实验吧 密码学 picked WriteUp

原创

FrancisQiu 于 2019-02-25 11:05:06 发布 6151 收藏

分类专栏： CTFwriteup crypto CTF 实验吧 密码学

本文链接：https://blog.csdn.net/qq_40737798/article/details/87902272

题目

> We just found a dead robot. It seems there is some useful data left but somehow it got confused with other data and now we don't know what's useful and what's junk. We just know there is only one way to go but there are many dead ends.
> Announcements:
> Think outside the box - being several types at once like an animal that can change its color. Excuse the inaccuracy, but that's what you're searching for.
> 解题链接： http://ctf5.shiyanbar.com/crypto/Packed/packed.rar

思路

下载packed.rar后，我们发现这是个二进制文件。用010 editor打开，看到以下内容，并分析：

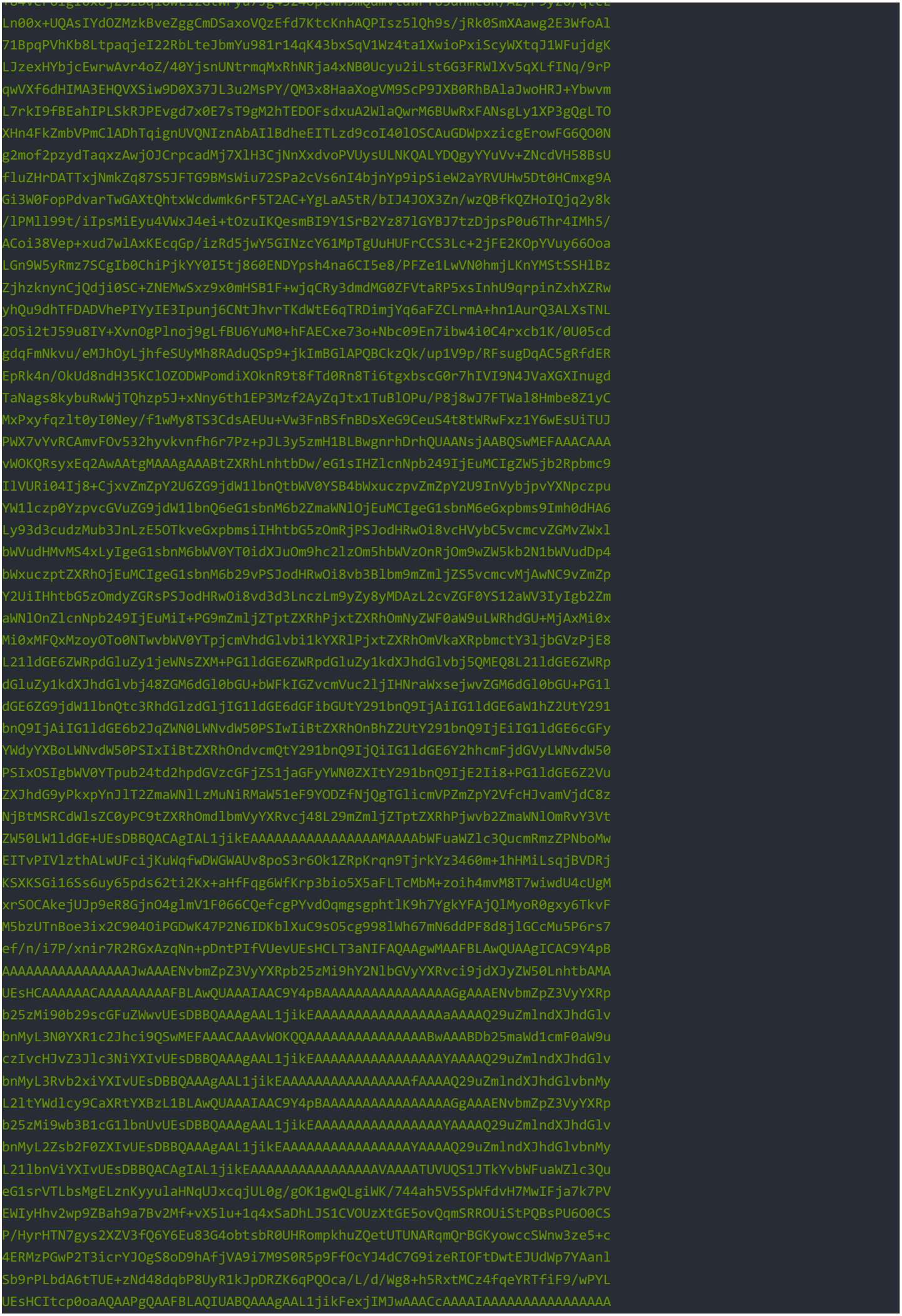
```
#disabled-encoding: _rot__..._13|....
#第一行太长，简单地说就是凸显了rot13这个关键词

"""%PDF-102.98
#此处为PDF文件内容
1 0 obj <</Type /Catalog /Pages 2 0 R>>
endobj
2 0 obj <</Type /Pages /Kids [3 0 R] /Count 1>>
endobj
3 0 obj<</Type /Page /Parent 2 0 R /Resources 4 0 R /MediaBox [0 0 500 800] /Contents 6 0 R>>
endobj
4 0 obj<</Font <</F1 5 0 R>>>>
endobj
5 0 obj<</Type /Font /Subtype /Type1 /BaseFont /Helvetica>>
endobj
```

```
6 0 obj
<</Length 44>>
stream
BT /F1 24 Tf 175 720 Td (no hint given)Tj ET
endstream
endobj
xref
0 7
0000000000 65535 f
0000000009 00000 n
0000000056 00000 n
0000000111 00000 n
0000000212 00000 n
0000000250 00000 n
0000000317 00000 n
trailer <</Size 7/Root 1 0 R>>
startxref
406
%%EOF
"""
#这一段内容十分特殊，通过分析可以发现：这一段其实是rot-13加密内容
pvcure="U51\\\'Hk2W&+(3M;Hkpk0Kkf\k13u\k014$I!E($E>\g/)E!\k01<.\k13,A-nC4Z4nEhT1-IhH0 ThU+n@0J=3E9\k01>(_0\k01,8
P0Ek ThA6\"I|\k1rmXM3\k014$]}E!2\k1q4F?7\k1nh\k1skf\g_\k01kn\k13<Tk)E&Vc2W&\k0s93G#mw\k1p\k1nc\k13ex\k00t\k01r|\
k13t\k19wh\k0on\k18wg\k02b+kn\k13h\k01kn\k13%F1/Th\k03\k1o.\\:A7.\\:A4b\k13\k0pA-3\k133Z9&\k13<Ek N2JwvM{QinK0Kw
u\k136A6\"E!\k01\k07eP0c\k138n\k1qp22vrh\k161Sj+=-@0\k1oEn\k13h\k01(3M;HkpE\'S.f\k1p>Q!f\k13<Ek,M&E1/Gj+E"
a =0 ;vzcbeg unfuyvo ,flf ;
gel :xrl =flf .neti [1 ]
rkprcg VaqrkReebe :flf .rkvg ("k\k9p\ks3A\knqG0G\kp8\kpq,.\kpr\kppXJ\kp8\kppFU,W/\k03\k00Z\k97\k07\\".qrpbqr ("z
ip"))
s =trgngge (unfuyvo ,"k\k9p\kpoZ1\k05\k00\k02T\k01\k07".qrpbqr ("zip"))
juvyr a <(5 *10 **6 ):xrl =(s (xrl ).qvtrfg ());a =a +1
xrl =xrl [:5 ].hccre ()
juvyr yra (xrl )<yra (pvcure ):xrl =xrl *2
cynva ="".wbva (znc (pue ,[beq (n )^beq (o )sbe n ,o va mvc (pvcure ,xrl )]))
gel :rkrp cynva
rkprcg :cevag "k\k9p\k0o/\kpn\kpsXJ\ks0A\knqG\k04\k00\k14q\k03k".qrpbqr ("zip"), erce(cynva)
#下面这段内容发现与base64内容很类似，但是通过base64解码却发现解码失败。
'''UEsDBBQAAAgAAL1jikFexjIMJwAAACcAAAAIAAAAbWltZXR5cGVhcHBsaWNhdGlvbi92bmQub2Fz
aXMub3BlbmRvY3VtZW50LnRleHRQSwMEFAAACAAAvWOKQemoBXVmBAAAZgQAABgAAABUaHVtYm5h
aWxzL3RodW1ibmFpbC5wbmeJUE5HDQoaCgAAAA1JSERSAAAAxgAAAQAIAgAAAGc/nhcAAAQtSURB
VHic7dJNi5ZlGIDheZ0ZmhklGWt8JTAiMwiZwFVRaNRK+0JaGAS1bhXUIn9Ae6HfEC0CoS/btKno
D4TZIqaIsjJTzNHUGUfnbaIW0jLOmGqOY/U81/08cN1wToxGozHoTKz3AvzfSIqYpIhJipikiEmK
mKSISYqYpIhJipikiEmKmKT+cO3nhbNLV8+e/Ojk4ui2x146ONx00+Hq
6eOvf7X/5Qcuf35mdn7n1O8fL87unpscG7uxvPjjlxfn7l8b8qcNn9T1H94/+t7McwfOv/XJypaZ
LVPTm8fG1xp697Vj30/fPhxOjlZn9z2/d+rqiTePnh1uunDu2OWVma0TK1ufOrLvmzc+OHF6esfm
i+eOT9517/RP3332ztsr+w4deOGVR7cN1vte62fDJzWYuXt+8OHX1/dMXT916Zelq4PBYLS8uml8
evtwcP7yje2bB6PR6NrS+Ny08QuLv15ZuWX7cOLipcHqyupgaeHTL259Ym55y4MPn/p48vGdCwv3
zA/vGFy4sjq2bXy977V+NnxS47N7Dr64Z+3hkd03j+989dmbX588cvivP9448+3eZ54eLe86dHjX
LWPza5OH7jv0z+7637Dhk/rbxof7Dx9Y7yX+jSRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxS
xCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVM
UsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLE
JEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRF
TFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxS
xCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQk
RUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVM
UsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLE
JEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRF
TFLEJEVMUsQkRUxSxCRFTFLEJEVMUsQkRUxSxCRFTFLEJEVMUsR+Awfwg+/LbFFaAAAAAElFTkSu
```

QmCCUEsDBBQACAgIAL1jikEAAAAAAAAAAAAAAAALAAAAY29udGVudC54bWylV99v2yoUft9fYXnS
3ihJsys1XpO9TNOu1ErT2k17pRg7bBh8AcfJf38POCY4rRtXeUkE5/vOOZxf4NvPu0okW6YNV3KV
zq9macIkVTmX5Sr9+fgV3aSf1+9uVVFwyrJc0aZi0iKqpIX/BNjSZJ10lTZaZooYbjJJKmYySzNV
M9mzshideVvdjrF7MZnuwTHbsp2dSnbYAZc8TbfswTE716SdSnZYCGpML9RU8s4IVCiIelUTy0+8
2Aku/67SjbV1hnHbtlft4krpEs+XyyX20uAwDbi60cKjcoqZYM6YwfOrOe6xFbNkqn8OG7skm+qJ
6cmhIZY8y6rZlpMrYluOhIZuiJ5cGx48TO8in57eRR5zK2I3Izm5wfcg9D/3d8da0NVUWw47CBXV
vJ58zA4d85VSwVVH6BrUu3s9m33E3TpCt6/CW80t0xGcvgqnRNAQcVW9FDTAzTEgENu6Mg2F7wJh
RgjXuBMHsMlHVf++v3ugG1aRI5ifByMujSXyGBntkjB60n+wZrXSNgSmmD4wIVvXwbeNrcR4uztp
Dy11nr8IBXcWGFofGg9tOWvfp4NJ/npBLE8Kwo/FcxQPCm0JKT8ObV2Ge6VQjQSn4C46BITtaqa5
ExHhadlAw2Cacib6dg0OvKQGXEWVgexBlao6i9jD4ayr3TR1rjJVXpxqPOlSaszCvpSMxx/YyZC7
m2D6HixFd/J1uu4v4K55DQ4bBVzEqCCUoZxRYda33SAN20m3dn6v0ju14Tb5xmXO52kCY7PHVVzs
V+kHUivzKQJ1Gyk+o5XDrPdJSx4gLMW45hPgQX0yUO4IqGQSADA3tKqIHCBqbilM1i3R3JfUG5wj
0kzyDXATXDMtN+YS176wP+RX87pbEWaKS3tjWXVRuI7Jf0OBXO4THivnwz5pLFSC5RR5PaHO/e/g
BN/nwdbB65poUmpSb3oBbLg3rF+gjvUAgzwnOk97xYGEamhppi1nJimUf0AiIngJnUmZ9Bddx/jT
GMuLPTLwygO9rdIwjQoiTBR0z44UhnEb2YO2B+JsNnta3CyWPjjRWccP/vjs4M5aet70OYN4NAcH
wZPK92Hh7Kxv/TvbsP8a+JgIuXy+mfitnJtakD1SjYXHKkMCrnm4tmByenF3vH+FgAB3Lep8vEjZ
Y193l2lx8b1UyZfuu8CHfTxqdUeJSxYKvTddE/lcDvWwhooUIpEq2cDdlJQcnk+9FeD0Futjjrvs
4UFi8cjH3/p/UEsHCD9pUz6mAwAAPQ4AAFBLAwQUAAgICAC9Y4pBAAAAAAAAAAAAAAAACgAAAHN0
eWxlcy54bWzdWkuP2zYQvvdXGCraG23LXu/abpwe+kALJCnQpr0WtERZbChRICk/8us7JEWJsiWv
Eid7cBYIYM5wOPzmm+FLr348Zmy0J0JSnm+CcDwNRiSPeEzz3Sb4+/2vaBn8+PqbVzxJaETWMY/K
jOQKSXViRI6gcy7XVrgJSpGvOZZUrnOcEblW0ZoXJHed1r722gxlW4yxod2Nst9bkaMa2lnrtvri
7fCRjbLf0xb4MLSz1gVM/e4JH9r5KBlKOIp4VmBFz7w4Mpp/2ASpUsV6MjkcDuPDfMzFbhKuVqu
yu3wzuPh4Z3Hft8Mq7QnJsvJWxCa/96+abggsqFjad0WVJGgxeBpWm2/P+e8dlV3sAlq3J1Npw8T
+9vTPlxVPwiqiPDUo6vqEWZRjTjPukADvXACGojsNU2dttCT7rW8mAhScKFqR5LhBQrQmdXplaqM
9aeXljrVnYjjTlVVz6BVAOizol2NVlPOF1Dk5B6a8
IceCCKpFmJlu65aFVm5JOVddk33/50TLkK61UE2qcu8tMbPgtVtPEg5rSYIjgmISMfn6la0DdfPI
/tbObYI3PKVq9BvNYxoGI8h6p5dRdtoE3+OCyx88JdsQTJ6xSqFUGQxGf8Hck37LZ4qV+VHLuO6A
diQHBaC94BnOWxoFVREUhj0W1ETToE5zDuRzkG+gNcE0eqJS3uPYz+Q//U153y9MZ4tJJKpLdBFcT
/E8gyO0+TfroXLXb3ZLzPSYJLlm1h3KWKx93AhcpjQKnW/1GhYAsF4rCnkvPSyrBPxBY9RiHZfbb
5VT/BSO9yVgnlLFaEiXkKVkEo4SvD2AK8UKZLMw50r+rLjLFMT8g8FYShY6bYDoOw2VI80756VKu
YOlEsNMgSBY4gn0OSrmgH7muI1Y7fLimvddzizp0oTgPtnuh22W1gpvBdA5UpcjuHhPMpMeuAgts
kG/hbkRaH+FScT0IUI7GhFtVzIoUumGMH1tBMOzMIFg0Uk6il0ftXMZj6M4EUtsWvYCYRK9Jepft
z8Y56XyECg8M4oXU/Ot3u1bXfl/MppQEYMh1cM3gFW2UKInhjGmU9CN4Gs4KZdoYzncl3kFTbJUi
WEWUAFb8/Es9faJgD4A+EJEb161Bb5baJoKVGOsVYTpeFDU+zryTfkydpBrHCX56d2lRbwMZOTpv
zyzW0pSe26xFv78LGkBbmTokfesIBFepBKClpyIluSnZiOE4BrSMLyY1Gc1o7f5AxhVlHqnSGtSp
DbOEeQP6z1PSUQnFFHIz14NMxw+r2aJJmDZrC0Czyb+PoJYXt75V+GuyT4/paNRex74mQ82oNdHa
a9XX4LFPM+LKxzn3BMkwzzE+/DgCzi6UilKmZyo3JIndm3p1jBGfPvaYveC54TmG9RvIA/DhdRs
vnVgJPjhbHBoOcvOD4QUSPEdUak+x+rse25gf0DL6b8gl2Is4qC3SLjgMSwluAd51GTVpb3fCI69
fO41Bw31FQ3qdiXXKesrvIeGf2fTf7c8PnW59Vw5y7CAWgOQFWaxfXw0taMRbLlS+nA3HU+Xc6+u
REB5sF9idr7O1WEwy3RulmnMDvgkn6s8PWXFJPhZVXlosuvzCkOngU/NcW2kYdczrOqKFNTtguGT
F8uRL76FKZ9NglvjPxiQN7Bofc5Mr1CeaZMDSNaiwmwoFcLhc/sJm/36FwwkTEbgT0xnF7Kb0tlc
vdqLQbOvlW2J2Z24e8Npf4r3bAtsm+4BGyQ4HMARanCgLGqVtKPzlUAOW7i9UWp5Nc5gJvwOJ4Lj
F+QBNfau8uALRexr5ItxgJfKHpAuwPrDSoIzRUb2hFXqdgq6AVypl8UyQ/p2FcNaU2Oji0HV9Rwc
X8QlNXslQNtwn7gTHt6CGIK9y/W9WpfZM5XKtmlMYOPDDyRG25MtS7BjCbzB6627G1/n5LxKVhMJ
ao6SmwA17VUSM5Iop94g3DldkPYhORDj2Z1h/NCD8UM3xg8vgfH8zjBe9GC86MZ48RIYP9wZxo89
GD92Y/z4Ehgv7gzjpx6MnjB/vDONlD8bLboyXL4Hx051hvOrBeNWN8eolMF7eFcZhJ8Jh
F77hS6C7ui90xz34jrsRHr8IxuH0zkCe9YA86wZ5diPIbZGPfM4VkXCkzBO6K6trwFqAqoNwwrnS
v7uCEFZztS+We8xK/TRSNbqO0pu8j6+95jJ39ejd8HTBU1lpYNCh7QLmX1PvBCIqzLjch6eVJ2ZIUZfhYT1nfzTYfUlQKkhTOnMVrOp5Onzxw3OMg2hLAyHQwSvPV
skMJJ/oJrlOnydNNIDmj9dUUjv8rpbJksRSy7QKSvXJqtviuuWu0nzhMzb/AfwbvooObcEqwflgy
PyY+Cl7jpaGGuJdMrQQZlrWNerSqUVu6+jbk++wx3EuYM/OT7i9yX/8PUEsHCBkHr6rtBwAA0SsA
AFBLAwQUAAgICAC9Y4pBAAAAAAAAAAAAAAAADAAAAHNldHRpbmdzLnhtbLVaXXPaOhB9v78i4/cE
QhraMAkdIKVNQwIDpJnbN2EvU9ZeKPXe3q7NmjNddfV6E4eQGlOcob
7/ys6p2A9DHgcnbjPY27p1+8r82/rnE65T40AvTjEKQ51WAMPaJP6HWpG+ntGy9WsoFMc92QLATd
MH4DI5Cb1xpvn24kztIrK8Hl4sabGxM1KpXlcnm2vDhDNaucX11dVZK7m0d9lFM+O9RV+vRbV4j4
6si+kC4mcVarVj9V0v+9k2yRb1JT85qbPGzCb15nDtI/p9xAaHNzkl22S7vxyGXjhcPyNWvevf+

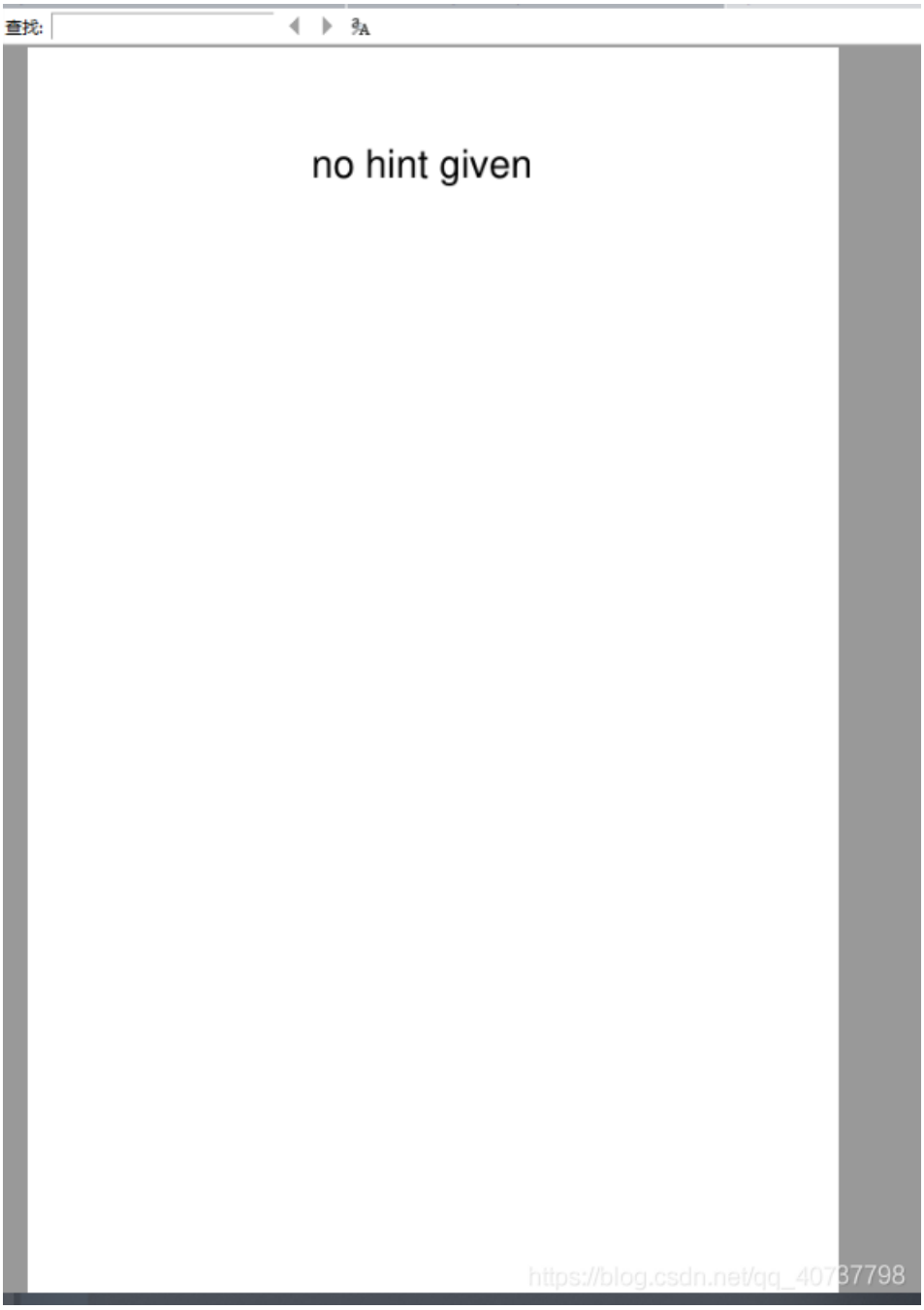
+84ver6lgI0x8jZ3zDqiQwLlzGtWryu7Jg4324OncWH3mOdmytdwrf65dnmc8R/AZ/P9yz6/qtcL



```
AAAAAABtaW1ldHlwZVBLAQIUABQAAAgAAL1jikHpqAV1ZgQAAGYEAAAYAAAAAAAAAAAAAAAAE0A
AABUaHVtYm5haWxzL3RodW1ibmFpbC5wbmdQSwECFAAUAAgICAC9Y4pBP2lTPqYDAAA9DgAACwAA
AAAAAAAAAAAAAADpBAAAY29udGVudC54bWxQSwECFAAUAAgICAC9Y4pBGQevqu0HAADRKwAACgAA
AAAAAAAAAAAAADICAAAc3R5bGVzLnhtbFBLAQIUABQACAgIAL1jikEnrhDrhQUAANsjAAAMAAAA
AAAAAAAAAAAAO0QAABzZXR0aW5ncy54bWxQSwECFAAUAAAIAAC9Y4pBGzLESrYDAAC2AwAACAAA
AAAAAAAAAAAAACsFgAAbWV0YS54bWxQSwECFAAUAAgICAC9Y4pBtPdo0gUBAACDAwAADAAAAAAA
AAAAAAAAACIGgAAbWFuaWZlc3QucmRmUEsBAhQAFAAICAgAvWOKQQAAAAACAAAAAAAAACcAAAAA
AAAAAAAAAAxxsAAENvbmZpZ3VyYXRpb25zMi9hY2NlbGVyYXRvci9jdXJyZW50LnhtbFBLAQIU
ABQAAAgAAL1jikEAAAAAAAAAAAAAAAaAAAAAAAAAAAAAAAAAB4cAABDb25maWd1cmF0aW9uczIv
dG9vbHBhbmVsL1BLAQIUABQAAAgAAL1jikEAAAAAAAAAAAAAAAaAAAAAAAAAAAAAAAAAFYcAABD
b25maWd1cmF0aW9uczIvc3RhdHVzYmFyL1BLAQIUABQAAAgAAL1jikEAAAAAAAAAAAAAAAcAAAA
AAAAAAAAAAAAI4cAABDb25maWd1cmF0aW9uczIvcHJvZ3Jlc3NiYXIvUEsBAhQAFAAACAAAvWOK
QQAAAAAAAAAAAAAABgAAAAAAAAAAAAAAAAAyBwAAENvbmZpZ3VyYXRpb25zMi90b29sYmFyL1BL
AQIUABQAAAgAAL1jikEAAAAAAAAAAAAAAAfAAAAAAAAAAAAAAAAAP4cAABDb25maWd1cmF0aW9u
czIvaW1hZ2VzL0JpdG1hcHMvUEsBAhQAFAAACAAAvWOKQQAAAAAAAAAAAAAAABoAAAAAAAAAAAA
AAAAOx0AAENvbmZpZ3VyYXRpb25zMi9wb3B1cG1lbnUvUEsBAhQAFAAACAAAvWOKQQAAAAAAAAA
AAAAABgAAAAAAAAAAAAAAAAAcx0AAENvbmZpZ3VyYXRpb25zMi9mbG9hdGVyL1BLAQIUABQAAAgA
AL1jikEAAAAAAAAAAAAAAAYAAAAAAAAAAAAAAAAKkdAABDb25maWd1cmF0aW9uczIvbWVudWJh
ci9QSwECFAAUAAgICAC9Y4pBi1ynShoBAAA+BAAAFQAAAAAAAAAAAAAAADfHQAATUVUQS1JTkYv
bWFuaWZlc3QueG1sUEsFBgAAAAARABEAcAQAADwfAAAAAA==
'''
```

pdf部分：

将第一部分独立地新建一个pdf文件，打开pdf看到以下内容：



因此第一部分内容无效。

第二段内容：

尝试rot-13解密，获取到以下解密内容：

> 在线解密网站：http://www.mxcz.net/tools/rot13.aspx

```
cipher="H51\\\'Ux2J&+(3Z;Uxcx0Xxs\x13h\x014$V!R($R>\t/)R!\x01<.\x13,N-aP4M4aRuG1-VuU0 GuH+a@0W=3R9\x01>(_0\x01,8
C0Rx GuN6\"V|\x1ezKZ3\x014$]}R!2\x1d4S?7\x1au\x1fxs\t_\x01xa\x13<Gx)R&Ip2J&\x0f93T#zj\x1c\x1ap\x13rk\x00g\x01e|\
x13g\x19ju\x0ba\x18jt\x02o+xa\x13u\x01xa\x13%S1/Gu\x03\x1b.\\:N7.\\:N4o\x13\x0cN-3\x133M9&\x13<Rx A2WjiZ{DvaX0Xj
h\x136N6\"R!\x01\x07rC0p\x138a\x1dc22ieu\x161Fw+=-@0\x1bRa\x13u\x01(3Z;UxcR\'F.s\x1c>D!s\x13<Rx,Z&R1/Tw+R"
n =0 ;import hashlib ,sys ;
try :key =sys .argv [1 ]
except IndexError :sys .exit ("x\x9c\xf3N\xadT0T\xc8\xcd,.\xce\xccKW\xc8\xccSH,J/\x03\x00M\x97\x07\\".decode ("m
vc"))
f =getattr (hashlib ,"x\x9c\xcbM1\x05\x00\x02G\x01\x07".decode ("mvc"))
while n <(5 *10 **6 ):key =(f (key ).digest ());n =n +1
key =key [:5 ].upper ()
while len (key )<len (cipher ):key =key *2
plain ="".join (map (chr ,[ord (a )^ord (b )for a ,b in zip (cipher ,key )]))
try :exec plain
except :print "x\x9c\x0b/\xca\xcfKW\xf0N\xadT\x04\x00\x14d\x03x".decode ("mvc"), repr(plain)
```

通过其中“decode（“mvc”）”可以发现：其中需要decode的内容加密方式为zlib加密。解密zlib加密内容后并对代码进行整理：

```
#解密脚本:
import zlib
zlib_s = b''#单引号内填写密文
print(zlib.decompress(zlib_s))
```

之后得到的整个内容进行整理可知：

```
import hashlib,sys;
cipher="H51\\\'Ux2J&+(3Z;Uxcx0Xxs\x13h\x014$V!R($R>\t/)R!\x01<.\x13,N-aP4M4aRuG1-VuU0 GuH+a@0W=3R9\x01>(_0\x01,8
C0Rx GuN6\"V|\x1ezKZ3\x014$]}R!2\x1d4S?7\x1au\x1fxs\t_\x01xa\x13<Gx)R&Ip2J&\x0f93T#zj\x1c\x1ap\x13rk\x00g\x01e|\
x13g\x19ju\x0ba\x18jt\x02o+xa\x13u\x01xa\x13%S1/Gu\x03\x1b.\\:N7.\\:N4o\x13\x0cN-3\x133M9&\x13<Rx A2WjiZ{DvaX0Xj
h\x136N6\"R!\x01\x07rC0p\x138a\x1dc22ieu\x161Fw+=-@0\x1bRa\x13u\x01(3Z;UxcR\'F.s\x1c>D!s\x13<Rx,Z&R1/Tw+R"
n=0
try:key=sys.argv[1]
except IndexError :sys .exit ("Key 1 missing in argv")
f =getattr (hashlib ,"md5")
while n<5000000:
    key=f(key).digest()
    n+=1
key=key[:5].upper()
while len(key)<len(cipher):
    key=key*2
plain ="".join (map (chr ,[ord (a )^ord (b )for a ,b in zip (cipher ,key )]))
try:exec plain
except :print "Wrong key!"
```

分析上述代码，我们可以认识到：我们有个cipher与一个key。程序接受argv内的key，key在md5中散列了五十万次；之后，key的前五个字节大写并在解密中充当异或key，并在key的长度等于cipher长度之前，将key值扩大2倍，并将zip(chipher,key)中每一对元素进行异或处理，之后通过map用chr函数转化成字符并添加到plain中，最终执行plain。
我们可以尝试写个暴力破解脚本，但是该脚本也会产生很多错误的key值。

```
import time
from itertools import permutations
from math import ceil

cipher = "H51\\\'Ux2J&+(3Z;Uxcx0Xxs\x13h\x014$V!R($R>\t/)R!\x01<.\x13,N-aP4M4aRuG1-VuU0 GuH+a@0W=3R9\x01>(_0\x01
,8C0Rx GuN6\"V|\x1ezKZ3\x014$]}R!2\x1d4S?7\x1au\x1fxs\t_\x01xa\x13<Gx)R&Ip2J&\x0f93T#zj\x1c\x1ap\x13rk\x00g\x01e
|\x13g\x19ju\x0ba\x18jt\x02o+xa\x13u\x01xa\x13%S1/Gu\x03\x1b.\\:N7.\\:N4o\x13\x0cN-3\x133M9&\x13<Rx A2WjiZ{DvaX0
Xjh\x136N6\"R!\x01\x07rC0p\x138a\x1dc22ieu\x161Fw+=-@0\x1bRa\x13u\x01(3Z;UxcR\'F.s\x1c>D!s\x13<Rx,Z&R1/Tw+R"
num_key_chars = 5
alphabet = "".join(map(chr, range(256)))
keylen = int(ceil(len(cipher) / float(num_key_chars)))
start = time.clock()
for key in permutations(alphabet , num_key_chars):
    expanded = key * keylen
    plain = "".join(map(chr, [ord(a)^ord(b) for a,b in zip(cipher ,expanded)]))
    try:
        exec plain
    except:
        pass
    else:
        print "=== Found key (%s s) ===" % ((time.clock() - start),)
        print key
```

结果：

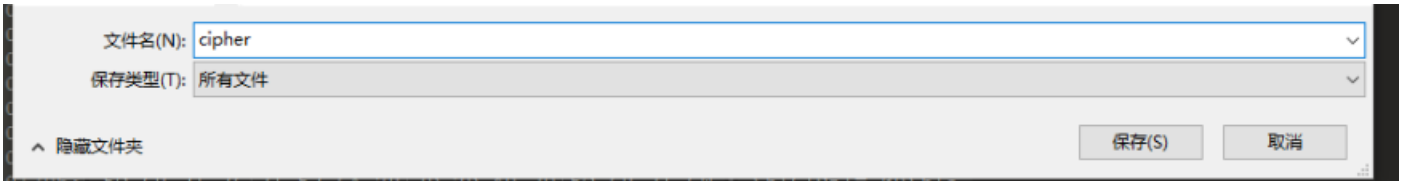
从这些结果中我们很难发现真正的key，因此我们采取使用xortool这个工具来进行暴力破解。

> ## Usage
>
> ! *python3 is not supported, use python 2.x*
> xortool [-h|–help] [OPTIONS] [<filename>]
> Options:
> -l,–key-length length of the key (integer)
> -c,–char most possible char (one char or hex code)
> -m,–max-keylen=32 maximum key length to probe (integer)
> -x,–hex input is hex-encoded str
> -b,–brute-chars brute-force all possible characters
> -o,–brute-printable same as -b but will only use printable
> characters for keys

首先，我们先用010editor或者使用python脚本将cipher内容写入二进制文件cipher。

010editor实现：

将cipher部分复制粘贴，选择文件->新建->16进制文件，右键粘贴，并在文件->另存为处修改名字并保存到指定位置。



python实现：

```
cipher = "H51\\\'Ux2J&+(3Z;Uxcx0Xxs\x13h\x014$V!R($R>\t/)R!\x01<.\x13,N-aP4M4aRuG1-VuU0 GuH+a@0W=3R9\x01>(_0\x01
,8C0Rx GuN6\"V|\x1ezKZ3\x014$]}R!2\x1d4S?7\x1au\x1fxs\t_\x01xa\x13<Gx)R&Ip2J&\x0f93T#zj\x1c\x1ap\x13rk\x00g\x01e
|\x13g\x19ju\x0ba\x18jt\x02o+xa\x13u\x01xa\x13%S1/Gu\x03\x1b.\\:N7.\\:N4o\x13\x0cN-3\x133M9&\x13<Rx A2WjiZ{DvaX0
Xjh\x136N6\"R!\x01\x07rC0p\x138a\x1dc22ieu\x161Fw+=-@0\x1bRa\x13u\x01(3Z;UxcR\'F.s\x1c>D!s\x13<Rx,Z&R1/Tw+R"
file = open('cipher','')
file.write(cipher)
file.close()
```

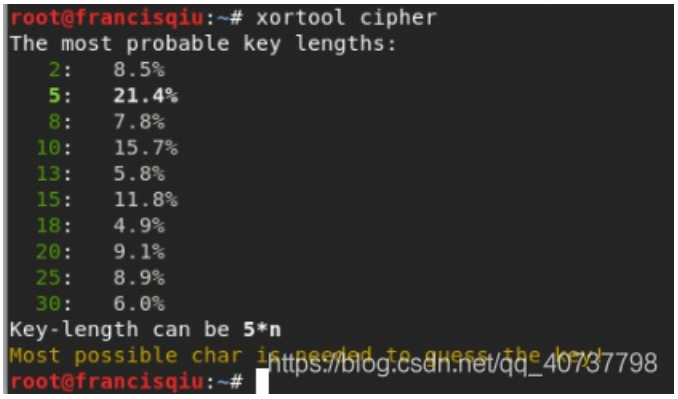
进入虚拟机，先安装xortool，之后执行命令(以kali为例，如果是ubuntu请在命令前输入sudo)：

安装xortool：

> pip install xortool

使用xortool：

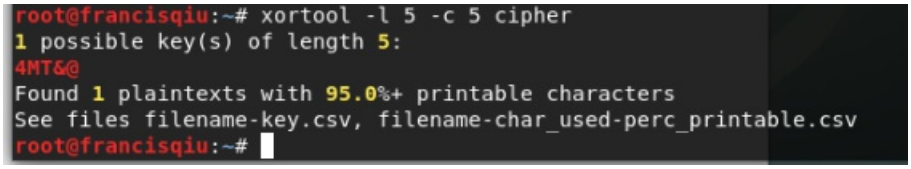
> xortool cipher



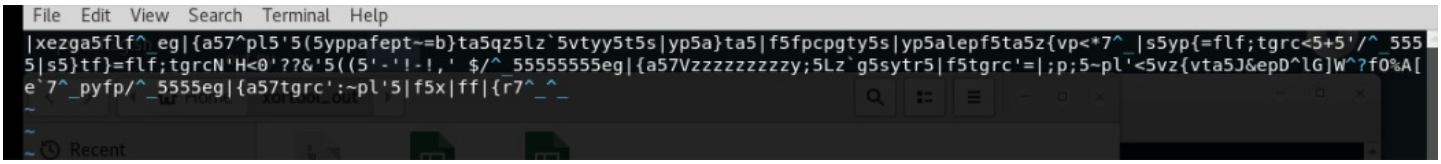
从使用中可以知道：char是5的n倍，依次尝试以下命令：

> xortool -l 5 -c 5 cipher

结果：



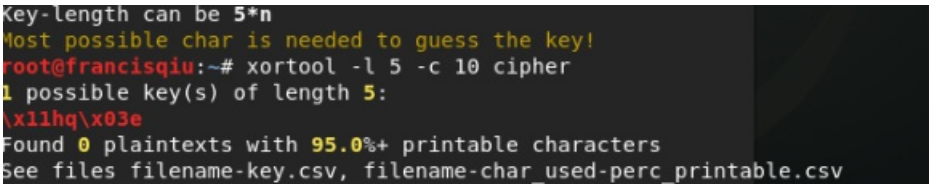
用vim打开输出文档，发现：



因此排除这种情况。

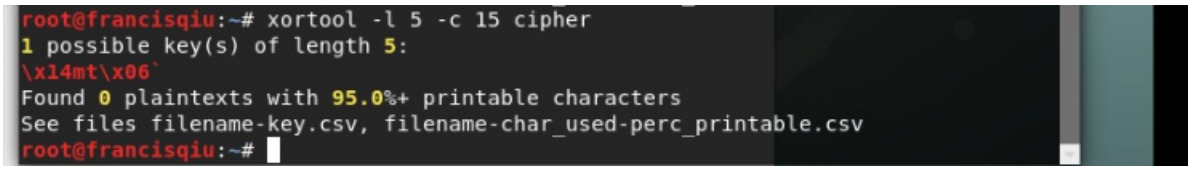
> xortool -l 5 -c 10 cipher

结果：



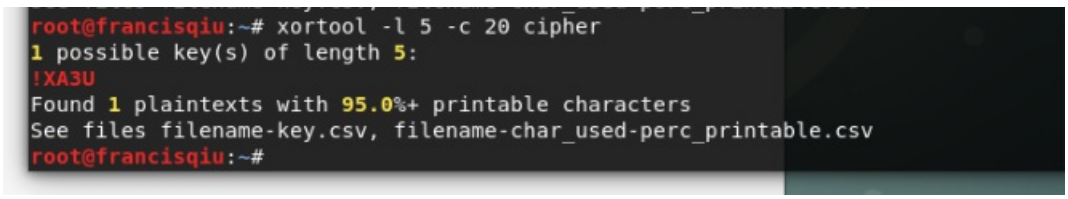
> xortool -l 5 -c 15 cipher

结果：



> xortool -l 5 -c 20 cipher

结果：



打开第四次的out文件，发现以下内容：

```
import sys
print "Key 2 = leetspeak(what do you call a file that is several file types at once)?"
if len(sys.argv) > 2:
    if hash(sys.argv[2])%2**32 == 2824849251:
        print "Coooooooool. Your flag is argv2(i.e. key2) concat _3peQKyRHBjsZ0TNpu"
else:
    print "argv2/key2 is missing"
```

这需要我们解开另一个问题：what do you call a file that is several file types at once?这其实是个脑筋急转弯，并且答案需要使用leetspeak书写。通过与google等搜索引擎,答案是“Chameleon”，用leetspeak书写则为“ch4m3l30n”。验证“hash(‘ch4m3l30n’）%2^32”，确实为2824849251。因此flag就是"ch4m3130n"与"_3peQKyRHBjsZ0TNpu"连接的组合字符串。