạy để đảm bảo tính toàn vẹn cho ứng dụng và dữ liệu của người dùng.

Nhưng với một Pentester, việc sử dụng các máy bị Rooted sẽ cho chúng ta một môi trường lý tưởng hơn, và chúng ta phải đối mặt với việc làm cách nào để vượt qua cơ chế kiểm tra này?

Vẫn sử dụng Frida nhé, và sử dụng script sau.

```
frida --codeshare dzonerzy/fridantiroot -U -f <app_name> --no-
pause
```

Có nhiều cơ chế để phát hiện và Bypass, tùy từng trường hợp mà ứng dụng sử dụng. Nên đôi khi các bạn chạy Script mà nó không hoạt động đúng theo mục đích thì việc tự viết một Script để Bypass là chuyện bình thường. Mình sẽ chia sẻ cụ thể hơn ở các bài sau nhé.