# KludgeCTF

Arjun Pavanje

May 2025

# 1 Miscellaneous

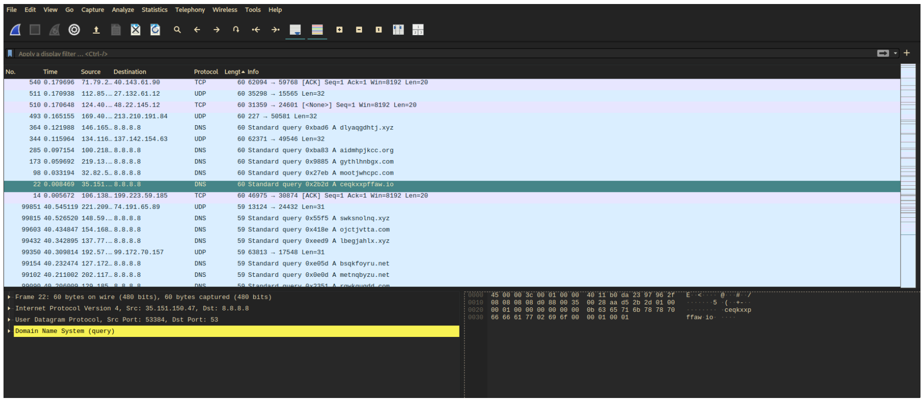
## 1.1 I am not MID

Tried the flag given in the question, and it worked :)

# 2 Forensics

## 2.1 Chatty Network

We were given a packet capture file with a suspicion that malware maybe stealing data during look ups.
Analysing using Wireshark,,,

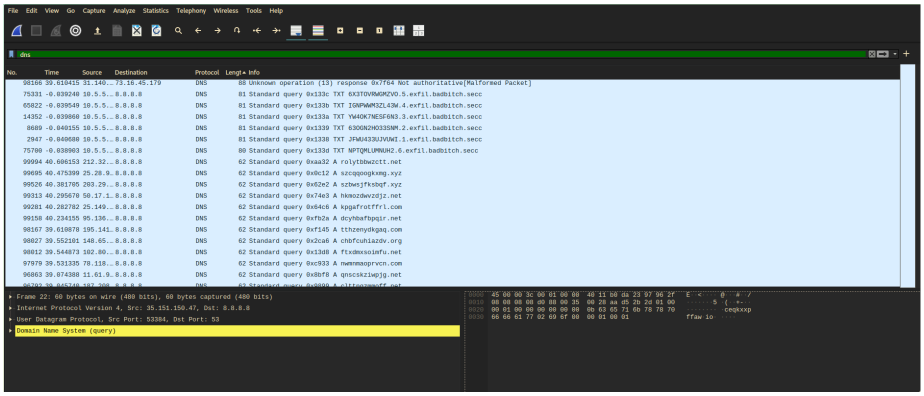


```
No.   Time      Source     Destination      Protocol Lengt▲ Info
  540 0.179696  71.79.2…   40.143.61.90     TCP      60 62094 → 59768 [ACK] Seq=1 Ack=1 Win=8192 Len=20
  511 0.170938  112.85.…   27.132.61.12     UDP      60 35298 → 15565 Len=32
  510 0.170648  124.40.…   48.22.145.12     TCP      60 31359 → 24601 [<None>] Seq=1 Win=8192 Len=20
  493 0.165155  169.40.…   213.210.191.84   UDP      60 227 → 50581 Len=32
  364 0.121988  146.165…   8.8.8.8          DNS      60 Standard query 0xbad6 A dlyaqgdhtj.xyz
  344 0.115964  134.116…   137.142.154.63   UDP      60 62371 → 49546 Len=32
  285 0.097154  100.218…   8.8.8.8          DNS      60 Standard query 0xba83 A aidmhpjkcc.org
  173 0.059692  219.13.…   8.8.8.8          DNS      60 Standard query 0x9885 A gythlhnbgx.com
   98 0.033194  32.82.5…   8.8.8.8          DNS      60 Standard query 0x27eb A mootjwhcpc.com
   22 0.008469  35.151.…   8.8.8.8          DNS      60 Standard query 0x2b2d A ceqkxxpffaw.io
   14 0.005672  106.138…   199.223.59.185   TCP      60 46975 → 30874 [ACK] Seq=1 Ack=1 Win=8192 Len=20
99851 40.545119 221.209…   74.191.65.89     UDP      59 13124 → 24432 Len=31
99815 40.526520 148.59.…   8.8.8.8          DNS      59 Standard query 0x55f5 A swksnolnq.xyz
99603 40.434847 154.168…   8.8.8.8          DNS      59 Standard query 0x418e A ojctjvtta.com
99432 40.342895 137.77.…   8.8.8.8          DNS      59 Standard query 0xeed9 A lbegjahlx.xyz
99350 40.309814 192.57.…   99.172.70.157    UDP      59 63813 → 17548 Len=31
99154 40.232474 127.172…   8.8.8.8          DNS      59 Standard query 0xe05d A bsqkfoyru.net
99102 40.211002 202.117…   8.8.8.8          DNS      59 Standard query 0x0e0d A metnqbyzu.net
99090 40.206009 129.185…   8.8.8.8          DNS      59 Standard query 0x2351 A rqwkqugdd.com
```

```
▸ Frame 22: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▸ Internet Protocol Version 4, Src: 35.151.150.47, Dst: 8.8.8.8
▸ User Datagram Protocol, Src Port: 53384, Dst Port: 53
▸ Domain Name System (query)
```

```
0000  45 00 00 3c 00 01 00 00  40 11 b0 da 23 97 96 2f   E··<····@···#··/
0010  08 08 08 08 d0 88 00 35  00 28 aa d5 2b 2d 01 00   ·······5·(··+-··
0020  00 01 00 00 00 00 00 00  0b 63 65 71 6b 78 78 70   ·········ceqkxxp
0030  66 66 61 77 02 69 6f 00  00 01 00 01               ffaw·io·····
```

Viewing all communication made using DNS protocol,



```
No.      Time      Source     Destination      Protocol  Lengt▲ Info
  98166 39.610415 31.140.… 73.16.45.179       DNS         88 Unknown operation (13) response 0x7f64 Not authoritative[Malformed Packet]
  75331 -0.039240 10.5.5.… 8.8.8.8            DNS         81 Standard query 0x133c TXT 6X3TOVRWGMZVO.5.exfil.badbitch.secc
  65822 -0.039549 10.5.5.… 8.8.8.8            DNS         81 Standard query 0x133b TXT IGNPWWM3ZL43W.4.exfil.badbitch.secc
  14352 -0.039860 10.5.5.… 8.8.8.8            DNS         81 Standard query 0x133a TXT YW4OK7NESF6N3.3.exfil.badbitch.secc
   8689 -0.040155 10.5.5.… 8.8.8.8            DNS         81 Standard query 0x1339 TXT 63OGN2HO33SNM.2.exfil.badbitch.secc
   2947 -0.040680 10.5.5.… 8.8.8.8            DNS         81 Standard query 0x1338 TXT JFWU433UJVUWI.1.exfil.badbitch.secc
  75700 -0.038903 10.5.5.… 8.8.8.8            DNS         80 Standard query 0x133d TXT NPTQMLUMNUH2.6.exfil.badbitch.secc
  99994 40.606153 212.32.… 8.8.8.8            DNS         62 Standard query 0xaa32 A rolytbbwzctt.net
  99695 40.475399 25.28.9… 8.8.8.8            DNS         62 Standard query 0x0c12 A szcqqoogkxmg.xyz
  99526 40.381705 203.29.… 8.8.8.8            DNS         62 Standard query 0x62e2 A szbwsjfksbqf.xyz
  99313 40.295670 50.17.1… 8.8.8.8            DNS         62 Standard query 0x74e3 A hkmozdwvzdjz.net
  99281 40.282782 25.149.… 8.8.8.8            DNS         62 Standard query 0x64c6 A kpgafrotffrl.com
  99158 40.234155 95.136.… 8.8.8.8            DNS         62 Standard query 0xfb2a A dcyhbafbpqir.net
  98167 39.610878 195.141… 8.8.8.8            DNS         62 Standard query 0xf145 A tthzenydkgaq.com
  98027 39.552101 148.65.… 8.8.8.8            DNS         62 Standard query 0x2ca6 A chbfcuhiazdv.org
  98012 39.544873 102.80.… 8.8.8.8            DNS         62 Standard query 0x13d8 A ftxdmxsoimfu.net
  97979 39.531335 78.118.… 8.8.8.8            DNS         62 Standard query 0xc933 A nwmnmaoprvcn.com
  96863 39.074388 11.61.9… 8.8.8.8            DNS         62 Standard query 0x8bf8 A qnscskziwpjg.net
  96792 39.045740 187.208… 8.8.8.8            DNS         62 Standard query 0x9899 A clttnqzmmoff.net

▸ Frame 22: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
▸ Internet Protocol Version 4, Src: 35.151.150.47, Dst: 8.8.8.8
▸ User Datagram Protocol, Src Port: 53384, Dst Port: 53
▸ Domain Name System (query)

0000  45 00 00 3c 00 01 00 00  40 11 b0 da 23 97 96 2f   E··<····@···#··/
0010  08 08 08 08 d0 88 00 35  00 28 aa d5 2b 2d 01 00   ·······5·(··+-··
0020  00 01 00 00 00 00 00 00  0b 63 65 71 6b 78 78 70   ·········ceqkxxp
0030  66 66 61 77 02 69 6f 00  00 01 00 01               ffaw·io·····
```

On sorting messages in descending order (size), we see 6 messages to the slightly suspicious domain *badbitch.secc*. To my observation, those 6 messages were the only ones that had the same source and destination location. So I guessed that the message may have been split into 6 parts, so I took the text from each image and tried to decode them.

String obtained, $FWU433UJVUWI63OGN2HO33SNMYW4OK7NESF6$ $N3IGNPWWM3ZL43W6X3TOVRWGMZVONPTQMLUMNUH2$

This looked to be base-32 code, so I wrote a python code to decode it by making use of pythons *base*64 library. Python code,

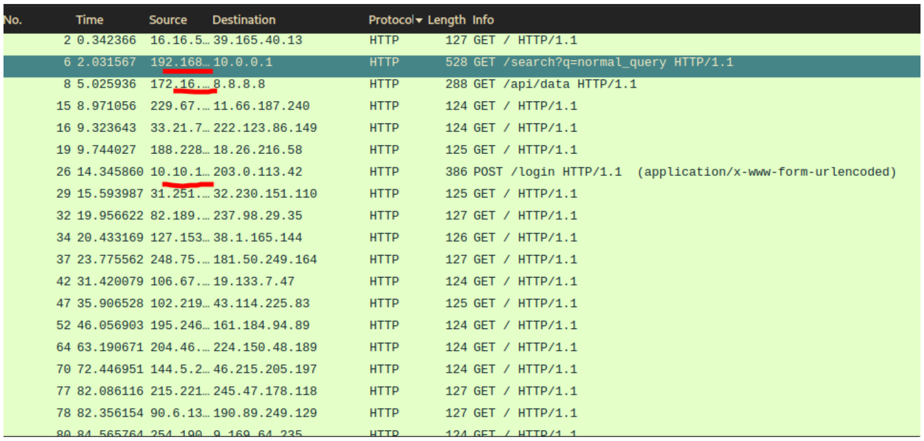
```
import base64
# The three '=='s have been added as padding
encoded = "JFWU433UJVUWI63OGN2HO33SNMYW4OK7NESF6
N3IGNPWWM3ZL43W6X3TOVRWGMZVONPTQMLUMNUH2==="
decoded = base64.b32decode(encoded)
print(decoded)
```

This gives us the flag,

```
ImNotMid{n3twork1n9_i$_7h3_k3y_7o_succ35s_81tch}
```

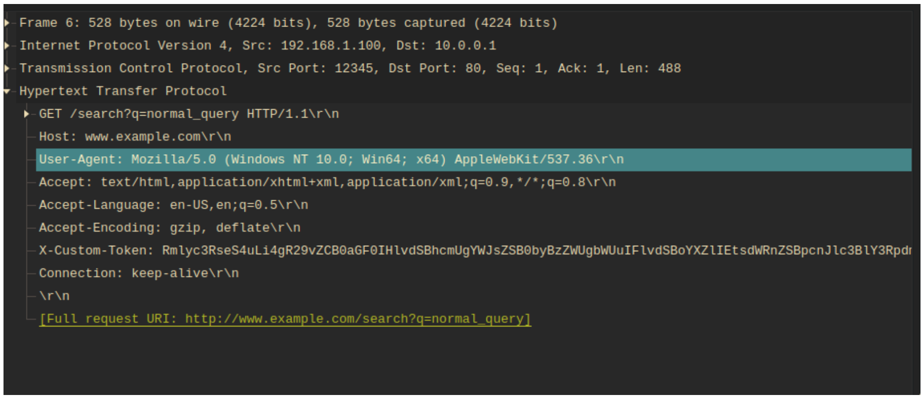
## 2.2 Sniffer

I found the three parts of interest,



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 2 | 0.342366 | 16.16.5… | 39.165.40.13 | HTTP | 127 | GET / HTTP/1.1 |
| 6 | 2.031567 | 192.168… | 10.0.0.1 | HTTP | 528 | GET /search?q=normal_query HTTP/1.1 |
| 8 | 5.025936 | 172.16.… | 8.8.8.8 | HTTP | 288 | GET /api/data HTTP/1.1 |
| 15 | 8.971056 | 229.67.… | 11.66.187.240 | HTTP | 124 | GET / HTTP/1.1 |
| 16 | 9.323643 | 33.21.7… | 222.123.86.149 | HTTP | 124 | GET / HTTP/1.1 |
| 19 | 9.744027 | 188.228… | 18.26.216.58 | HTTP | 125 | GET / HTTP/1.1 |
| 26 | 14.345860 | 10.10.1… | 203.0.113.42 | HTTP | 386 | POST /login HTTP/1.1  (application/x-www-form-urlencoded) |
| 29 | 15.593987 | 31.251.… | 32.230.151.110 | HTTP | 125 | GET / HTTP/1.1 |
| 32 | 19.956622 | 82.189.… | 237.98.29.35 | HTTP | 127 | GET / HTTP/1.1 |
| 34 | 20.433169 | 127.153… | 38.1.165.144 | HTTP | 126 | GET / HTTP/1.1 |
| 37 | 23.775562 | 248.75.… | 181.50.249.164 | HTTP | 127 | GET / HTTP/1.1 |
| 42 | 31.420079 | 106.67.… | 19.133.7.47 | HTTP | 124 | GET / HTTP/1.1 |
| 47 | 35.906528 | 102.219… | 43.114.225.83 | HTTP | 125 | GET / HTTP/1.1 |
| 52 | 46.056903 | 195.246… | 161.184.94.89 | HTTP | 124 | GET / HTTP/1.1 |
| 64 | 63.190671 | 204.46.… | 224.150.48.189 | HTTP | 124 | GET / HTTP/1.1 |
| 70 | 72.446951 | 144.5.2… | 46.215.205.197 | HTTP | 124 | GET / HTTP/1.1 |
| 77 | 82.086116 | 215.221… | 245.47.178.118 | HTTP | 127 | GET / HTTP/1.1 |
| 78 | 82.356154 | 90.6.13… | 190.89.249.129 | HTTP | 127 | GET / HTTP/1.1 |
| 80 | 84.565764 | 254.190… | 9.169.64.235 | HTTP | 124 | GET / HTTP/1.1 |

On analysing the first part, we find a message (aHR0cHM6Ly93d3cuYW5vbmZpbGUuGEvMjM5ZGY1) which is in base64. On decoding we get the url `https://www.anonfile.la/239df5` from where we get a zip file which has the flag but is password locked. On analyzing the second part we get,



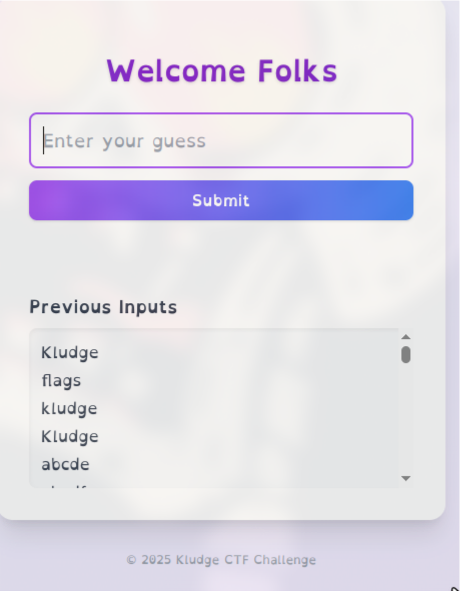
```
▸ Frame 6: 528 bytes on wire (4224 bits), 528 bytes captured (4224 bits)
▸ Internet Protocol Version 4, Src: 192.168.1.100, Dst: 10.0.0.1
▸ Transmission Control Protocol, Src Port: 12345, Dst Port: 80, Seq: 1, Ack: 1, Len: 488
▾ Hypertext Transfer Protocol
  ▸ GET /search?q=normal_query HTTP/1.1\r\n
    Host: www.example.com\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    X-Custom-Token: Rmlyc3RseS4uLi4gR29vZCB0aGF0IHlvdSBhcmUgYWJsZSB0byBzZWUgbWUuIFlvdSBoYXZlIEtsdWRnZSBpcnJlc3BlY3Rpd
    Connection: keep-alive\r\n
    \r\n
    [Full request URI: http://www.example.com/search?q=normal_query]
```

On decoding the X-Custom-Token (base64 encoded) we get, ”Firstly.... Good that you are able to see me. You have Kludge irrespective of your selection. Now go search more with –SUSANO–”.

# 3 Cryptography

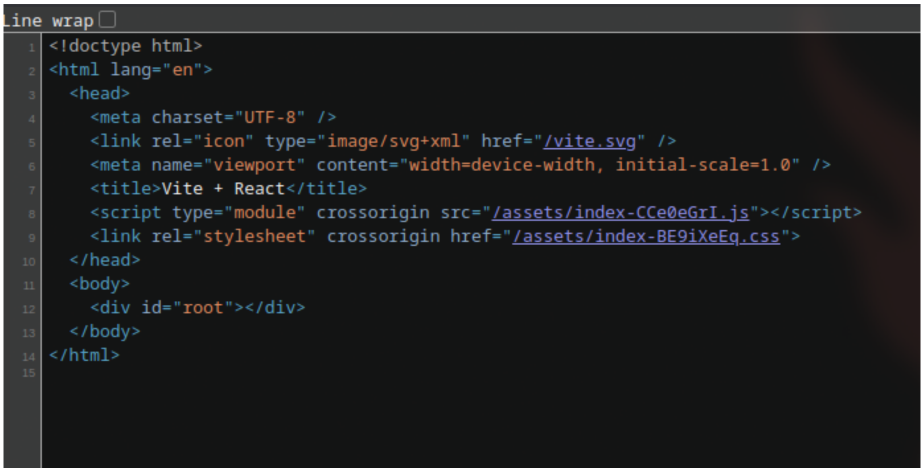
## 3.1 Wordle

Opened the website `https://core-ctf.vercel.app/`,



Kludge
flags
kludge
Kludge
abcde

Initially I thought that it would be a substitution or rot cypher, but after a long while of guessing words I realized that wasn't the case. I tried to find the individual number of each alphabet but realized that the number of an alphabet would be constant only for a given word size (for example the number corresponding to 'A' in PLANT and SAD would be different, but would be the same in case of 'PLANT' and 'ABCDE'). Then I abandoned such ideas and tried to check the source code by pressing $ctrl + U$,



```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
    <script type="module" crossorigin src="/assets/index-CCe0eGrI.js"></script>
    <link rel="stylesheet" crossorigin href="/assets/index-BE9iXeEq.css">
  </head>
  <body>
    <div id="root"></div>
  </body>
</html>
```

Then went to `https://core-ctf.vercel.app/assets/index-CCe0eGrI.js` (after realizing that there was nothing on the second link), where on searching for flag (using $ctrl + F$ I got the flag

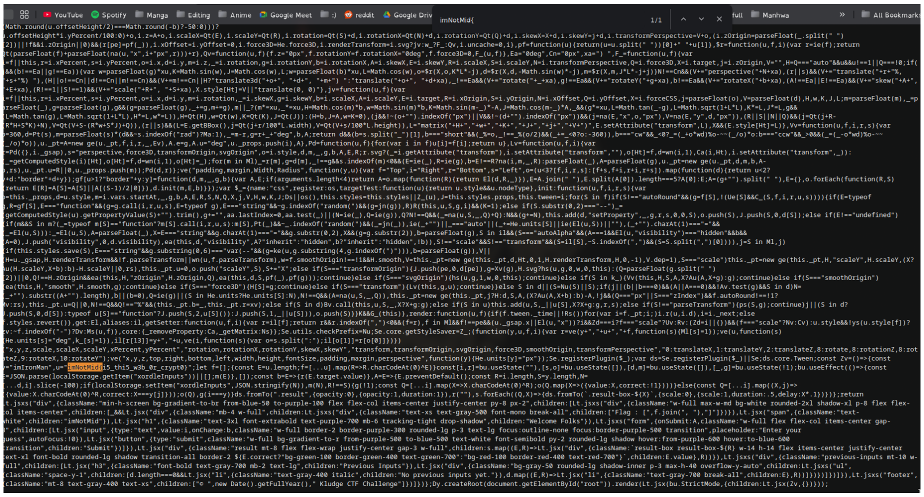
```
imNotMid{i5\_thi5\_w3b\_0r\_crypt0}
```

## 3.2 Crypto Misstep

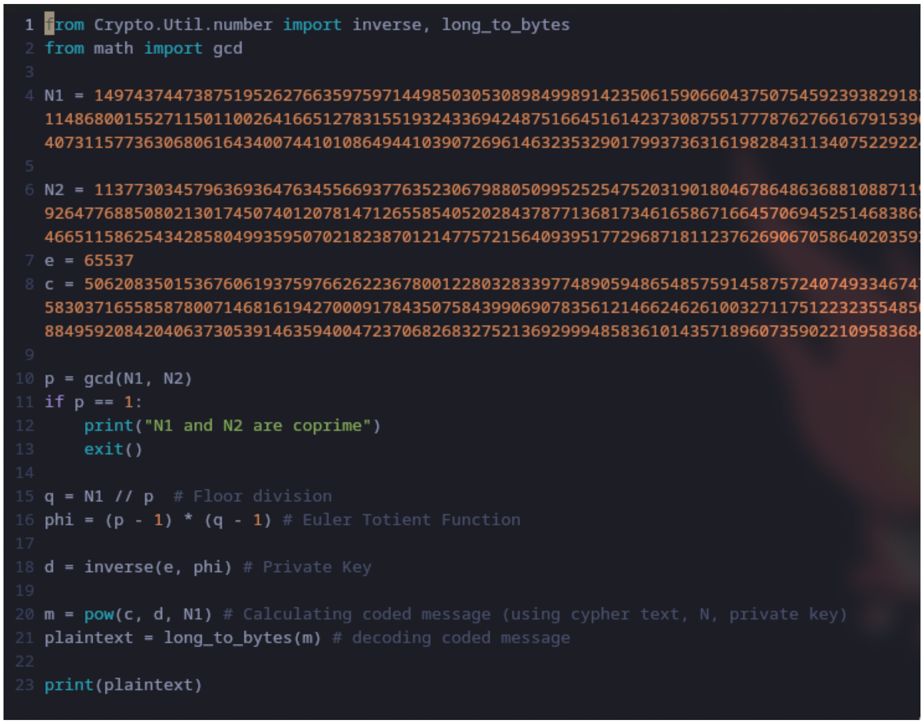
Here, we are given two values of $N$ used in RSA and the standard $e = 65537$ and the cypher text. We are required to obtain the plaintext flag. Usually, it would be extremely difficult to obtain the private key, (which is given by the relation $e * d \equiv 1 mod \varphi(n)$, whre $\varphi$ is Euler Totient function) due $N$ being an extremely large prime number (hence it is extremely difficult to calculate two prime numbers $p$, $q$ which satisfy $p * q = N$ ($\varphi(N)$ is given be $(p-1)(q-1)$).

In this case, we have two $N$ values $(N_1, N_2)$ so if they have a GCD we have found $p$, $q$ for $N_1$ and $N_2$, using which we can calculate Euler Totient function, using which we can calculate private key $d$.

Python code,



```
(Math.round(u.offsetHeight/2)===Math.round(-b)?-50:0)))?
u.offsetHeight*i.yPercent/100:0)+o,i.z=A+o,i.scaleX=Qt(E),i.scaleY=Qt(R),i.rotation=Qt(S)+d,i.rotationX=Qt(N)+d,i.rotationY=Qt(Q)+d,i.skewX=X+d,i.skewY=j+d,i.transformPerspective=V+o,(i.zOrigin=parseFloat(_.split(" ")
[2])||!f&&i.zOrigin||0)&&(r[pe]=pf(_)),i.xOffset=i.yOffset=0,i.force3D=He.force3D,i.renderTransform=i.svg?jv:w_?F_:Qv,i.uncache=0,i},pf=function(u){return(u=u.split(" "))[0]+" "+u[1]},$r=function(u,f,i){var r=ie(f);return
Qt(parseFloat(f)+parseFloat(na(u,"x",i+"px",r)))+r},Qv=function(u,f){f.z="0px",f.rotationY=f.rotationX="0deg",f.force3D=0,F_(u,f)},Ea="0deg",Cn="0px",xa=") ",F_=function(u,f){var
i=f||this,r=i.xPercent,s=i.yPercent,o=i.x,d=i.y,m=i.z,_=i.rotation,g=i.rotationY,b=i.rotationX,A=i.skewX,E=i.skewY,R=i.scaleX,S=i.scaleY,N=i.transformPerspective,Q=i.force3D,X=i.target,j=i.zOrigin,V="",H=Q==="auto"&&u&&u!==1||Q===!0;if(
j&&(b!==Ea||g!==Ea)){var w=parseFloat(g)*xu,K=Math.sin(w),J=Math.cos(w),L;w=parseFloat(b)*xu,L=Math.cos(w),o=$r(X,o,K*L*-j),d=$r(X,d,-Math.sin(w)*-j),m=$r(X,m,J*L*-j+j)}N!==Cn&&(V+="perspective("+N+xa),(r||s)&&(V+="translate("+r+"%,
"+s+"%) "),(H||o!==Cn||d!==Cn||m!==Cn)&&(V+=m!==Cn||H?"translate3d("+o+", "+d+", "+m+") ":"translate("+o+", "+d+xa),_!==Ea&&(V+="rotate("+_+xa),g!==Ea&&(V+="rotateY("+g+xa),b!==Ea&&(V+="rotateX("+b+xa),(A!==Ea||E!==Ea)&&(V+="skew("+A+",
"+E+xa),(R!==1||S!==1)&&(V+="scale("+R+", "+S+xa),X.style[Ht]=V||"translate(0, 0)"},jv=function(u,f){var
i=f||this,r=i.xPercent,s=i.yPercent,o=i.x,d=i.y,m=i.rotation,_=i.skewX,g=i.skewY,b=i.scaleX,A=i.scaleY,E=i.target,R=i.xOrigin,S=i.yOrigin,N=i.xOffset,Q=i.yOffset,X=i.forceCSS,j=parseFloat(o),V=parseFloat(d),H,w,K,J,L;m=parseFloat(m),_=p
arseFloat(_),g=parseFloat(g),g&&(g=parseFloat(g),_+=g,m+=g),m||_?(m*=xu,_*=xu,H=Math.cos(m)*b,w=Math.sin(m)*b,K=Math.sin(m-_)*-A,J=Math.cos(m-_)*A,_&&(g*=xu,L=Math.tan(_-g),L=Math.sqrt(1+L*L),K*=L,J*=L,g&&
(L=Math.tan(g),L=Math.sqrt(1+L*L),H*=L,w*=L)),H=Qt(H),w=Qt(w),K=Qt(K),J=Qt(J)):(H=b,J=A,w=K=0),(j&&!~(o+"").indexOf("px")||V&&!~(d+"").indexOf("px"))&&(j=na(E,"x",o,"px"),V=na(E,"y",d,"px")),(R||S||N||Q)&&(j=Qt(j+R-
(R*H+S*K)+N),V=Qt(V+S-(R*w+S*J)+Q)),(r||s)&&(L=E.getBBox(),j=Qt(j+r/100*L.width),V=Qt(V+s/100*L.height)),L="matrix("+H+","+w+","+K+","+J+","+j+","+V+")",E.setAttribute("transform",L),X&&(E.style[Ht]=L)},Vv=function(u,f,i,r,s){var
o=360,d=Pt(s),m=parseFloat(s)*(d&&~s.indexOf("rad")?Ma:1),_=m-r,g=r+_+"deg",b,A;return d&&(b=s.split("_")[1],b==="short"&&(_%=o,_!==_%(o/2)&&(_+=_<0?o:-360)),b==="cw"&&_<0?_=(_+o*wd)%o-~~(_/o)*o:b==="ccw"&&_>0&&(_=(_-o*wd)%o-~~
(_/o)*o)),u._pt=A=new ge(u._pt,f,i,r,_,Ev),A.e=g,A.u="deg",u._props.push(i),A},Pd=function(u,f){for(var i in f)u[i]=f[i];return u},Lv=function(u,f,i){var
r=Pd({},i._gsap),s="perspective,force3D,transformOrigin,svgOrigin",o=i.style,d,m,_,g,b,A,E,R;r.svg?(_=i.getAttribute("transform"),i.setAttribute("transform",""),o[Ht]=f,d=wn(i,1),Ca(i,Ht),i.setAttribute("transform",_)):
(_=getComputedStyle(i)[Ht],o[Ht]=f,d=wn(i,1),o[Ht]=_);for(m in Ml)_=r[m],g=d[m],_!==g&&s.indexOf(m)<0&&(E=ie(_),R=ie(g),b=E!==R?na(i,m,_,R):parseFloat(_),A=parseFloat(g),u._pt=new ge(u._pt,d,m,b,A-
b,rs),u._pt.u=R||0,u._props.push(m));Pd(d,r)};ve("padding,margin,Width,Radius",function(y,u){var f="Top",i="Right",r="Bottom",s="Left",o=(u<3?[f,i,r,s]:[f+s,f+i,r+i,r+s]).map(function(d){return u<2?
y+d:"border"+d+y});gf[u>1?"border"+y:y]=function(d,m,_,g,b){var A,E;if(arguments.length<4)return A=o.map(function(R){return El(d,R,_)}),E=A.join(" "),E.split(A[0]).length===5?A[0]:E;A=(g+"").split(" "),E={},o.forEach(function(R,S)
{return E[R]=A[S]=A[S]||A[(S-1)/2|0]}),d.init(m,E,b)}});var $_={name:"css",register:os,targetTest:function(u){return u.style&&u.nodeType},init:function(u,f,i,r,s){var
o=this._props,d=u.style,m=i.vars.startAt,_,g,b,A,E,R,S,N,Q,X,j,V,H,w,K,J;Ds||os(),this.styles=this.styles||Z_(u),J=this.styles.props,this.tween=i;for(S in f)if(S!=="autoRound"&&(g=f[S],!(Ue[S]&&C_(S,f,i,r,u,s)))){if(E=typeof
g,R=gf[S],E==="function"&&(g=g.call(i,r,u,s),E=typeof g),E==="string"&&~g.indexOf("random(")&&(g=jn(g)),R)R(this,u,S,g,i)&&(K=1);else if(S.substr(0,2)==="--")_=
(getComputedStyle(u).getPropertyValue(S)+"").trim(),g+="",aa.lastIndex=0,aa.test(_)||(N=ie(_),Q=ie(g)),Q?N!==Q&&(_=na(u,S,_,Q)+Q):N&&(g+=N),this.add(d,"setProperty",_,g,r,s,0,0,S),o.push(S),J.push(S,0,d[S]);else if(E!=="undefined")
{if(m&&S in m?(_=typeof m[S]=="function"?m[S].call(i,r,u,s):m[S],Pt(_)&&~_.indexOf("random(")&&(_=jn(_)),ie(_+"")||_==="auto"||(_+=He.units[S]||ie(El(u,S))||""),(_+"").charAt(1)==="="&&
(_=El(u,S))):_=El(u,S),A=parseFloat(_),X=E==="string"&&g.charAt(1)==="="&&g.substr(0,2),X&&(g=g.substr(2)),b=parseFloat(g),S in il&&(S==="autoAlpha"&&(A===1&&El(u,"visibility")==="hidden"&&b&&
(A=0),J.push("visibility",0,d.visibility),ea(this,d,"visibility",A?"inherit":"hidden",b?"inherit":"hidden",!b)),S!=="scale"&&S!=="transform"&&(S=il[S],~S.indexOf(",")&&(S=S.split(",")[0]))),j=S in Ml,j)
{if(this.styles.save(S),E==="string"&&g.substring(0,6)==="var(--"&&(g=ke(u,g.substring(4,g.indexOf(")"))),b=parseFloat(g)),V||
(H=u._gsap,H.renderTransform&&!f.parseTransform||wn(u,f.parseTransform),w=f.smoothOrigin!==!1&&H.smooth,V=this._pt=new ge(this._pt,d,Ht,0,1,H.renderTransform,H,0,-1),V.dep=1),S==="scale")this._pt=new ge(this._pt,H,"scaleY",H.scaleY,(X?
Au(H.scaleY,X+b):b)-H.scaleY||0,rs),this._pt.u=0,o.push("scaleY",S),S+="X";else if(S==="transformOrigin"){J.push(pe,0,d[pe]),g=Xv(g),H.svg?hs(u,g,0,w,0,this):(Q=parseFloat(g.split(" ")
[2])||0,Q!==H.zOrigin&&ea(this,H,"zOrigin",H.zOrigin,Q),ea(this,d,S,pf(_),pf(g)));continue}else if(S==="svgOrigin"){hs(u,g,1,w,0,this);continue}else if(S in k_){Vv(this,H,S,A,X?Au(A,X+g):g);continue}else if(S==="smoothOrigin")
{ea(this,H,"smooth",H.smooth,g);continue}else if(S==="force3D"){H[S]=g;continue}else if(S==="transform"){Lv(this,g,u);continue}}else S in d||(S=Nu(S)||S);if(j||(b||b===0)&&(A||A===0)&&!Av.test(g)&&S in d)N=
(_+"").substr((A+"").length),b||(b=0),Q=ie(g)||(S in He.units?He.units[S]:N),N!==Q&&(A=na(u,S,_,Q)),this._pt=new ge(this._pt,j?H:d,S,A,(X?Au(A,X+b):b)-A,!j&&(Q==="px"||S==="zIndex")&&f.autoRound!==!1?
Mv:rs),this._pt.u=Q||0,N!==Q&&Q!=="%"&&(this._pt.b=_,this._pt.r=xv);else if(S in d)Bv.call(this,u,S,_,X?X+g:g);else if(S in u)this.add(u,S,_||u[S],X?X+g:g,r,s);else if(S!=="parseTransform"){ps(S,g);continue}j||(S in d?
J.push(S,0,d[S]):typeof u[S]=="function"?J.push(S,2,u[S]()):J.push(S,1,_||u[S])),o.push(S)}}K&&G_(this)},render:function(u,f){if(f.tween._time||!Rs())for(var i=f._pt;i;)i.r(u,i.d),i=i._next;else
f.styles.revert()},get:El,aliases:il,getSetter:function(u,f,i){var r=il[f];return r&&r.indexOf(",")<0&&(f=r),f in Ml&&f!==pe&&(u._gsap.x||El(u,"x"))?i&&Zd===i?f==="scale"?Uv:Rv:(Zd=i||{})&&(f==="scale"?Nv:Cv):u.style&&!ys(u.style[f])?
zv:~f.indexOf("-")?Dv:Ms(u,f)},core:{_removeProperty:Ca,_getMatrix:Ns}};Se.utils.checkPrefix=Nu;Se.core.getStyleSaver=Z_;(function(y,u,f,i){var r=ve(y+","+u+","+f,function(s){Ml[s]=1});ve(u,function(s)
{He.units[s]="deg",k_[s]=1}),il[r[13]]=y+","+u,ve(i,function(s){var o=s.split(":");il[o[1]]=r[o[0]]})})
("x,y,z,scale,scaleX,scaleY,xPercent,yPercent","rotation,rotationX,rotationY,skewX,skewY","transform,transformOrigin,svgOrigin,force3D,smoothOrigin,transformPerspective","0:translateX,1:translateY,2:translateZ,8:rotate,8:rotationZ,8:rot
ateZ,9:rotateX,10:rotateY");ve("x,y,z,top,right,bottom,left,width,height,fontSize,padding,margin,perspective",function(y){He.units[y]="px"});Se.registerPlugin($_);var ds=Se.registerPlugin($_)||Se;ds.core.Tween;const Zv=()=>{const
y="imIronMan",u="imNotMid{i5_thi5_w3b_0r_crypt0}";let f=[];{const E=u.length;f=[...u].map(R=>R.charCodeAt(0)^E)}const[i,r]=bu.useState(""),[s,o]=bu.useState([]),[d,m]=bu.useState([]),[_,g]=bu.useState(!1);bu.useEffect(()=>{const
E=JSON.parse(localStorage.getItem("xordleInputs"))||[];m(E)},[]);const b=E=>{r(E.target.value)},A=E=>{E.preventDefault();const R=i.length,S=y.length,N=
[...d,i].slice(-100);if(localStorage.setItem("xordleInputs",JSON.stringify(N)),m(N),R!==S){g(!1);const Q=[...i].map(X=>X.charCodeAt(0)^R);o(Q.map(X=>({value:X,correct:!1})))}else{const Q=[...i].map((X,j)=>
({value:X.charCodeAt(0)^R,correct:X===y[j]}));o(Q),g(i===y)}ds.fromTo(".result",{opacity:0},{opacity:1,duration:1}),r(""),s.forEach((Q,X)=>{ds.fromTo(`.result-box-${X}`,{scale:0},{scale:1,duration:.5,delay:X*.1})})};return
Lt.jsxs("div",{className:"min-h-screen bg-gradient-to-br from-blue-50 to-purple-100 flex flex-col items-center justify-center py-8 px-2",children:[Lt.jsxs("div",{className:"w-full max-w-md bg-white rounded-2xl shadow-xl p-8 flex flex-
col items-center",children:[_&&Lt.jsx("div",{className:"mb-4 w-full",children:Lt.jsxs("div",{className:"text-xs text-gray-500 font-mono break-all",children:["Flag : [",f.join(", "),"]"]})}),Lt.jsx("span",{className:"text-
white",children:"imNotMid"}),Lt.jsx("h1",{className:"text-3xl font-extrabold text-purple-700 mb-6 tracking-tight drop-shadow",children:"Welcome Folks"}),Lt.jsxs("form",{onSubmit:A,className:"w-full flex flex-col items-center gap-
3",children:[Lt.jsx("input",{type:"text",value:i,onChange:b,className:"w-full border-2 border-purple-300 rounded-lg p-3 text-lg focus:outline-none focus:border-purple-500 transition",placeholder:"Enter your
guess",autoFocus:!0}),Lt.jsx("button",{type:"submit",className:"w-full bg-gradient-to-r from-purple-500 to-blue-500 text-white font-semibold py-2 rounded-lg shadow hover:from-purple-600 hover:to-blue-600
transition",children:"Submit"})]}),Lt.jsx("div",{className:"result mt-8 flex flex-wrap justify-center gap-3 w-full",children:s.map((E,R)=>Lt.jsx("div",{className:`result-box result-box-${R} w-14 h-14 flex items-center justify-center
text-xl font-bold rounded-lg shadow transition-all border-2 ${E.correct?"bg-green-100 border-green-400 text-green-700":"bg-red-100 border-red-400 text-red-700"}`,children:E.value},R))}),Lt.jsxs("div",{className:"previous-inputs mt-10 w-
full",children:[Lt.jsx("h3",{className:"font-bold text-gray-700 mb-2 text-lg",children:"Previous Inputs"}),Lt.jsx("div",{className:"bg-gray-50 rounded-lg shadow-inner p-3 max-h-40 overflow-y-auto",children:Lt.jsxs("ul",
{className:"space-y-1",children:[d.length===0&&Lt.jsx("li",{className:"text-gray-400 italic",children:"No previous inputs yet."}),d.map((E,R)=>Lt.jsx("li",{className:"text-gray-700 break-all",children:E},R))]})})]})]}),Lt.jsxs("footer",
{className:"mt-8 text-gray-400 text-xs",children:["© ",new Date().getFullYear()," Kludge CTF Challenge"]})]})});Dy.createRoot(document.getElementById("root")).render(Lt.jsx(bu.StrictMode,{children:Lt.jsx(Zv,{})}));
```



```python
from Crypto.Util.number import inverse, long_to_bytes
from math import gcd

N1 = 14974374473875195262766359759714498503053089849989142350615906604375075459239382918
114868001552711501100264166512783155193243369424875166451614237308755177787627661679153
407311577363068061643400744101086494410390726961463235329017993736316198284311340752292

N2 = 11377303457963693647634556693776352306798805099525254752031901804678648636881088711
926477688508021301745074012078147126558540520284378771368173461658671664570694525146838
466511586254342858049935950702182387012147757215640939517729687181123762690670586402035
e = 65537
c = 50620835015367606193759766222367800122803283397748905948654857591458757240749334674
583037165558587800714681619427000917843507584399069078356121466246261003271175122323554
884959208420406373053914635940047237068268327521369299948583610143571896073590221095836

p = gcd(N1, N2)
if p == 1:
    print("N1 and N2 are coprime")
    exit()

q = N1 // p  # Floor division
phi = (p - 1) * (q - 1) # Euler Totient Function

d = inverse(e, phi) # Private Key

m = pow(c, d, N1) # Calculating coded message (using cypher text, N, private key)
plaintext = long_to_bytes(m) # decoding coded message

print(plaintext)
```

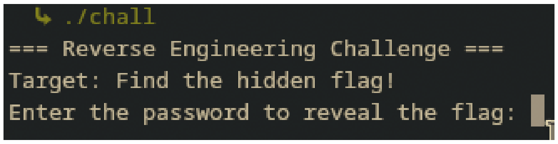
On running the code we get the flag,

ImNotMid{r54_!s_n0t_50_c00l_4nym0r3_n1994}

# 4 Reverse Engineering

## 4.1 JJK

We are given only a binary executable, so on running it we get,

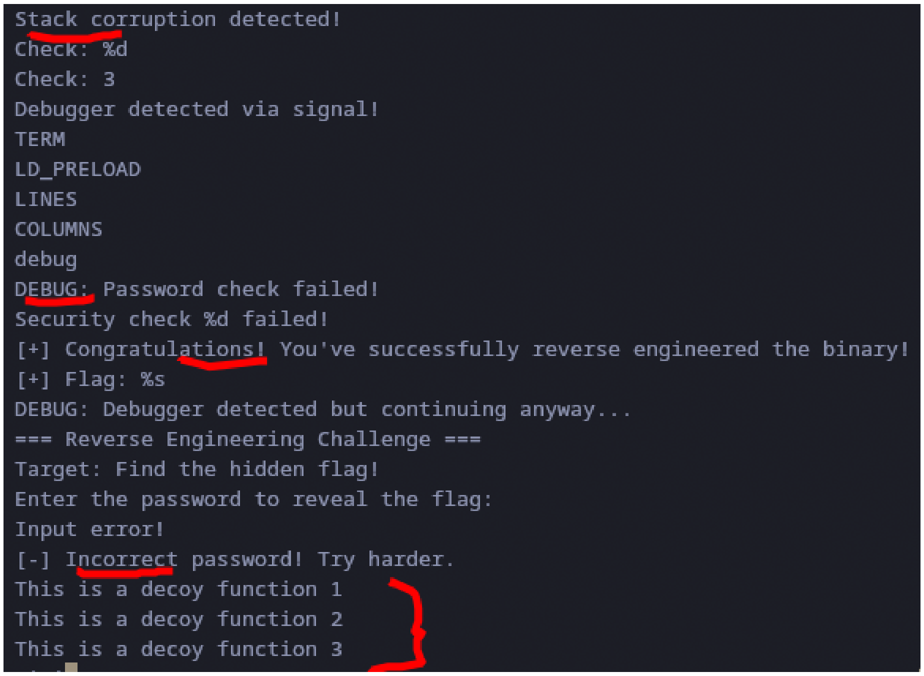


```
  ↳ ./chall
=== Reverse Engineering Challenge ===
Target: Find the hidden flag!
Enter the password to reveal the flag:
```

Also on running the command

```
strings chall > jjk.txt
```

we can observe a few lines of interest,



```
Stack corruption detected!
Check: %d
Check: 3
Debugger detected via signal!
TERM
LD_PRELOAD
LINES
COLUMNS
debug
DEBUG: Password check failed!
Security check %d failed!
[+] Congratulations! You've successfully reverse engineered the binary!
[+] Flag: %s
DEBUG: Debugger detected but continuing anyway...
=== Reverse Engineering Challenge ===
Target: Find the hidden flag!
Enter the password to reveal the flag:
Input error!
[-] Incorrect password! Try harder.
This is a decoy function 1
This is a decoy function 2
This is a decoy function 3
```

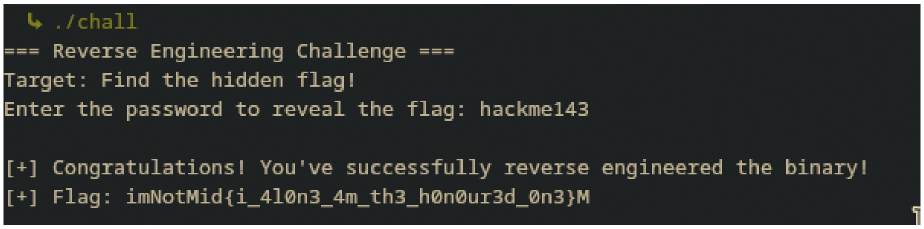
So we can infer that we will obtaiin the flag upon entering the right password. Since we are given nothing else, we must obtain the passphrase from the binary executable. Firstly, I disassembled the binary executable into assembly code using the command,

```
objdump -D chall > chall.asm
```

We can analyze it in even more detail using a tool like Ghidra (which even gives us the c-code behind the assembly function). We can tell on analyzing the assembly code that there mainly a few functions of interest,

- main
- verify_password: returns 1 if input matches password
- compare: compares user input and decoded obfuscated key
- decoded_string: obfuscated key is encoded

We observe that obfuscated key is held at the memory location 00104080 and holds the value 25 $2c$ $2e$ 26 20 28 $7c$ 79 $7e$ 00 00 00 00 00 00 00. Encoding scheme is to XOR each 4-bit hexadecimal number with $0x4d$ i.e. 77 (in decimal). On decoding we get the passphrase to be, ”hackme143”.



```
  ↳ ./chall
=== Reverse Engineering Challenge ===
Target: Find the hidden flag!
Enter the password to reveal the flag: hackme143

[+] Congratulations! You've successfully reverse engineered the binary!
[+] Flag: imNotMid{i_4l0n3_4m_th3_h0n0ur3d_0n3}M
```

Figure 1: ”Throughout Heaven and Earth, I Alone Am The Honored One”, Satoru Gojo

# 5 Stegnography

## 5.1 99.9 % truth

I tried running stegseek (with rockyou.txt), gave nothing.

## 5.2 Osint

### 5.2.1 Dora

I got parts of the flag, ” imNotMid051nt_1 ” and a file _neverTEL_ and a README and no idea what to do after.