

# Proof Of Concept

## 0Byte CTF 2023

NAMA Peserta : TunangannyaChizuru

Minggu, 19 Agustus 2023



# KATEGORI Cryptography

## [Token]

### Executive Summary

Diberikan sebuah file bernama chall.py di mana menenunjukkan sebuah source code dari server. Pada chall ini peserta diminta untuk mengerjakan sebuah minigame berupa RSA yang memiliki kelemahan.

### Technical Report

```
def get_prime(n):
    r = getRandomInteger(n)
    p = nextprime(r)
    q = nextprime(r + getRandomInteger(32))
    return p, q

def get_token(l):
    return ''.join(choice('0123456789abcdef') for i in range(l))

correct = 0
while correct < 100:
    token = get_token(32)
    print(token)
    p, q = get_prime(256)
    n = p * q
    e = 0x10001
    m = bytes_to_long(token.encode())
    c = pow(m, e, n)

    print(f'[*] {n = }')
    print(f'[*] {e = }')
    print(f'[*] {c = }')
```

Bisa dilihat pada source code di atas, terdapat fungsi untuk mengambil p dan q dengan nilai yang berdekatan. Hal ini yang bisa dimanfaatkan untuk mendapatkan nilai token menggunakan teorema fermat di mana  $N = (a + b)(a - b) \rightarrow = a^2 - b^2$ .

Berdasarkan persamaan di atas, bisa disimpulkan  $\text{squareroot}(N) \leq a^2$ . Dan  $a^2 - N = b^2$ . Sehingga nilai p (a + b) dan nilai q (a - b) dapat dicari dengan:

```
def crackClosepq(n):
    t = round(gmpy2.iroot(n, 2)[0])
    while True:
        t += 1
        s = round(gmpy2.iroot(t ** 2 - n, 2)[0])
        if s ** 2 == t ** 2 - n:
            q = t - s
            p = t + s
            return (p - 1) * (q - 1)
```

Full payload:

```
def get_param(r):
    r.recvuntil(b"n = ")
    n = int(r.recvline().strip())
    r.recvuntil(b"e = ")
    e = int(r.recvline().strip())
    r.recvuntil(b"c = ")
    c = int(r.recvline().strip())
    return n, e, c

r = remote('0x7e7ctf.zerobyte.me', 10021, level = 'debug')

def crackClosepq(n):
    t = round(gmpy2.iroot(n, 2)[0])
    while True:
        t += 1
        s = round(gmpy2.iroot(t ** 2 - n, 2)[0])
        if s ** 2 == t ** 2 - n:
            q = t - s
            p = t + s
            return (p - 1) * (q - 1)
```

```

for i in range(100):
    n, e, c = get_param(r)
    # print(n, e, c)
    phi = crackClosepq(n)
    d = inverse(e, phi)
    m = pow(c, d, n)
    token = (long_to_bytes(m)).decode()
    r.sendline(str(token))
    print(i + 1)
    r.recv(1024)

r.recv(1024)

```

Flag : 0byteCTF{emang\_boleh\_sedekat\_ini\_dek?}

## Conclusion

Pemilihan P dan q yang berdekatan memiliki kelemahan sehingga tidak aman untuk digunakan. Gunakan nilai p dan q yang dipilih secara acak agar informasi tidak bocor.

## [Enkripsi Jadoel]

### Executive Summary

Diberikan sebuah file bernama app.py di mana menunjukkan sebuah source code dari server. Pada chall ini peserta diminta untuk menemukan celah keamanan dari enkripsi aes yang telah diberi salt.

### Technical Report

```

def enc(plaintext):
    plaintext = bytes.fromhex(plaintext)
    salt = os.urandom(8)
    padded = pad(plaintext + salt + flag.encode(), 16)
    cipher = AES.new(KEY, AES.MODE_ECB)
    encrypted = cipher.encrypt(padded)
    return encrypted.hex()

```

Pada app.py bisa dilihat bahwa menu enkripsi menambahkan salt dan padding pada enkripsi aes. Yang bermasalah pada enkripsi ini adalah adanya padding pada enkripsi. Hal ini bisa dimanfaatkan

untuk melakukan brute untuk setiap byte flag. Langkah pertamanya adalah mencari panjang flag terlebih dahulu dengan mencoba mengirimkan hex string hingga panjang ciphertext bertambah 16 bytes. Setelah itu dengan memanfaatkan urutan padding flag dapat diextract dari byte paling belakang hingga byte pertama.

Full source code:

```
import string
from pwn import *
from Crypto.Util.number import *
import gmpy2

char_list = [hex(ord(char))[2:] for char in string.printable]
padd_list = ["0x10", "0x01", "0x02", "0x03", "0x04", "0x05", "0x06", "0x07", "0x08", "0x09", "0x0a", "0x0b", "0x0c", "0x0d", "0x0e", "0x0f"]

r = remote('0x7e7ctf.zerobyte.me', 10027)

r.recvuntil(b'Masukkan pilihan: ')
r.sendline(b'1')
r.recvuntil(b'Masukkan pesan: ')
r.sendline(b'00000000000000000000000000000000')
r.recvuntil(b'enkrupsi: ')
last_block = (r.recvline().strip())[-32:]
print(last_block)
pad_count = 16
flag = ""
for i in range(58):
    guess = ("00" * 14 + "00" * (i + 1))
    r.recvuntil(b'Masukkan pilihan: ')
    r.sendline(b'1')
    r.recvuntil(b'Masukkan pesan: ')
    r.sendline(guess.encode("utf-8"))
    r.recvuntil(b'enkrupsi: ')
    ciphertext1 = r.recvline().strip()
    for char in char_list:
        if len(char) == 1:
            char = "0" + char
        if ciphertext1[-32:] == last_block:
            remove_pad = (len(guess) - 28) // 32
            comparer = ciphertext1[-32 * (remove_pad + 1):-32 * remove_pad]
            pad_count = 16
        elif len(flag) // 2 >= 16:
            remove_pad = (len(guess) - 28) // 32
            comparer = ciphertext1[-32 * (remove_pad + 1):-32 * remove_pad]
        else:
            comparer = ciphertext1[-32:]
```

```

payload = (char + flag + padd_list[16 - ((i % 16) + 1)][2:] * pad_count)
r.recvuntil(b'Masukkan pilihan: ')
r.sendline(b'1')
r.recvuntil(b'Masukkan pesan: ')
r.sendline(payload.encode("utf-8"))
r.recvuntil(b'enkripsi: ')
ciphertext2 = r.recvline().strip()
first_16bytes = ciphertext2[:32]
print("Trying : " + payload)
if first_16bytes == comparer:
    flag = char + flag
    print("bytes : " + str(i + 1))
    print("Found : " + char + " ! flag now is : " + flag)
    print("ascii : " + bytes.fromhex(flag).decode('utf-8'))
    break

```

Flag : 0byteCTF{y4\_m4u\_b4g4im4n4\_l4g1\_3nkr1ps1\_j4d03l\_347fd48d64}

## Conclusion

Penggunaan padding pada enkripsi AES mode ecb dalam kasus ini bisa menimbulkan kelemahan. Penggunaan salt di situ tidak begitu memiliki dampak dikarenakan posisi salt yang ada sebelum byte flag.

## KATEGORI Forensic

### [Who the Hack]

#### Executive Summary

Diberikan sebuah log di mana berisikan http request pada sebuah server. Pada deskripsi soal dikatakan ada anomali pada <2023-08-12 19:46:26> dan server sempat down untuk beberapa menit.

#### Technical Report

Dengan statik analisis kita bisa tahu bahwa server yang down akan memberikan response 404. Dan setelah melihat log pada [11/Aug/2023:19:00:05 +0700], ada sebuah request get dengan endpoint /%30 yang dimana penulis berpikir kalau itu adalah byte depan dari flag. Sehingga penulis mengekstrak dengan memanfaatkan response server dan user agent.

Full source code:

```

from urllib.parse import unquote
with open(r'D:\My Documents\Downloads\access\access.log', "r") as file1:
    log_entries1 = file1.read()

log_lines1 = log_entries1.strip().split('\n')
ip_addresses_403Duliet = []
for line in log_lines1:
    if " 404 " in line and "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu Chromium/98.0.4674.123 Chrome/98.0.4674.123 Safari/537.36" in line:
        ip_addresses_403Duliet.append(line.split(' ')[6])

flag = ""
for i in ip_addresses_403Duliet:
    flag += unquote(i[1:])
print(flag)

```

## Conclusion :

Memahami server response sangat krusial pada forensic karena hal tersebut bisa menjadi parameter untuk seorang analis mengekstrak data dari sebuah log file.

Full flag:

0byteCTF{W3\_Suff3r\_M0r3\_0ft3n\_1n\_1m4g1n4t10n\_Th4n\_1n\_R34l1ty}

**[Pink]**

## Executive Summary

Diberikan sebuah pcap file untuk dianalisis lebih lanjut.

## Technical Report

Menggunakan fitur protocol hierarki yang ada pada wireshark sangat mempermudah analisis. Sehingga dengan memanfaatkan deskripsi soal, seperti protocol yang perlu dianalisis lebih lanjut adalah ICMP dengan ping request sebagai bintang utama.

data						
No.	Time	Source	Destination	Protocol	Length	Info
100	14.404124	52.80.178.206	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 101)
101	14.404155	206.189.38.179	52.80.178.206	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 100)
102	14.596676	71.136.117.85	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 103)
103	14.596706	206.189.38.179	71.136.117.85	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 102)
104	14.911227	52.81.22.35	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=231 (reply in 105)
105	14.911265	206.189.38.179	52.81.22.35	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 104)
204	32.667540	140.179.224.106	206.189.38.179	ICMP	104	Echo (ping) request id=0x000e, seq=12908/27698, ttl=26 (reply in 205)
205	32.667549	206.189.38.179	140.179.224.106	ICMP	104	Echo (ping) reply id=0x000e, seq=12908/27698, ttl=64 (request in 204)
432	74.958335	52.80.210.120	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=231 (reply in 433)
433	74.958364	206.189.38.179	52.80.210.120	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 432)
434	75.066601	52.81.192.55	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 435)
435	75.066636	206.189.38.179	52.81.192.55	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 434)
436	75.126571	52.81.179.116	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 437)
437	75.126600	206.189.38.179	52.81.179.116	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 436)
885	134.443467	52.80.134.128	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 886)
886	134.443497	206.189.38.179	52.80.134.128	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 885)
887	134.474991	52.81.22.35	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=231 (reply in 888)
888	134.475020	206.189.38.179	52.81.22.35	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 887)
889	134.665092	71.136.71.158	206.189.38.179	ICMP	62	Echo (ping) request id=0x0005, seq=20389/42319, ttl=232 (reply in 890)
890	134.665120	206.189.38.179	71.136.71.158	ICMP	56	Echo (ping) reply id=0x0005, seq=20389/42319, ttl=64 (request in 889)
932	145.401801	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0xc01a, seq=1/256, ttl=62 (reply in 933)
933	145.401836	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0xc01a, seq=1/256, ttl=64 (request in 932)
934	145.410015	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x0e2f, seq=1/256, ttl=62 (reply in 935)
935	145.410046	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x0e2f, seq=1/256, ttl=64 (request in 934)
936	145.414903	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x6e76, seq=1/256, ttl=62 (reply in 937)
937	145.414930	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x6e76, seq=1/256, ttl=64 (request in 936)
938	145.418887	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x91f3, seq=1/256, ttl=62 (reply in 939)
939	145.418921	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x91f3, seq=1/256, ttl=64 (request in 938)
940	145.422609	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x9cbc, seq=1/256, ttl=62 (reply in 941)
941	145.422632	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x9cbc, seq=1/256, ttl=64 (request in 940)
942	145.426235	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x6aa0, seq=1/256, ttl=62 (reply in 943)
943	145.426266	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x6aa0, seq=1/256, ttl=64 (request in 942)
944	145.429434	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x7bb9, seq=1/256, ttl=62 (reply in 945)
945	145.429460	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x7bb9, seq=1/256, ttl=64 (request in 944)
946	145.433304	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x23bc, seq=1/256, ttl=62 (reply in 947)
947	145.433332	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x23bc, seq=1/256, ttl=64 (request in 946)
948	145.436919	167.172.66.19	206.189.38.179	ICMP	104	Echo (ping) request id=0x9148, seq=1/256, ttl=62 (reply in 949)
949	145.436944	206.189.38.179	167.172.66.19	ICMP	104	Echo (ping) reply id=0x9148, seq=1/256, ttl=64 (request in 948)

```

> Frame 936: 104 bytes on wire (832 bits), 104 bytes captured (832 bits)
> Linux cooked capture v2
> Internet Protocol Version 4, Src: 167.172.66.19, Dst: 206.189.38.179
> Internet Control Message Protocol

```

```

0000 06 00 00 00 00 00 02 00 01 00 06 fe 00 00 00 .....
0010 01 01 04 02 45 00 00 54 c9 db 40 00 3e 01 93 9d ...E..T...@...
0020 87 ac 42 13 ce bd 26 b3 00 00 f5 c2 6e 76 00 01 ...B...&...nv...
0030 32 9a d5 64 00 00 00 00 6a cb 0c 00 00 00 00 00 2...d....j.....
0040 67 65 6c 61 70 0a 48 75 6a 61 6e 20 74 61 6b 20 gelap Hu jan tak
0050 67 65 6c 61 70 0a 48 75 6a 61 6e 20 74 61 6b 20 gelap Hu jan tak
0060 67 65 6c 61 70 0a 48 75

```

Jika dilihat dari paket di atas, terdapat seperti lirik lagu yang menarik perhatian penulis.

```

0000 08 00 00 00 00 00 00 02 00 01 04 06 52 87 cf 3a .....R...
0010 f0 8b 00 00 45 00 00 54 d1 f8 00 00 40 01 c9 80 ...E..T...@...
0020 ce bd 26 b3 a7 ac 42 13 00 00 8a 62 59 6b 00 01 ...&...B...bYk..
0030 32 9a d5 64 00 00 00 00 07 ab 0d 00 00 00 00 00 2...d....
0040 6e 0a 44 61 6e 20 4d 47 4a 35 64 47 56 44 56 45 n-Dan MG J5dGVDVE
0050 6e 0a 44 61 6e 20 4d 47 4a 35 64 47 56 44 56 45 n-Dan MG J5dGVDVE
0060 6e 0a 44 61 6e 20 4d 47 n-Dan MG

```

Jika dilihat lebih lanjut, paket paket berikutnya ada yang memiliki potongan huruf seperti encoding base64. Dan jika semua potongan tersebut digabung maka akan didapat flag.



## Decode from Base64 format


Simply enter your data then push the decode button.

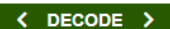

```
MGJ5dGVDVEZ7aWNtcF9wNGNrZXRfN2ZzZGFkOTE2MX0=
```

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

 Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE**  Decodes your data into the area below.

```
0byteCTF{icmp_p4cket_7fsdad9161}
```

## Conclusion :

Langkah awal menganalisa wireshark dapat dilakukan dengan hierarki protokol agar bisa melihat jenis jenis protokol yang ada pada pcap file.

Full flag:

```
0byteCTF{icmp_p4cket_7fsdad9161}
```