攻防世界crypto高手题之OldDriver

原创

沐一 · 林 已于 2022-04-12 20:34:51 修改 122 收藏

分类专栏： CTF 密码学 文章标签： ctf

于 2021-09-13 11:00:36 首次发布

本文链接：https://blog.csdn.net/xiao__1bai/article/details/120262433

CTF 同时被 2 个专栏收录

167 篇文章 6 订阅

订阅专栏

密码学

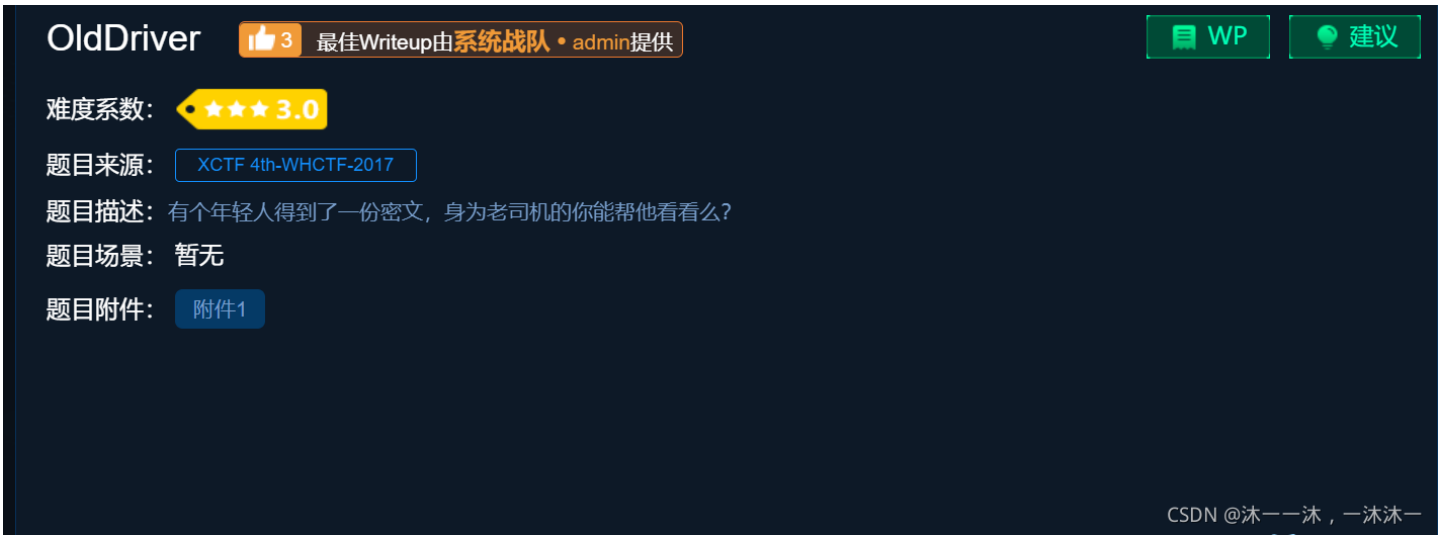
51 篇文章 1 订阅

订阅专栏

攻防世界crypto高手题之OldDriver

继续开启全栈梦想之逆向之旅~

这题是攻防世界crypto高手题的OldDriver



下载附件，一个.txt文件，打开，发现是c、n、e的RSA类型题：

[{"c": 73660675747411714617220651332429160804955059136632503300827474653836768939704114765507483948414
{"c": 21962825323300469151795920289886886562790942771546858500842179806566435767103803978885148772139
{"c": 65696894202740669578359833905835852865700876190481101411877005841937926952354050778115443551692
{"c": 45082461680445135184524938827135363906367415415518058217903389737976159712718672485843798131141
{"c": 22966105670291282335588843018244161552764486373117942865966904076191122337435542553276743938817
{"c": 17963313063405045742968136916219838352135561785389534381262979264585397896844470879023686508540
{"c": 16524175347090294503805706539737053209861176795975638730226831408005074825604829483101315409482
{"c": 15585771734488351039456631394040497759568679429510619219766191780807675361741859290490732451112
{"c": 89651234216376940500442168445233791633474780291248150328328132250507325585242396606487462848841
{"c": 13560945756543023008529388108446940847137853038437095244573035888531288577370829065666320069397

.

.

(这里积累第一个经验)

上网查了查，发现类型是`低加密指数广播攻击`，这里附上别人的话：(里面的中国剩余定理我还没看懂)

> 首先介绍什么是广播，加入我们需要将一份明文进行多份加密，但是每份使用不同的密钥，密钥中的模数n不同但指数e相同且很小，我们只要拿到多份密文和对应的n就可以利用中国剩余定理进行解密。关于**中国剩余定理**请参考文章：
> `https://www.cnblogs.com/freinds/p/6388992.html`
> .
> 只要满足以下情况，我们便可以考虑使用低加密指数广播攻击：
> .
> 加密指数e非常小
> 一份明文使用不同的模数n，相同的加密指数e进行多次加密
> 可以拿到每一份加密后的密文和对应的模数n、加密指数e
> .
>
> RSA秘钥的指数e=10，这些特征暗示低加密指数广播攻击；当加密一份消息给多人时，
> $c_i = m^e \bmod N_i, i = 1, \cdots, k$，当$k \geq e$时，可由中国剩余定理求解$m^e$，再开方即得明文$m$。

.

.

.

(这里积累第二个经验)

本来想参考着写一个脚本的，发现`CTF-RSA-tool`工具里面好像可以解决`低加密指数广播攻击`。附上我以前的笔记：

Basic Broadcast Attack(低加密指数广播攻击)

python2 solve.py --verbose -i examples/Basic_Broadcast_Attack.txt

```
c : 6867940996691870031530411714485906844552818193325528906319305401428815108346680759433216763381096732182463314444
e : 7
n : 24810910852704603048663349011054669655631146433543459534796438815331335687309113943583212235150241971378068933 1

c : 1117905220184329685115491669660129193823567641747584717324745389247133369892051586020708442213591696394252275222
e : 7
n : 4712783910529936103379120873779889977678125538150303038168690908215575736101910410328062054071689469913314217311

c : 4011807694373533755953737969298207094392151534800021109759935626333076007590674837412972752674043888369509450311
e : 7
n : 4313429171104682135845535135888408777702100383947029650599045058170621937935627239179422012903689519987338580251

c : 1264907659222264937119216486904402540823137171762778004621934637785202454433705015265267657712234253486895809171
e : 7
n : 1930083892114922100729894488747859908280022904521927160627203810397065655994391419728165415858746873054182830641

c : 2889908993543526758823589751984612039343321411434152123869638412250731689945732705502954697233328145256398483841
e : 7
n : 3075412148882763569297184959926774937507794918255030314572932537531492640190578383093162873865887932017994488001

c : 1408662941385567240363983067611804246584602032014382331881504807036850568420814165260359623496096870321778847291
e : 7
n : 3043047798347042619563114265966807177225664120552592989198587299611585801074464877937098353994218768919240651741

c : 2004929920758895590715527609578791340258965237913415140334036049837189311985595783357644385653403788629873170191
e : 7
n : 3548927512653680597428163594290748046391608966306912977142054861[redacted]
```

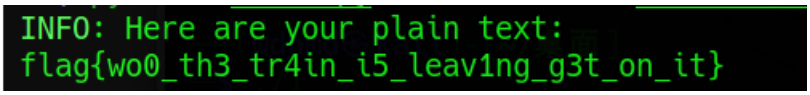

.

.

那么把题目的`enc.txt`照着格式摆放，然后运行脚本即可：

```
c : 7366067574741171461722065133242916080495505913663250330082747465383676893970411476550748394841437418105312353971095003424322679616940371123028982189502042
e : 10
n : 2516250705233971442183968887373459617775112403672383100330095976113781149071520574294173840654815024086177930178413365216590822791741548313758538898627480 3


c : 2196282532330046915179592028988688656279094277154685850084217980656643576710380397888514877213930548431968824936899950378444150738347609594625801131795146 1
e : 10
n : 2397685958990441979832081209768185865232547379189123271043199720289781958063493707090062521321809533076687719021241802329734173280883948830855112640998319 3

c : 6569689420274066957835983390583585286570087619048110141187700584193792695235405077811544355169290382357149374107076406086154103351897890793598997687053983
e : 10
n : 1850378283685854004397455803560165461094891550564521982015025106230512014874554590656754865019183209082348285260434647833535378450107676192260536184870362 3

c : 4508246168044513518452493882713536390636741541551805821790338973797615971271867248584379813114125478195284692695928668946553625483179633266057122967547052
e : 10
n : 2338308747854551221871315793293474611072170681907742341806022008365771342850358280190980714280264736799428975015595100541168367083097506193809451365010723
```

```
c : 22966105670291282335588843018244161552764486373117942865966904076191122337435542553276743938817686729554714315494818922753880198945897222422137268427611672
e : 10
n : 31775649089861428671057909076144152870796722528112580479442073365053916012507273433028451755436987054722496057749731758475958301164082755003195632005308493

c : 17963313063405045742968136916219838352135561785389534381262979264585397896844470879023686508540355160998533122970239261072020689217153126649390825646712087
e : 10
n : 22246342022943432820696190444155665289928378653841172632283227888174495402248633061010615572642126584591103750338919213945646074833823905521643025879053949

c : 16524175347090294503805706539737053209861176795975638730226831408005074825604829483101315409482277970455053903331461915867492692495481682473164040740146 39
e : 10
n : 25395461142670631268156106136028325744393358436617528677967249347353524924655001151849544022201772500033280822372661344352607434738696051779095736547813043

c : 15585771734488351039456631394040497759568679429510619219766191780807675361741859290490732451112648776648126779759368428205194684721516497026290981786239352
e : 10
n : 32056508892744184901289413287728039891303832311548608141088227876326753674154124775132776928481935378184756756785107540781632570295330486738268173167809047

c : 89651234216376940500442168445233791633474780291248150328328132250507325585242396606487462848841407467888236818860105773422548410145945700674679056823597 97
e : 10
n : 52849766269541827474228189428820648574162539595985395992261649809907435742263020551050064268890333392877173572811691599841253150460219986817964461970736553

c : 13560945756543023008529388108446940847137853038437095244573035888531288577370829065666320069397898394848484847030321018915638381833935580958342719988978247
e : 10
n : 30415984800307578932946399987559088968355638354344823359397204419191241802721772499486615661699080998502439901585573950889047918537906687840725005496238621
```



.

.

(这里积累第三个经验)

积累一下别人的脚本以备后用：(python2)

```
import libnum
import gmpy2
dic =[{"c":736606757474117146172206513324291608049550591366325033008274746538367689397041147655074839484143741810531235971095003424322679616940371123028982189502042, "e": 10, "n":251625070523397144218396888737345961777511240367238310033009597611378114907152057429417384065481502408617793017841336521659082279174154831375853889862748037}
,
{"c":219628253233004691517959202898868865627909427715468585008421798065664357671038039788851487721393054843196882493689995037844415073834760959462580113179514611, "e": 10, "n":239768595899044197983208120976818586523254737918912327104319972028978195806349370709006252132180953307668771902124180232973417328088394883085511264099831937}},
{"c":656968942027406695783598339058358528657008761904811014118770058419379269523540507781154435516929038235714937410707640608615410335189789079359899768705398 3, "e": 10, "n":185037828368585400439745580356016546109489155056452198201502510623051201487455459065675486501918320908234828526043464783353537845010767619226053618487036237}},
{"c":450824616804451351845249388271353639063674154155180582179033897379761597127186724858437981311412547819528469269592866894655362548317963326605712296754705 2, "e": 10, "n":233830874785455122187131579329347461107217068190774234180602200836577134285035828019098071428026473679942897750155951005411683670830975061938094513650107237}},
{"c":229661056702912823355888430182441615527644863731179428659669040761911223374355425532767439388176867295547143154948189227538801989458972224221372684276116 72, "e": 10, "n":317756490898614286710579090761441528707967225281125804794420733650539160125072734330284517554369870547224960577497317584759583011640827550031956320053084937},
{"c":179633130634050457429681369162198383521355617853895343812629792645853978968444708790236865085404355160998533122970239261072020689217153126649390825646712087, "e": 10, "n":222463420229434328206961904441556652899283786538411726322827888174495402248633061010615572642126584591103750338919213945646074833823905521643025879053949},
{"c":165241753470902945038057065397370532098611767959756387302268314080050748256048294831013154094822779704550539033146191586749269249548168247316404074014639, "e": 10, "n":253954611426706312681561061360283257443933584366175286779672493473535249246550011518495440222017725000332808223726613443526074347386960517790957365478130437, "e": 10, "n":155857717344883510394566313940404977595686794295106192197661917808076753617418592904907324511126487766481267797593684282051946847215164970262909817862393 52, "e": 10, "n":320565088927441849012894132877280398913038323115486081410882278763267536741541247751327769284819353781847567567851075407816325702953304867382681731678090477, "e": 10, "n":896512342163769405004421684452337916334747802912481503283281322505073255852423966064874628488414074678882368188601057734225484101459457006746790568235979 7, "e": 10, "n":528497662695418274742281894288206485741625395959853959922616498099074357422630205510500642688903333928771735728116915998412531504602199868179644619707365537}},
{"c":135609457565430230085293881084469408471378530384370952445730358885312885773708290656663200693978983948484847030321018915638381833935580958342719988978247, "e": 10, "n":304159848003075789329463999875590889683556383543448233593972044191912418027217724994866156616990809985024399015855739508890479185379066878407250054962386217}}]
n = []
C = []
for i in dic:
    n.append(i["n"])
    C.append(i["c"])

N = 1
for i in n:
    N *= i

Ni = []
for i in n:
    Ni.append(N // i)

T = []
for i in range(10):
    T.append(int(gmpy2.invert(Ni[i], n[i])))

X = 0
for i in range(10):
    X += C[i] * Ni[i] * T[i]

m10 = X % N
m = gmpy2.iroot(m10, 10)
print (libnum.n2s(m[0]))
```

.
.
.
总结：

> 1：
> (这里积累第一个经验) 上网查了查，发现类型是 `低加密指数广播攻击` ，这里附上别人的话：(里面的中国剩余定理我还没看懂)
> 首先介绍什么是广播，加入我们需要将一份明文进行多份加密，但是每份使用不同的密钥，密钥中的模数n不同但指数e相同且很小，我们只要拿到多份密文和对应的n就可以利用中国剩余定理进行解密。关于**中国剩余定理**请参考文章：
> `https://www.cnblogs.com/freinds/p/6388992.html`
> .
> 只要满足以下情况，我们便可以考虑使用低加密指数广播攻击：
> .
> 加密指数e非常小
> 一份明文使用不同的模数n，相同的加密指数e进行多次加密
> 可以拿到每一份加密后的密文和对应的模数n、加密指数e
> .
>
> RSA秘钥的指数e=10，这些特征暗示低加密指数广播攻击；当加密一份消息给多人时，
> $c_i = m^e \bmod N_i, i = 1, \cdots, k$，当$k \geq e$时，可由中国剩余定理求解$m^e$，再开方即得明文$m$。

> 2：
> (这里积累第二个经验) 本来想参考着写一个脚本的，发现 `CTF-RSA-tool` 工具里面好像可以解决 `低加密指数广播攻击` 。

> 3：
> (这里积累第三个经验)
> 积累一下别人的脚本以备后用。

解毕！
敬礼！