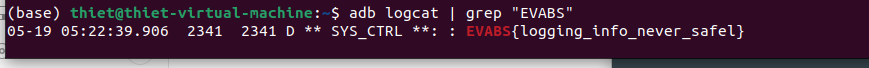
D.1 EVABS Tập tin APK: EVABSv5.apk

* Challenge 1:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động

* Truy cập vào challenge 1 thì ta thấy màn hình có 2 nút để tương tác
* Khi ta ấn nút log the key thì nó sẽ xuất cho ta 1 log và trong đó sẽ chứa key
* Khi đọc hint thì em biết là ta sẽ dùng lệnh “adb logcat” để có thể xem log
* Nhưng khi dùng lệnh không thì sẽ ra rất nhiều log nên không thể xem hết
* Em sẽ dùng lệnh grep để lọc các từ khóa liên quan đến “EVABS” là định dạng của flag



* Flag: EVABS{logging\_info\_never\_safel}
* Challenge 2:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Hệ điều hành

Mô tả được tạo tự động

* Ở challenge này thì ta cần tìm file tài nguyên assets của 1 tệp tin apk
* Khi tìm hiểu thì biết là ta có thể tìm được bằng cách đổi tên của file .apk thành .zip sau đó giải nén ra thì ta sẽ có được các thư mục

Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

* Sau khi đổi tên và giải nén ta sẽ có 1 thư mục là EVABSv5

Ảnh có chứa ảnh chụp màn hình, văn bản

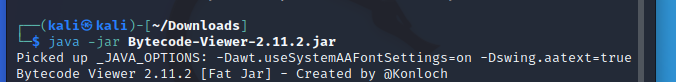
Mô tả được tạo tự động

* Vào thư mục assets thì có 1 file tên là secret và trong đó có chứa flag
* Flag: EVABS{fil3s\_!n\_ass3ts\_ar3\_eas!ly\_hackabl3}
* Challenge 3

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Hint cho ta biết là flag đang được giấu ở trong 1 file android thường chứa những string
* Để xem được string trong file android thì ra có nhiều cách
* Ở đây ta dùng bytecode-viewer để xem



* Sử dụng lệnh này để có thể mở bytecode-viewer

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, Phông chữ

Mô tả được tạo tự động

* Ta kiếm file tên là “strings.xml” trong đây sẽ chứa những chuỗi có trong file apk

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Flag: EVABS{saf3ly\_st0red\_in\_Strings?}
* Challenge 4:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Hệ điều hành

Mô tả được tạo tự động

* Ở challenge này thì nó chỉ bảo ta tìm kiếm trong thư mục “res”

Ảnh có chứa văn bản, phần mềm, ảnh chụp màn hình, số

Mô tả được tạo tự động

* Lướt qua các thư mục thì trong res/raw/link.txt chứ flag trong challenge này
* Flag: EVABS{th!s\_plac3\_is\_n0t\_as\_s3cur3\_as\_it\_l00ks}
* Challenge 5

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Ở challenge này thì yêu cầu ta tìm key-value
* Ở trong android key-value pairs được lưu trữ theo cơ chế SharedPreferences sẽ được lưu trữ trong một tập tin văn bản đơn giản trên bộ nhớ trong thiết bị

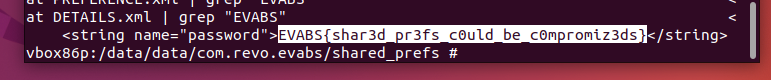


* Ta dùng adb để mở shell của thiết bị lên
* Sau đó truy cập vào đường dẫn “/data/data/com.revo.evabs/shared\_prefs”
* Trong đó sẽ chứa 2 file

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, Đồ họa

Mô tả được tạo tự động

* Ta dùng lệnh “cat DETAIL.xml | grep EVABS” để lọc ra các ký tự có chứa flag



* Flag: EVABS{shar3d\_pr3fs\_c0uld\_be\_c0mpromiz3ds}
* Challenge 6:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Hệ điều hành

Mô tả được tạo tự động

* Ở đây ta cần truy cập vào database của ứng dụng



* Ta vào đường dẫn “/data/data/com.revo.evabs/databases” sau khi ấn fetch creds thì nó sẽ xuất hiện database của ứng dụng



* Sử dụng sqlite3 để có thể vào databases

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

* Sau khi xem tables thì ta thấy có 2 bảng ta xem thông tin trong bảng CREDS trước

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, số

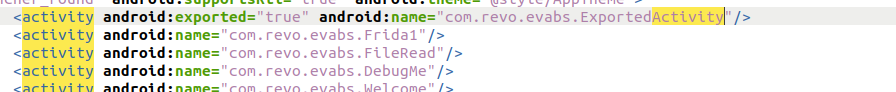
Mô tả được tạo tự động

* Ta thấy được flag
* Flag: EVABS{sqlite\_is\_not\_safe}
* Challenge 7:

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Trong ứng dụng Android có khái niệm activity. Activity trong Android là nơi diễn ra mọi hoạt động tương tác với người dùng, bởi vì tất cả các màn hình ứng dụng đều phải được “đính” trên một Activity.
* Thông tin về các activity được lưu trong file AndroidManifest.xml, xuất hiện trong các thẻ <activity ….. />. Trong thẻ này có 1 thuộc tính quan trọng là android:exported. Nếu thuộc tính này có giá trị true thì activity đó có thể bị kích hoạt bởi các ứng dụng khác.



* Ta thấy có 1 export activity tên là “com.revo.evabs.ExportedActivity”
* Để có thể gọi activity thì ta vào adb shell và dùng lệnh : “am start -n <package-name>/<activity-name>”

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Challenge 8

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

* Ở challenge này yêu cầu ta phải tìm key trong source code nhưng nó bị tách thành 2 phần

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Mô tả được tạo tự động

* Ta dùng jadx để reverse file apk về lại thành file java để xem code

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Xem qua các file thì ta thấy có file tên là Decode.java
* Trong đây chứa 1 chuỗi base64 ta thử decode

Ảnh có chứa văn bản, Phông chữ, phần mềm, ảnh chụp màn hình

Mô tả được tạo tự động

* Sau khi decode ta có được flag
* Sau khi nhập checkflag thì nó đúng nên em nghĩ đây là full flag
* Flag: EVABS{nev3r\_st0re\_s3ns!tiv3\_data\_1n\_7h3\_s0urcec0de}
* Challenge 9

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Ở challenge này ta cần bật được SYS\_CTRL\_CODE: LAB\_ON
* Nhưng ở đây thì ta SYS\_CTRL: ACCESS\_DENIED không thể turn on được
* Vì vậy ta cần chỉnh sửa file smali và recompile lại

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* Xem qua các file thì ta thấy có 1 file là smaliinject.java
* Đọc qua thì chương trình định nghĩa 1 biến là SIGNAL và gán là “LAB\_OFF”
* Đoạn code ở dưới kiểm tra điều kiện nếu biến SIGNAL là ON thì sẽ cấp quyền cho ta và gọi đến textView.setText và cho ta flag
* Còn không thì sẽ không cấp quyền
* Vậy ở đây ta cần đổi giá trị của biến SIGNAL để có thể có quyền

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* Ta xem trong file smaliinject.smali có 1 dòng để gán v0 là “LAB\_OFF” giờ ta đổi thành “LAB\_ON”

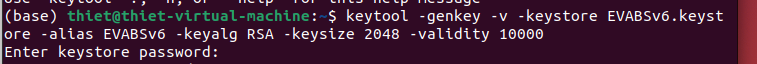
Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Sau đó ta build lại



* Tạo key

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Ký vào ứng dụng sau đó install nó

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

* Ta thấy SYS\_CTRL\_CODE: LAB\_ON và khi ấn turn on ta đã được cấp quyền và có được flag
* Challenge 10

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Ở challenge này ta cần dùng burpsuite để có thể bắt các gói tin trao đổi giữa điện thoại và network
* Challenge 11

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Mục tiêu của challenge này là bypass được quyền của ứng dụng
* Để có thể thực hiện 1 custom action đề bypass
* Nó yêu cầu ta nhập 1 chuỗi để có thể exploit được hệ thống
* Khi nhập 1 chuỗi ngẫu nhiên thì nó báo lỗi

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

* Ta đọc code chương trình trong file customaccrss.java
* Nó sẽ kiểm tra xem ta nhập vào có đúng với chuỗi “cust0m\_p3rm” hay không
* Nếu khớp, một thông báo hiển thị lên màn hình và một Intent được gửi đi để chuyển đến hoạt động khác với thông điệp chứa chuỗi kết quả từ phương thức stringFromJNI(). Nếu không khớp, một thông báo khác hiển thị lên màn hình.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thiết kế

Mô tả được tạo tự động

* Kết quả khi ta nhập đúng chuỗi
* Vậy thì luồng chương trình sẽ là khi ta nhập đúng chuỗi vào nó sẽ gửi flag thông qua 1 intent có hành động com.revo.evabs.action.SENSOR\_KEY
* Nhưng khi ta nhập đúng chuỗi thì nó chỉ nhảy ra 1 popup thông báo mà không có thực hiện như dự định
* Ý tưởng ở đây ta sẽ hook hàm putExtra() để nó có thể in ra flag cho ta

Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

* Ta dùng lệnh “frida-ps -U” sau khi cài đặt frida server vào điện thoại để có thể hiển thị các process trong điện thoại
* ở đây ta thấy có 1 process là “EVABS”

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

* Đây là code để có thể hook được hàm putExtra()
* Mã JavaScript này thực hiện các bước sau:

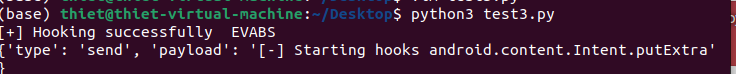
+ Sử dụng hàm Java.perform() để thực hiện mã JavaScript trong môi trường Java.

+ Gửi một thông điệp đến hàm onMessage() để thông báo rằng việc móc hàm đang được bắt đầu.

+ Sử dụng Java.use() để lấy đối tượng Java của lớp "android.content.Intent" và gán cho biến intent.

+ Sử dụng hàm overload() để chỉ định phiên bản chính xác của hàm putExtra() nhận hai tham số là chuỗi.

+ Ghi đè (override) hàm putExtra() để thay đổi hành vi của nó. Trong trường hợp này, khi hàm putExtra() được gọi, một thông điệp chứa cờ (flag) sẽ được gửi đến -hàm onMessage() để in ra.



* Sau khi hook thành công thì trên điện thoại nó sẽ mở ứng dụng và level 11
* Sau đó ta cần nhập chuỗi xác thực “cust0m\_p3rm” như ở trên

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, thông tin

Mô tả được tạo tự động

* Sau đó hàm sẽ được hook vào và in ra flag cho ta
* Flag: EVABS{always\_ver1fy\_packag3sa}
* Challenge 12

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, thiết kế

Mô tả được tạo tự động

* Nó có tiêu đề là instrument và hint cho ta biết đó là dynamic instrumentation
* Theo tìm hiểu thì đó là 1 kỹ thuật cho phép ta can thiệp và thay đổi hoạt động của 1 ứng dụng trong thời gian runtime
* Nó cho phép mở rộng tính năng mà không thay đổi mã nguồn
* Khi một ứng dụng android được dynamic instrumentated, các mã can thiệp sẽ được chèn vào quá trình chạy của ứng dụng, cho phép thực hiện các thay đổi và tương tác với bên ngoài
* Khi ta ấn vào nút map area thì nó hiện ra các tọa độ và hiện ra 1 thông báo không tìm thấy có vẻ như nó không thay đổi dù đã ấn nhiều lần
* Đã thử bật định vị cũng không có gì khác

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Xem file frida1.java thì ta thấy các tọa độ đã được set cứng

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

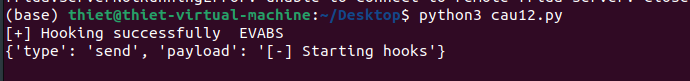
Mô tả được tạo tự động

* Đây là code xử lý khi người dùng nhấn vào button
* Ta thấy nó sẽ tính toán giá trị x = a \*b sau đó nó sẽ tạo 1 giá trị ngẫu nhiên rồi gán vào biến netInt. Giá trị ngẫu nhiên trong khoảng 70
* Nếu giá trị biến x lớn hơn nextInt + 150 thì nó in ra thông báo và hiển thị log trong đây ta có thể đoán sẽ chứa flag hay thông tin gì đó
* Còn không thì ngược lại nó sẽ in ra thông báo “Co-ordinates Not Found!”

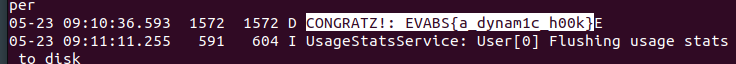
Ảnh có chứa văn bản, ảnh chụp màn hình

Mô tả được tạo tự động

* Ta cũng sẽ sử dụng frida để có thể hook
* Ý tưởng ở đây ta sẽ ghi đè giá trị trả về của nextInt để trả về 1 giá trị nhỏ sau đó + 150 thì kết quả sẽ nhỏ hơn giá trị của x = a\*b thì khi đó nó sẽ thực hiện điều kiện và trả về log có chứa flag cho ta



* Bắt đầu quá trình hooking



* Sử dụng abd logcat để xem được log và thấy thấy nó đã trả về flag cho ta
* Flag: EVABS{a\_dynam1c\_h00k}

Challenges 2 Hoàn thành 5 challenges

* Level 1

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Ở challenge này ta cần nhập vào 1 chuỗi gì đó sau khi ấn nít thì nó sẽ có log của ứng dụng và trong đó chứa flag của ta

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Flag: picoCTF{a.moose.once.bit.my.sister}
* Level 2

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

* Ở level này thì ta cần nhập vào 1 chuỗi đúng thì nó sẽ trả về flag cho ta
* Level này yêu cầu ta không được brute force

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Hệ điều hành

Mô tả được tạo tự động

* Sử dụng apktool để decompile file apk
* ở trong two/res/values có file strings.xml nó sẽ chứa các chuỗi trong chương trình của ta

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* có 1 chuỗi đáng ngờ được set là password

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

* Khi ta nhập chuỗi vào thì nó trả cho ta flag
* Flag: picoCTF{pining.for.the.fjords}
* Level 3

Ảnh có chứa văn bản, Phông chữ, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

* Ở đây ta cần tìm vào các file smali của chương trình để có thể có được flag

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, đa phương tiện

Mô tả được tạo tự động

* Ta sử dụng công cụ jadx để có thể chuyển từ file apk về thành file java cho dễ đọc hơn file smali

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Trong đường dẫn /sources/com/hellocmu/picoctf/ có file flagstaffhill.java khá đáng ngờ và mở lên thì ta được đoạn code java như trên
* Có vẻ như chương trình làm ta rối bằng cách thêm các phép tính để có thể lấy index trong mảng sau đó lấy giá trị tại index đó nối lại với nhau bằng lệnh concat để có thể ra password
* Sau khi có password nó sẽ so sánh giá trị đó với giá trị người dùng nhập. Nếu trùng nhau thì sẽ trả về flag còn không thì sẽ in ra “NOPE”

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Đây là các giá trị sau khi được tính ra

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Ta đưa các chỉ số đó vào và in ra password

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Đây là password sau khi tính được

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, màu trắng

Mô tả được tạo tự động

* Ta nhập vào chương trình và nó trả về flag cho ta
* Flag: picoCTF{what.is.your.favourite.colour}
* Level 4

Ảnh có chứa văn bản, hàng, ảnh chụp màn hình

Mô tả được tạo tự động

* Ở challenge này dựa vào gợi ý thì ta cần phải sửa lại code chương trình và compile lại

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* Ta dùng jadx để decompile chương trình
* Trong /sources/com/hellocmu/picoctf/ có file flagstaffHill.java
* Đoạn code trên có thể hiểu để có thể lấy được flag thì trong hàm getflag() biến flag được gán bằng hàm nope() trả về “don’t wanna”
* Ta cần thay đổi để biến flag được gán với 1 hàm yep() trong đó nó gọi hàm cilantro()

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Hệ điều hành

Mô tả được tạo tự động

* Ta sử dụng apktool để decompile thành file smali để chỉnh sửa

Ảnh có chứa văn bản, Phông chữ, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

* Trong đường dẫn four/smali/com/hellocmu/picoctf/ có file FlagstaffHill.smali trong đây chứa luồng chương trình của ta
* Hình trên là đoạn mà hàm getflag() gọi tới hàm nope()
* Vì hàm nope() và yep() có cùng tham số nên ta chỉ cần đổi từ hàm nope() thành hàm yep() là được

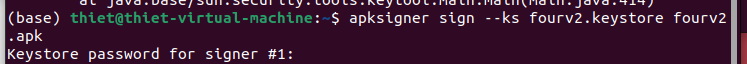
Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

* Sau khi sửa thành yep()
* Giờ ta build lại và ký sau đó cài đặt là được

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động



* Tạo ra file fourv2.apk sau đó tạo key rồi dùng key đó để ký lên ứng dụng

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Sử dụng adb install để cài đặt ứng dụng mới lên(cần xóa ứng dụng cũ đi trước)

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

* Giờ ta nhập bất cứ gì nó cũng trả lại flag cho ta
* Flag: picoCTF{tis.but.a.scratch}
* Level 5

Ảnh có chứa văn bản, hàng, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động

* Challenge này không cho ta gợi ý gì nhiều

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* Sau khi decompile thì ta thấy nó khá giống với level 3
* Trong phương thức này, bốn biến kiểu StringBuilder được khởi tạo với các giá trị ban đầu đều là "aaa". Sau đó, các ký tự trong các biến này được thay đổi bằng cách thêm một giá trị cố định vào từng ký tự. Cụ thể, giá trị của từng ký tự được thay đổi bằng cách cộng với một giá trị cố định khác nhau cho mỗi biến.
* Sau đó, chuỗi password được tạo ra bằng cách nối các giá trị của các biến ace, jack, queen và king lại với nhau. Cuối cùng, phương thức kiểm tra xem giá trị đầu vào của người dùng (input) có khớp với giá trị của chuỗi password hay không. Nếu giá trị đầu vào khớp với password, phương thức trả về chuỗi "call it". Nếu không, phương thức trả về chuỗi "NOPE"
* Ta cũng viết code để lấy được password

Ảnh có chứa văn bản, Phần mềm đa phương tiện, Phần mềm đồ họa, phần mềm

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

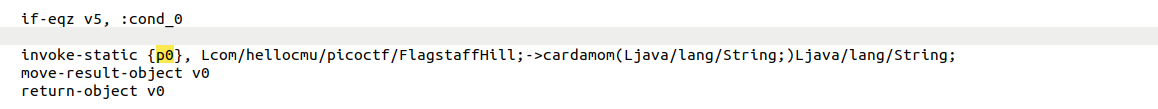
Mô tả được tạo tự động

* Ta có được password
* Nhưng level này đâu dễ đến thế
* Nếu nhìn kỹ lại thì nếu như khớp pass thì nó sẽ trả về chuỗi “call it” chứ không phải flag của ta
* ở đây ta cần chỉnh sửa để nó gọi được đến hàm cardamon() được định nghĩa ở trên

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biên lai

Mô tả được tạo tự động

* Đây là dòng smali định nghĩa nếu biến bằng password thì sẽ là cond\_0 và nó gán v5 là “call it” sau đó return
* Giờ ta cần chỉnh lại



* Ở level trước ta đã biết cách gọi hàm yep() sau đó trả về giá trị
* Ở đây ta chỉ cần coppy lại phần gọi hàm yep() và thay bằng hàm cardamon() là được
* Sau đó build lại ký vào cài đặt

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, hàng

Mô tả được tạo tự động

* Sau khi cài ta nhập password và có được flag
* Flag: picoCTF{not.particularly.silly}