InnoCTF

Writeups of Some challenges of Innoctf by batman/joker of Scr!ptk!dd!es

Hex64

Download the the attachment

it is a big file name Hex64 with text in it. one can easily recognize by Looking at it that its a Base64.
decode using : **cat Hex64 | base64 -d** in terminal gives us a hex which decoded to base64 again
it seems like flag is encrypted multiple time with base64 then hex combination.
SO i write a python script to perform the task

my scipt was

```
f=open('Hex64','r')
c=""

for line in f:
   c+=line.replace('\n','')
while 'InnoCTF{' not in c:
  try:
    h=c.decode('base64')
    c = h.decode('hex')
  except:
    print c
    break
print c
```

which give the flag : **InnoCTF{why_s0_larg3}**

graphity

challenge at : **nc 188.130.155.66 4999**

connecting to port we get edges of a tree in (src,dest,distance) format and we also get point A and point B.
our task is to find the shortest path from A to B and send the distance back to server.
repeat this for several times to get the flag.
So i write python script with implementation of Dijkastra algorithm.

python3:

```
from collections import deque, namedtuple
from pwn import *
from ast import literal_eval as make_tuple
# we'll use infinity as a default distance to nodes.
inf = float('inf')
Edge = namedtuple('Edge', 'start, end, cost')
host='188.130.155.66'
port=4999

def make_edge(start, end, cost=1):
  return Edge(start, end, cost)


class Graph:
    def __init__(self, edges):
        # let's check that the data is right
        wrong_edges = [i for i in edges if len(i) not in [2, 3]]
        if wrong_edges:
            raise ValueError('Wrong edges data: {}'.format(wrong_edges))

        self.edges = [make_edge(*edge) for edge in edges]

    @property
    def vertices(self):
        return set(
            sum(
                ([edge.start, edge.end] for edge in self.edges), []
            )
        )

    def get_node_pairs(self, n1, n2, both_ends=True):
        if both_ends:
            node_pairs = [[n1, n2], [n2, n1]]
        else:
            node_pairs = [[n1, n2]]
        return node_pairs

    def remove_edge(self, n1, n2, both_ends=True):
        node_pairs = self.get_node_pairs(n1, n2, both_ends)
        edges = self.edges[:]
        for edge in edges:
            if [edge.start, edge.end] in node_pairs:
                self.edges.remove(edge)

    def add_edge(self, n1, n2, cost=1, both_ends=True):
        node_pairs = self.get_node_pairs(n1, n2, both_ends)
        for edge in self.edges:
            if [edge.start, edge.end] in node_pairs:
                return ValueError('Edge {} {} already exists'.format(n1, n2))

        self.edges.append(Edge(start=n1, end=n2, cost=cost))
        if both_ends:
            self.edges.append(Edge(start=n2, end=n1, cost=cost))

    @property
    def neighbours(self):
        neighbours = {vertex: set() for vertex in self.vertices}
        for edge in self.edges:
            neighbours[edge.start].add((edge.end, edge.cost))

        return neighbours

    def dijkstra(self, source, dest):
        assert source in self.vertices, 'Such source node doesn\'t exist' #'
        distances = {vertex: inf for vertex in self.vertices}
        previous_vertices = {
            vertex: None for vertex in self.vertices
        }
        distances[source] = 0
        vertices = self.vertices.copy()

        while vertices:
            current_vertex = min(
                vertices, key=lambda vertex: distances[vertex])
```

```
            vertices.remove(current_vertex)
            if distances[current_vertex] == inf:
                break
            for neighbour, cost in self.neighbours[current_vertex]:
                alternative_route = distances[current_vertex] + cost
                if alternative_route < distances[neighbour]:
                    distances[neighbour] = alternative_route
                    previous_vertices[neighbour] = current_vertex

        path, current_vertex = deque(), dest
        while previous_vertices[current_vertex] is not None:
            path.appendleft(current_vertex)
            current_vertex = previous_vertices[current_vertex]
        if path:
            path.appendleft(current_vertex)
        return path

sh=remote(host,port)
sh.recvuntil('Lets start!\n')
while 1:
 o=sh.recvuntil('\n')
 p=((o.decode(encoding='utf-8', errors='strict')))
 points=make_tuple(p)
 print(points)
 gra=[]
 g1=[]
 for p in points:
    i='("'+chr(p[0]+ord('A'))+'", "'+chr(p[1]+ord('A'))+'", '+str(p[2])+')'
    j='("'+chr(p[1]+ord('A'))+'", "'+chr(p[0]+ord('A'))+'", '+str(p[2])+')'
    #print(i)
    #print(make_tuple(i))
    gra.append(make_tuple(i))
    gra.append(make_tuple(j))
 for p in points:
    i='("'+chr(p[0]+ord('A'))+'", "'+chr(p[1]+ord('A'))+'", '+str(p[2])+')'
    #j='("'+chr(p[1]+ord('A'))+'", "'+chr(p[0]+ord('A'))+'", '+str(p[2])+')'
    #print(i)
    #print(make_tuple(i))
    g1.append(make_tuple(i))
    #gra.append(make_tuple(j))

 print(gra)
 graph = Graph(gra)
 o=(sh.recvuntil('\n').decode(encoding='utf-8', errors='strict')).split(' ')
 #print(int(o[4]),int(o[7]))
 src=chr(ord('A')+int(o[4]))
 dst=chr(ord('A')+int(o[7]))
 #print(src,dst)
 path=(graph.dijkstra(src,dst))
 print(path)
 a=path.popleft()
 distance=0
 for e in g1:
    if e[0]==src and e[1]==dst:
      distance=e[2]
 if distance > 0:
     print(distance)
 else:
  while len(path)!=0:
    b=path.popleft()
    for e in gra:
      if e[0]==a and e[1]==b:
        distance+=e[2]
        break
    a=b
 #print(distance)
 sh.sendline(str(distance))
 print(sh.recv())
```

python2:

```
from collections import *
import re
import time
from pwn import *

class Graph:
        def __init__(self):
                self.nodes = set()
                self.edges = defaultdict(list)
                self.distances = {}

        def add_node(self, value):
                self.nodes.add(value)

        def add_edge(self, from_node, to_node, distance):
                self.edges[from_node].append(to_node)
                self.edges[to_node].append(from_node)
                self.distances[(from_node, to_node)] = distance


def dijsktra(graph, initial):
        visited = {initial: 0}
        path = {}

        nodes = set(graph.nodes)

        while nodes:
                min_node = None
                for node in nodes:
                        if node in visited:
                                if min_node is None:
                                        min_node = node
                                elif visited[node] < visited[min_node]:
                                        min_node = node

                if min_node is None:
                        break

                nodes.remove(min_node)
                current_weight = visited[min_node]

                for edge in graph.edges[min_node]:
                        weight = current_weight + graph.distances[(min_node, edge)]
                        if edge not in visited or weight < visited[edge]:
                                visited[edge] = weight
                                path[edge] = min_node

        return visited, path

def process(string):
        return str(string.count('(')) + '\n' + '\n'.join(map(lambda
        t:t.replace(',',''), re.findall('[0-9 ]+,[0-9 ]+,[0-9 ]+', string)))

ip, port = '188.130.155.66', 4999
p = remote(ip, port)

time.sleep(2)

boo = 0

while True:
        if boo == 0:
                u = p.recvuntil('Lets start!\n')
                boo = 1
        r = p.recv()
        print (r)
        if 'CTF' in r:
            print (r)
            open('flag1','w').write(str(r))
            break
        # if 'No!' in r:
        #       break
        cp = r.split('\n')
        print cp
```

```
        print "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
        if cp[0]=='Right!':
           cp.remove('Right!')
           if cp[0] == '':
              r = p.recv()
              cp = r.split('\n')
              print cp
        cp = cp[-3:-1]
        print "Processing:", cp[0]
        open('abc','w').write(str(process(cp[0])))
        g = Graph()
        f = open('abc','r')
        for i in range(int(f.readline().strip())):
                a, b, c = map(int, f.readline().strip().split())
                # a, b = sorted([a, b])
                if c == 0: continue
                g.add_node(a)
                g.add_node(b)
                g.add_edge(a, b, c)
                g.add_edge(b, a, c)

        # My way from point 0 to point 9
        print "*******************************************"
        print cp
        nodes = re.findall('My way from point ([0-9]+) to point ([0-9]+)', cp[1])[0]
        x, y = map(int, nodes)
        po = 0
        print x, y
        try: po = dijsktra(g, x)[0][y]
        except: pass
        try: po = dijsktra(g, y)[0][x]
        except: pass
        print "My answer:", po
        p.sendline(str(po))

        time.sleep(1)
```

running this script we get the flag after couple of minutes

although there are some issue with python3 algo... if found please fix

sorry for bad english!!

look_ahead

this challenge is in misc category

we get a pcapng file . Opening that in wireshark we get lots of packets info.
There were TCP,TLS,DNS,HTTP streams in file. Ichecked all stream using right click and follow option in wire shark.
Found intresting things in HTTP stream.

the host was : http://spbctf.ppctf.net:43317/

i vist that page and got this page

0/37

```
00000000: 42 5A 68 39 31 41 59 26  53 59 29 76 31 28 00 1C  BZh91AY&SY)v1(..
00000010: F5 FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000020: FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ................
00000030: FF FF FF E0 42 7F 7C 17  B6 02 A9 50 00 00 1A 55  ....B.|....P...U
00000040: 7C C1 F0 F9 20 00 BC 0F  83 80 1D 85 80 C3 61 6C  |... .........al
00000050: 36 60 DA C1 91 AA 1A 16  C5 14 D3 4D B5 4A AA 00  6`.........M.J..
00000060: F1 F0 92 AA 2A A5 48 29  29 48 25 28 49 11 15 01  ....*.H))H%(I...
00000070: 01 15 08 A4 AA 2A 8A 95  50 95 52 92 A8 31 EF 61  .....*..P.R..1.a
00000080: 6B 36 D6 B4 A9 53 4D 2A  D9 92 D8 B6 5A C4 8C 86  k6...SM*....Z...
00000090: CC 08 40 0D 8F 8A 6E 12  2A A9 07 DD 7D B4 00 00  ..@...n.*...}...
000000A0: 00 00 3E 0E C3 AA 7B 40  68 1A 34 D0 00 00 13 00  ..>...{@h.4.....
000000B0: 26 00 02 60 1A 00 06 A7  A0 00 01 30 00 02 64 C0  &..`.......0..d.
000000C0: 00 8C 00 98 13 00 04 C0  04 C0 00 4C 00 02 60 9A  ...........L..`.
000000D0: 60 00 8C A1 13 48 80 81  A1 30 4C 06 82 6D 13 D1  `....H...0L..m..
000000E0: 32 63 46 84 66 46 81 31  30 1A 06 A6 4D 0C A7 93  2cF.fF.10...M...
000000F0: 13 23 41 91 84 D1 A3 44  F0 4D 06 4D 08 9F A1 30  .#A....D.M.M...0
00000100: 9A 7A 4D A2 7A 32 68 04  F4 A9 FE 9A 99 A4 C8 CD  .zM.z2h.........
00000110: 34 34 8F 10 99 30 26 A7  84 C2 A7 E5 06 A7 A4 88  44...0&.........
00000120: 14 CC 63 40 4C 04 00 8C  09 30 89 89 A4 9F A5 3C  ..c@L....0.....<
00000130: D5 3C A7 92 36 A3 CA 1B  29 A7 A8 1A 34 D0 34 F5  .<..6...)...4.4.
00000140: 34 7A 80 3D 40 03 D2 68  C8 3D 4D 00 00 03 6A 00  4z.=@..h.=M...j.
00000150: 18 80 D0 68 00 00 06 40  03 40 F5 1E A3 0D 4C 88  ...h...@.@....L.
00000160: 40 36 02 23 43 40 34 02  26 8D 32 A7 91 32 83 4D  @6.#C@4.&.2..2.M
00000170: 34 34 F1 41 A0 0C 80 03  40 0D 00 D3 40 00 01 E9  44.A....@...@...
00000180: 00 00 03 40 0D 03 40 00  00 0D 00 00 00 00 00 06  ...@..@.........
00000190: A7 88 84 4D 01 00 53 C8  99 30 A6 CA 9E A7 B4 49  ...M..S..0.....I
000001A0: A7 A9 FA A7 EA 8D 3F 4A  7A 9A 7A 8F 14 1E 53 D2  ......?Jz.z...S.
000001B0: 7A 99 34 3D 35 32 3D 20  64 F5 34 1E A0 69 88 03  z.4=52= d.4..i..
000001C0: D4 D0 00 DA 80 68 00 06  46 80 3D 40 34 0C 86 80  .....h..F.=@4...
000001D0: 01 A0 34 06 83 D4 34 07  A0 93 4A 45 34 44 CA 9E  ..4...4...JE4D..
000001E0: D4 DA 6A 32 9E 29 FA 53  63 4A 79 4F 4D 4F 35 3C  ..j2.).ScJyOMO5<
000001F0: BA 0F 49 E8 F2 93 CA 1E  A1 ED 53 D4 7A 9E A6 31  ..I.......S.z..1
```

avi	bmp	cab	cer
class	dat	dex	doc
elf	evtx	exe	gif
inf	jpg	mbr	mov
mp3	pcap	pdf	png
pwl	pyc	rar	rtf
swf	tif	txt	wav
wav.bz2	wav.gz	zip	zlib

I was quite sure that i was on right track then a member of my team suggest its related to etags og html streams... After checking for http streams, he was right... flags was in etags of all http streams b/w Src: 78.46.101.237, Dst: 192.168.0.5

you can see that etag in above two images combines for ‘In’
similarly follow the all http streams i get the flag InnoCTF{h34ders_h4ve_Secr3ts_To0_1296d}

It matches

We get a flag file with many flags in it plus a hint how the correct flag looks like for us it was
Flag is of 2 words
upper+nums+lower and lower+nums

a friend suggest me to use regex101
you can see regex used here is **InnoCTF{[[:alnum:]]*_([[:lower:]]|[[:digit:]])*}**

and at the right side we have a match **InnoCTF{p0W3rFUl_r3g3xp}** which was also the flag

back in time

we have a website
visiting it just show a backgroubd image and some text

looking at the source code i found assets folder which is reached using this

exploring it doen't get me anywhere. A friend told me its related to git and send me this link

i follow couples of link and found a handy GITTOOLS repository

using its gitdumper i dumped the al git commits

command used

```
./gitdumper.sh http://188.130.155.66:1111/HWmIoprRTKguSHBsEgsUkLJnfDOQHkjR/.git/ /tmp/git-dump
./extractor.sh /tmp/git-dump/ /tmp/mygitrepodump
```

it downloaded all objects and then i use grep to find flag in those files

cat */index.php|grep Inno

it gives the flag

sorry for bad clarification as i used /tmp folder and everything wiped when the system rebooted and challenge sites are down so i can't post actual screenshots of POC