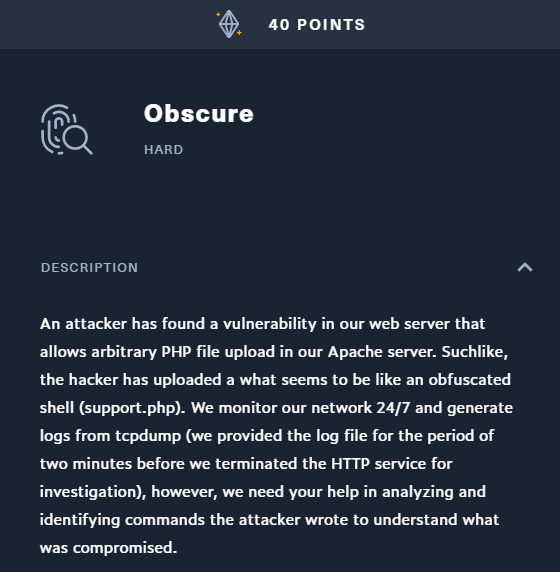
HACK THE BOX WRITE UP– OBSCURE



Với tôi, đây là 1 challenge khá là thú vị. Mục tiêu forensic là phân tích để thu được bất cứ thông tin nào về một sự cố đã xảy ra để hiểu được nguyên nhân gốc là do đâu hoặc tác động nào đã gây ra sự cố này. Các bạn cũng sẽ thích challeng này như tôi. Let’s go!!!

Challeng Introduction

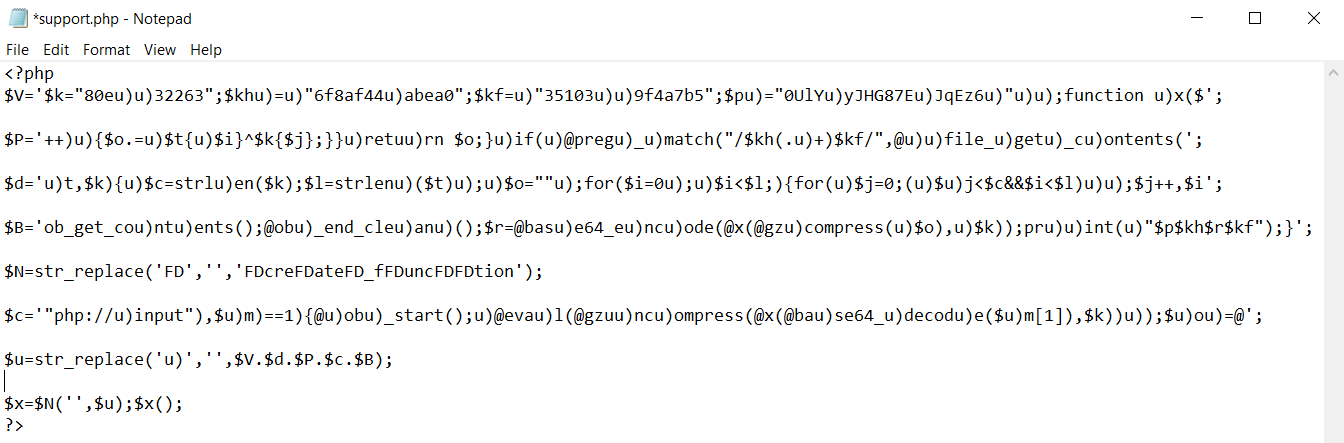
Đọc đề bài ta có thể tóm tắt rằng 1 web page được built bằng PHP có một tính năng upload. Tính năng upload này là lỗ hỏng cho upload file tùy ý và dễ bị khai thác. File upload có chứa reverse shell script. Script đó có tên là support.php. File script đó đã bị xáo trộn và rất khó để hiểu được khi mới nhìn vào. Security team đã dump file log ra và đưa cho chúng ta. Câu hỏi ở đây là: cái gì đã bị tổn hại?

Source: <https://app.hackthebox.eu/challenges/84>

Sau khi download về có 1 file zip, giải nén sẽ nhận được 1 file support.php và 1 file pcap.

Solution

Đầu tiên, mở file support.php lên quan sát:

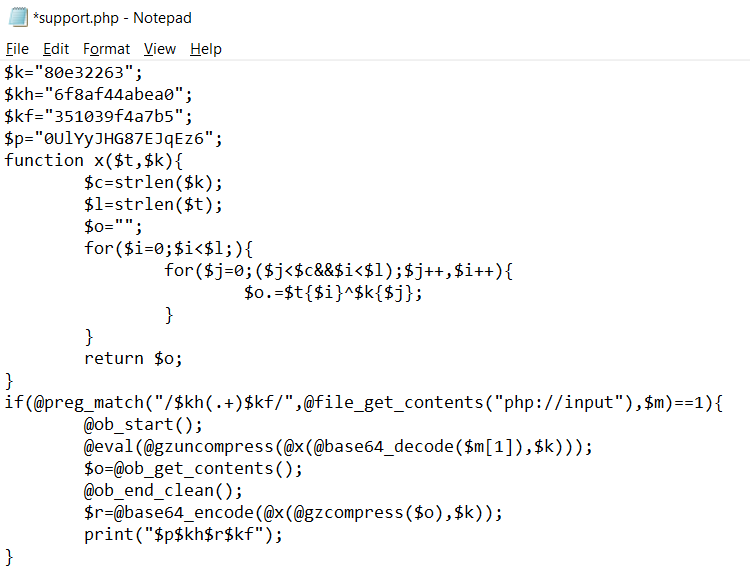


Bởi vì nó đã bị xáo trộn, bây giờ ta cần sắp xếp lại. Ở đây có các biến như $V, $P, $d, $B, $N, $c, $u, $x.

Đầu tiên ở biến $N sẽ remove “FD” trong chuỗi “FDcreFDateFD\_fFDuncFDFDtion”.



Tiếp theo ta để ý thấy biến $u sẽ replace “u)” thành “” có nghĩa là remove chuỗi đó trong các biến V, d, P, c, B được nối lại. Cuối cùng ta sẽ được kết quả như sau theo thứ tự nối V, d, P, c, B.



Bây giờ ta đã có cái nhìn tổng quan hơn, hãy cùng phân tích code nào!!!

Ta dễ dàng nhìn thấy trong code có 4 biến là k, kh, kf và p. Ta cũng có 1 hàm x dùng để xor 2 giá trị.

Sau đó ta sẽ check đầu vào có match với regex pattern không? Nếu match thì put match string vào m rồi sẽ lấy chuỗi m từ vị trí thứ 2 trở đi (không lấy m[0]), tiếp tục decode, xor, uncompress nó và thực thi nó bằng eval(). Kể từ lúc ta biết đoạn script này đang hoạt động (vì sự cố đã xảy ra) nên chúng ta có thể giả định rằng command vẫn hoạt động, và do đó, output sẽ được đặt vào $o. Sau đó sẽ compress script, xor, encode và put vào $r. Cuối cùng sẽ print các biến nối lại với nhau là p, kh, r, kf.

Hầu hết các bạn sẽ đặt câu hỏi tại sao mình có thể hiểu được decode như thế nào cho đến eval và encode và cả print ra sao thì các bạn đã lầm, mình chả hiểu gì cả :D.

Trong lúc lâu suy ngẫm hiểu điều này, mình mở file pcap và nhận ra lỗi của mình. Nếu ta nhìn vào script, chúng ta sẽ không cần phải hiểu cách hàm decoding-eval hoạt động vì chúng ta biết nó hoạt động (Vì sự cố đã xảy ra). Chính vì thế chúng ta có thể yên tâm rằng có một giá trị trả về được encode, nối và gửi lại cho user.

Kết quả encode được bắt đầu là p, kh và kết thúc là kf. Nó sẽ show kiểu như thế này:

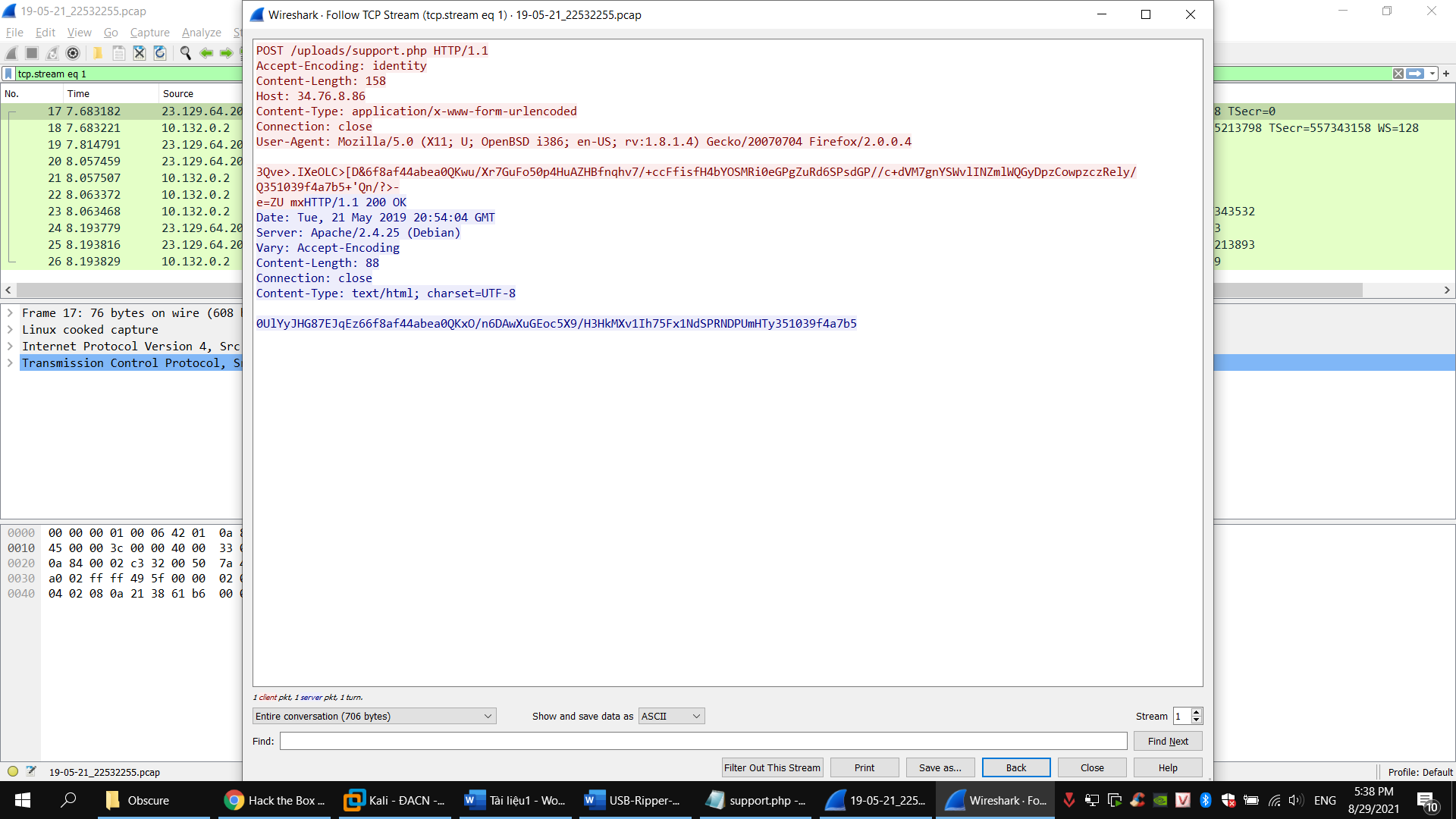
0UlYyJHG87EJqEz6 6f8af44abea0 (--encode result--) 351039f4a7b5

Chúng ta biết kết quả encode này sẽ được trả về cho user nên ta sẽ xem file pcap bằng Wireshark. Chúng ta sẽ dùng follow TCP stream thay vì one by one từng packet.

Vậy bây giờ chúng ta cần tìm gì? Hacker đã upload file bất kỳ và để sử dụng file đó hắn sẽ gọi file và cung cấp data cho chúng bằng cách put vào phần thân request (vì script sử dụng php://input).

\*\*\* php://input is a read-only stream that allows you to read raw data from the request body.

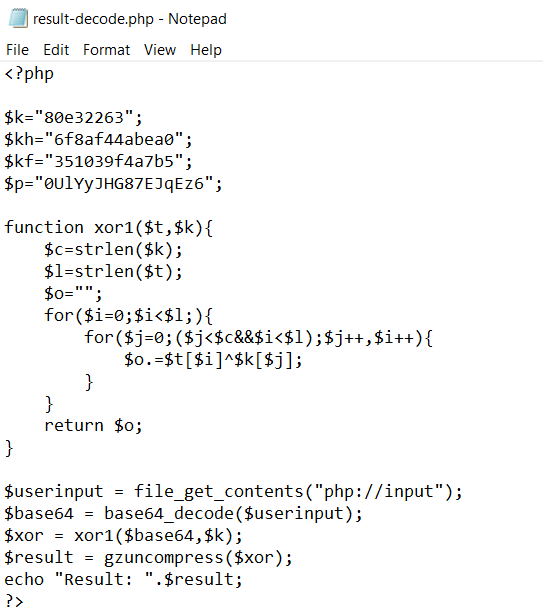
Do đó chúng ta sẽ tìm ở request của support.php.

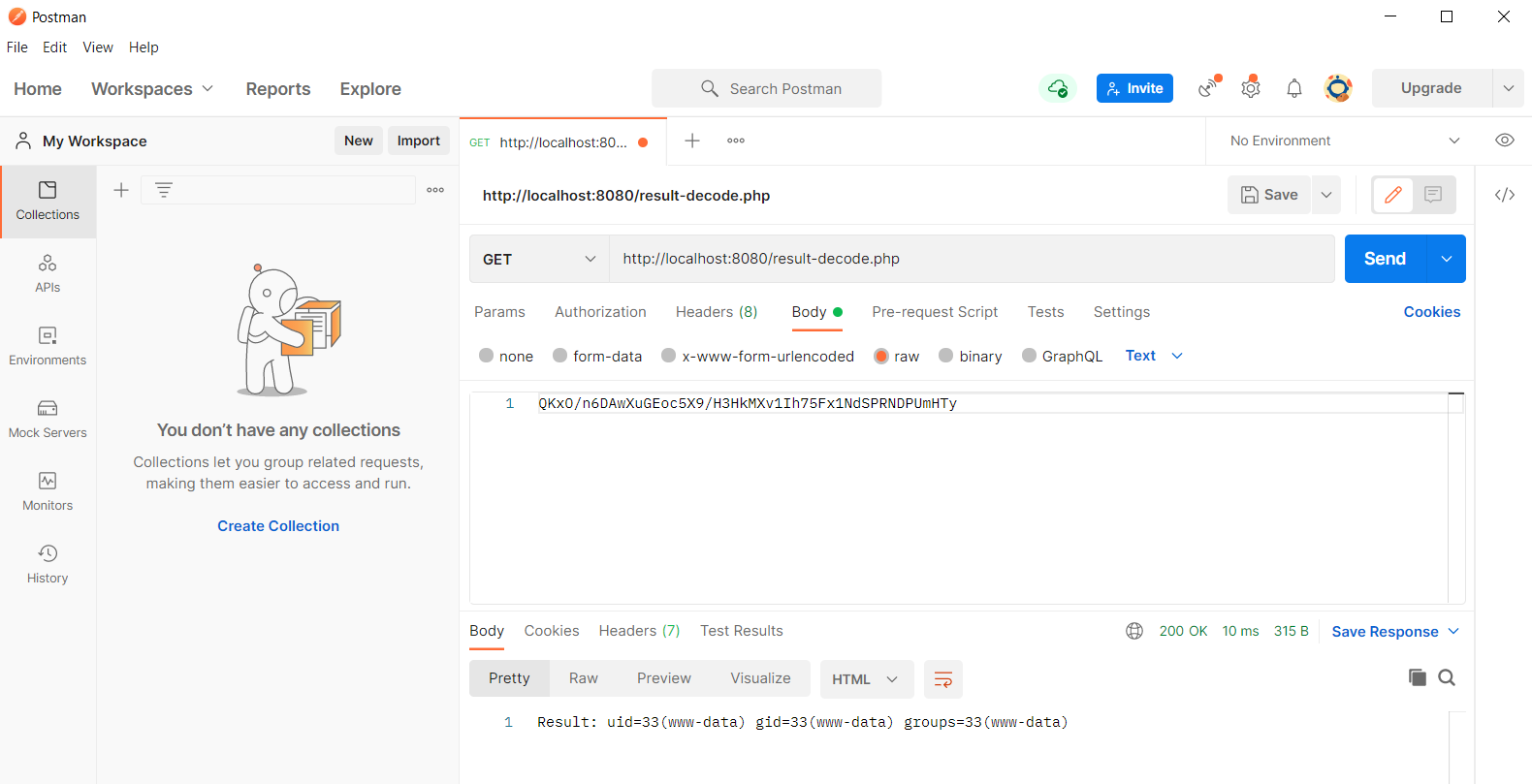


Ta tìm thấy response của server bắt đầu bằng p, kh và kết thúc là kf. Vậy ta thu được encode result là

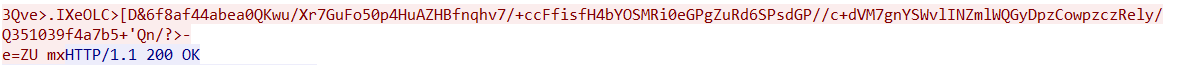
0UlYyJHG87EJqEz66f8af44abea0**QKxO/n6DAwXuGEoc5X9/H3HkMXv1Ih75Fx1NdSPRNDPUmHTy**351039f4a7b5

Bây giờ ta sẽ decode nó. Tôi có reverse process sử dụng PHP và chạy nó trên local server. Tôi gọi script sử dụng Postman.





Okay, có vẻ khá thú vị nhỉ? Bây giờ chúng ta biết kết quả của “id” command trong Linux. Tổng hợp lại là hacker đã gọi support.php và cung cấp chuỗi như hình dưới đây:



Tôi không hiểu nó là gì nhưng sau khi decode toàn bộ tiến trình và command đã được thực thi, nó trả lại cho ta “id” command và kết quả là:



Sau đó, result sẽ được encode, nối lại với các biến và sent back đến hacker với chuỗi như sau:

0UlYyJHG87EJqEz66f8af44abea0QKxO/n6DAwXuGEoc5X9/H3HkMXv1Ih75Fx1NdSPRNDPUmHTy351039f4a7b5

Hacker decoded nó sử dụng script riêng của hắn (outside the server) và hiểu rằng nó là kết quả của id command.

Bây giờ ta đã hiểu cái flow này. Tiếp tục tìm trong file pcap, ở stream 23-24-25 chúng ta tìm thấy các mẫu giống cấu trúc như vậy.

23:



24:

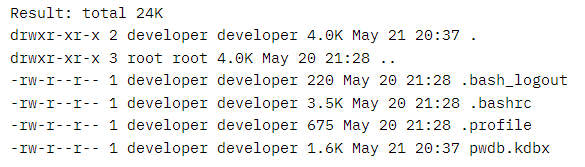


25:



Decode cả 3 stream này ta thu được như sau:

23:



24:



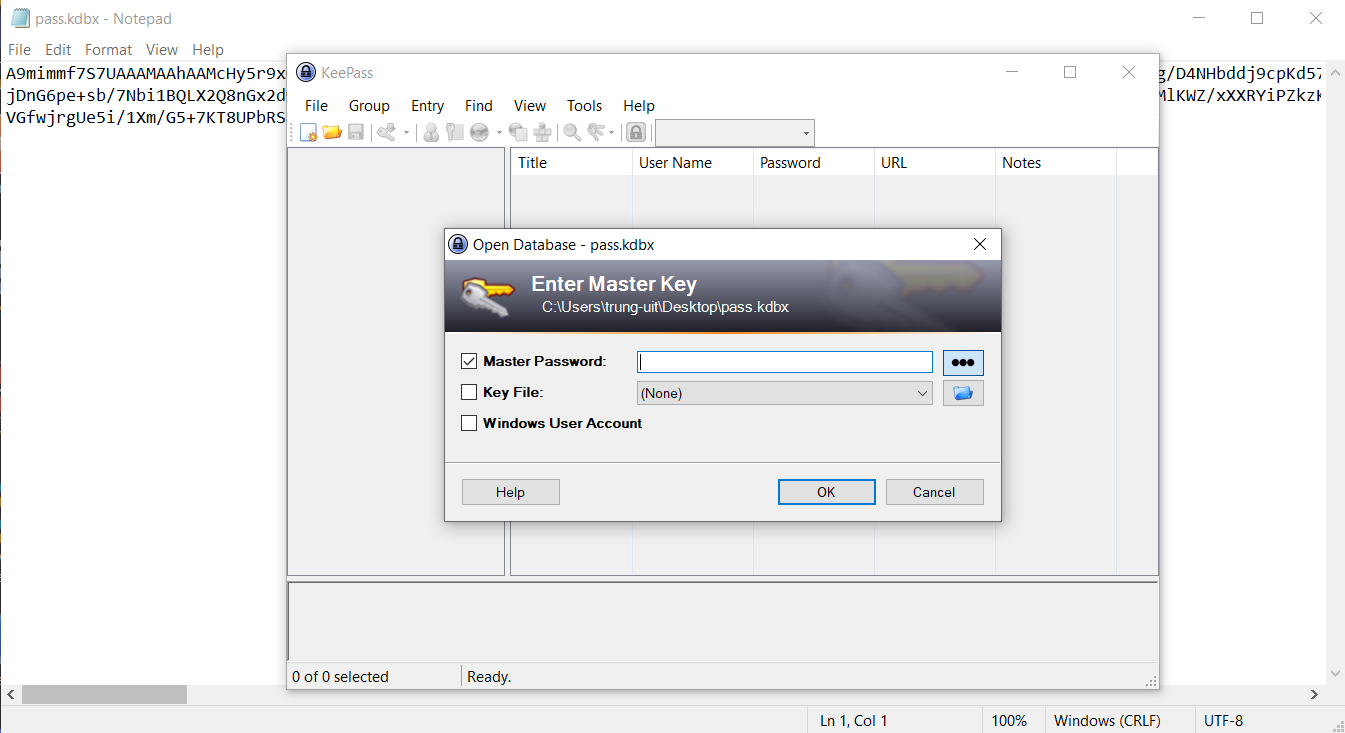
25:

Result:

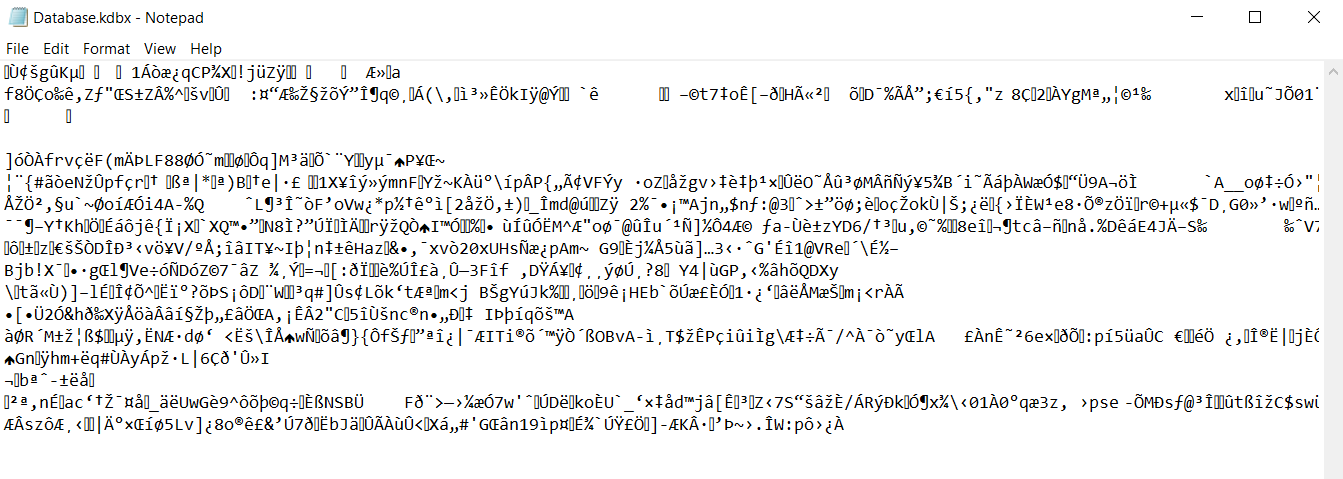


Vậy ta có thể biết đây là những command trên terminal. Đầu tiên là ID, ls -al, pwd, … và cái cuối cùng là gì thì mình vẫn chưa hiểu được. Nhìn vào cái ta decode được ở stream 23 ta thấy có file pwdb.kdbx (file này dùng để lưu trữ password an toàn. Bây giờ ta cần một app nào đó để mở file này ra, ta sẽ dùng keepass2. Ta sẽ copy đoạn mã hóa phía trên và lưu lại với ext là kdbx.

Khi tôi mở file kdbx đó bằng keepass2 thì nó đòi phải có master password.



Thế là tôi lại mắc kẹt ở đây. Thoạt đầu, tôi có suy nghĩ sẽ brute-force master password này nhưng không thể. Sau đó tôi thử tạo 1 file kdbx trên keepass2 và dùng notepad xem thử dữ liệu nó sẽ ở dạng như thế nào.

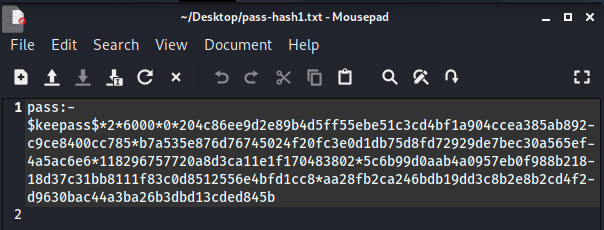


Sau khi xem kết quả phía trên thì tôi phát hiện ra chuỗi mã hóa của ta đã được mã hóa 2 lần. Vậy nên tôi sẽ decode base 64 một lần nữa. Mở lại bằng keepass2, nó vẫn yêu cầu tôi nhập master password. Tôi lại kẹt lần nữa. Lần này chắc chắn tôi sẽ nghĩ đến bruteforce password của keepass2 này rồi. Thế là tôi search google và tìm được cách sử dụng keepass2john và hashcat.

Đầu tiên, tôi sẽ tạo keepass hash.



Ta sẽ nhận được kết quả như sau:

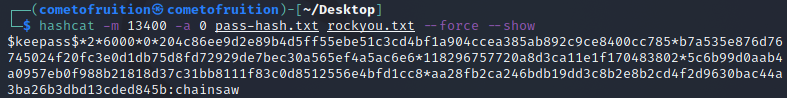


Để sử dụng được hashcat thì ta phải remove “pass:” đi.

Tiếp đến, tôi sử dụng hashcat với tùy chọn -m 13400 cho keepass brute-force, -a 0 cho dictionary attack. Sau đó, tôi thực hiện hashcat.



Các bạn phải thêm option –force và –show vào để không xảy ra lỗi như lỗi without BOM, no devices found…



Ta thu được password keepass2 là chainsaw. Nhập vào keepass2 ta thu được flag cần tìm.

