

p4-team / ctf

Watch78

Star296

Fork63

<> Code

🔔 Issues 0

🔗 Pull requests 0

📁 Projects 0

📶 Pulse

📊 Graphs

Branch: master ▾ ctf / 2015-09-16-csaw / forensics_100_transfer /



Create new file

Find file

History

 Pharisaesus added eng versions

Latest commit aac03de on Oct 7 2015

..		
 README.md	added eng versions	2 years ago
 transfer.pcap	add csaw: Transfer (forensics 100) writeup	2 years ago

 README.md

Transfer (forensics, 100p, 541 solves)

PL

ENG

I was sniffing some web traffic for a while, I think i finally got something interesting. Help me find flag through all these packets.

[net_756d631588cb0a400cc16d1848a5f0fb.pcap](#)

Pobrany plik pcap ładujemy do Wiresharka żeby po chwili przeglądania transmisji HTTP (menu File -> Export Objects -> HTTP) znaleźć następujący kod źródłowy programu:

```
import string
import random
from base64 import b64encode, b64decode

FLAG = 'flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}'

enc_ciphers = ['rot13', 'b64e', 'caesar']
# dec_ciphers = ['rot13', 'b64d', 'caesard']

def rot13(s):
    _rot13 = string.maketrans(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZnopqrstuvwxyz",
        "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    return string.translate(s, _rot13)

def b64e(s):
    return b64encode(s)

def caesar(plaintext, shift=3):
    alphabet = string.ascii_lowercase
    shifted_alphabet = alphabet[shift:] + alphabet[:shift]
    table = string.maketrans(alphabet, shifted_alphabet)
    return plaintext.translate(table)

def encode(pt, cnt=50):
    tmp = '2{}'.format(b64encode(pt))
    for cnt in xrange(cnt):
        c = random.choice(enc_ciphers)
        i = enc_ciphers.index(c) + 1
        _tmp = globals()[c](tmp)
        tmp = '{}{}'.format(i, _tmp)

    return tmp

if __name__ == '__main__':
    print encode(FLAG, cnt=?)
```

W tej samej transmisji (opcja Follow TCP Stream) była również zakodowana wiadomość.

Po odwróceniu wszystkich algorytmów otrzymujemy taki program dekodujący:

```
import string
import random
from base64 import b64encode, b64decode

FLAG = 'flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}'

#enc_ciphers = ['rot13', 'b64e', 'caesar']
dec_ciphers = ['rot13', 'b64d', 'caesard']

def rot13(s):
    _rot13 = string.maketrans(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZnopqrstuvwxyz",
        "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    return string.translate(s, _rot13)

def b64d(s):
    return b64decode(s)

def caesard(plaintext, shift=-3):
    alphabet = string.ascii_lowercase
    shifted_alphabet = alphabet[shift:] + alphabet[:shift]
    table = string.maketrans(alphabet, shifted_alphabet)
    return plaintext.translate(table)

def encode(pt, cnt=50):
    tmp = '2{}'.format(b64encode(pt))
    for cnt in xrange(cnt):
        c = random.choice(enc_ciphers)
        i = enc_ciphers.index(c) + 1
        _tmp = globals()[c](tmp)
        tmp = '{}{}'.format(i, _tmp)

    return tmp

def decode(pt, cnt=61):
    for i in xrange(cnt):
        c = pt[0]
        if c == '1':
            pt = rot13(pt[1:])
        if c == '2':
            pt = b64d(pt[1:])
        if c == '3':
            pt = caesard(pt[1:])

    print pt

if __name__ == '__main__':
    x = '2Mk16Sk5iakYxVFZoS1RswNZbFZaYjFaa1prWmFkMDVWVGs1U2IyODFXa1ZuTUZadU1YVldiVkpVVFVaS1dGwX1kbUZXTVdkMVprWnJWM1Z'
    decode(x)
```

Odkodowana wiadomość i flaga to: `flag{li0ns_and_tig3rs_4nd_b34rs_0h_mi}`.

ENG version

I was sniffing some web traffic for a while, I think i finally got something interesting. Help me find flag through all these packets.

[net_756d631588cb0a400cc16d1848a5f0fb.pcap](#)

We load the downloaded file to Wireshark and after looking for a while on HTTP transmissions (menu File -> Export Objects -> HTTP) we find a source code:

```
import string
import random
from base64 import b64encode, b64decode

FLAG = 'flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}'

enc_ciphers = ['rot13', 'b64e', 'caesar']
# dec_ciphers = ['rot13', 'b64d', 'caesard']
```

```

def rot13(s):
    _rot13 = string.maketrans(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZnopqrstuvwxyz",
        "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    return string.translate(s, _rot13)

def b64e(s):
    return b64encode(s)

def caesar(plaintext, shift=3):
    alphabet = string.ascii_lowercase
    shifted_alphabet = alphabet[shift:] + alphabet[:shift]
    table = string.maketrans(alphabet, shifted_alphabet)
    return plaintext.translate(table)

def encode(pt, cnt=50):
    tmp = '2{}'.format(b64encode(pt))
    for cnt in xrange(cnt):
        c = random.choice(enc_ciphers)
        i = enc_ciphers.index(c) + 1
        _tmp = globals()[c](tmp)
        tmp = '{}{}'.format(i, _tmp)

    return tmp

if __name__ == '__main__':
    print encode(FLAGS, cnt=?)

```

In the same transmission (Follow TCP Stream option) there was also an encoded message.

After reversing all the algorithms we get a decoding software:

```

import string
import random
from base64 import b64encode, b64decode

FLAG = 'flag{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}'

#enc_ciphers = ['rot13', 'b64e', 'caesar']
dec_ciphers = ['rot13', 'b64d', 'caesard']

def rot13(s):
    _rot13 = string.maketrans(
        "ABCDEFGHIJKLMNOPQRSTUVWXYZnopqrstuvwxyz",
        "NOPQRSTUVWXYZnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ")
    return string.translate(s, _rot13)

def b64d(s):
    return b64decode(s)

def caesard(plaintext, shift=-3):
    alphabet = string.ascii_lowercase
    shifted_alphabet = alphabet[shift:] + alphabet[:shift]
    table = string.maketrans(alphabet, shifted_alphabet)
    return plaintext.translate(table)

def encode(pt, cnt=50):
    tmp = '2{}'.format(b64encode(pt))
    for cnt in xrange(cnt):
        c = random.choice(enc_ciphers)
        i = enc_ciphers.index(c) + 1
        _tmp = globals()[c](tmp)
        tmp = '{}{}'.format(i, _tmp)

    return tmp

def decode(pt, cnt=61):
    for i in xrange(cnt):
        c = pt[0]
        if c == '1':
            pt = rot13(pt[1:])
        if c == '2':
            pt = b64d(pt[1:])
        if c == '3':
            pt = caesard(pt[1:])

```

```
print pt

if __name__ == '__main__':
    x = '2Mk16Sk5iakYxVFZoS1RsWnZXbFZaYjFaa1prWmFkMDVWVGs1U2IyODFXa1ZuTUZadU1YVldiVkpVhVFaS1dGWX1kbUZXTVdkMVprWnJWM1Z
    decode(x)
```

Decoded message and the flag is: `flag{li0ns_and_tig3rs_4nd_b34rs_0h_mi}`.

