HITCON Knowledge Base

HITCON CTF Quals 2015 - Simple (Crypto 100)

作者: QUAN YANG
來源: http://nusgreyhats.org/write-ups/HITCONCTF-Quals-2015-Simple-(Crypto-100)/

HITCON CTF Quals 2015 was from 17 October 2015, 10 am to 18 October 2015, 10pm. CTFTIME Page. Most of the challenges were very tedious, and this is one of the challenges that we solved (Although we only managed to solve this after the CTF ended).

Simple

> Points: 100
> Category: Cryptography
> Description Become admin!
> http://52.69.244.164:51913
> simple-01018f60e497b8180d6c92237e2b3a67.rb
> md5: 4bd00c892d5e71f6d1d25d0bff2f49ec

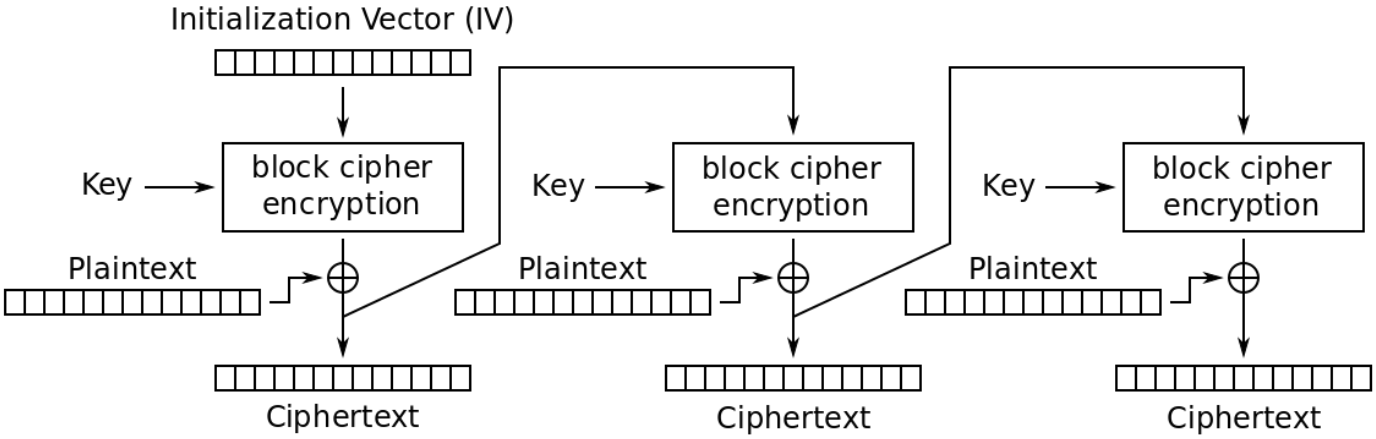
Our solution

Given the source code of the website, we're told to get admin. Looking at the source code provided, to be able to print the flag out, we have to get the conditon r['admin'] to be equal to true.

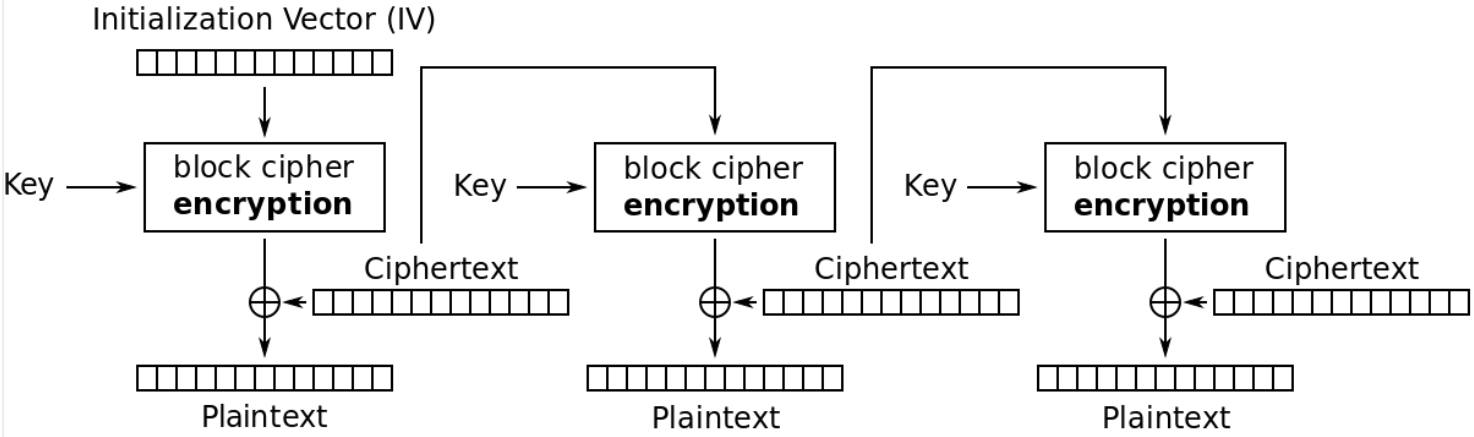
```
#!/usr/bin/env ruby

require 'sinatra/base'
require 'sinatra/cookies'
require 'openssl'
require 'json'

KEY = IO.binread('super-secret-key')
FLAG = IO.read('/home/simple/flag').strip

class SimpleApp < Sinatra::Base
  helpers Sinatra::Cookies

  get '/' do
    auth = cookies[:auth]
    if auth
      begin
        auth = auth.b
        c = OpenSSL::Cipher.new('AES-128-CFB')
        c.decrypt
        c.key = KEY
        c.iv = auth[0...16]
        json = c.update(auth[16..-1]) + c.final
        r = JSON.parse(json)
        if r['admin'] == true
          "You're admin! The flag is #{FLAG}"
        else
          "Hi #{r['username']}, try to get admin?"
        end
      rescue StandardError
        'Something wrong QQ'
      end
    else
      <<-EOS
<html><body><form action='/' method='POST'>
<input type='text' name='username'/>
<input type='password' name='password'/>
<button type='submit'>register!</button>
```

```
39 </form></body></html>
40       EOS
41     end
42   end
43
44   post '/' do
45     username = params['username']
46     password = params['password']
47     if username && password
48       data = {
49         username: username,
50         password: password,
51         db: 'hitcon-ctf'
52       }
53       c = OpenSSL::Cipher.new('AES-128-CFB')
54       c.encrypt
55       c.key = KEY
56       iv = c.random_iv
57       json = JSON.dump(data)
58       enc = c.update(json) + c.final
59       cookies[:auth] = iv + enc
60       redirect to('/')
61     else
62       'Invalid input!'
63     end
64   end
65 end
```

It seems that the IV used as well as the encrypted json is kept in the client's cookie, and that the same cookie is used to determine if you're an admin. (This indicates that if we can spoof the encrypted json, we can become admin)



Cipher Feedback (CFB) mode encryption



Cipher Feedback (CFB) mode decryption

AES-128 in CFB mode has a block size of 16 bytes.
Simply put,
Ciphertext of block #1 = E(IV, key) ^ Plaintext

Therefore, with knowledge of plaintext and ciphertext, we are able to obtain E(IV, key) and to forge for the first block of cipher text.

With a username and password of b, the Plaintext of the first block will be
`{"username":"b",`
and we'll use that knowledge to obtain our E(IV, key)

This is our exploit script that forges our first block to be: `{"admin": true }`
and allows us to obtain our flag!

```
1 import requests
2 import urllib
3
4 def main():
5
6     original_cookie =
"\xE9a\x89\xEC\xC7\x7C\xBC\x15\x92\xAD\xF8\x17\xF8\x40" \
7
"wV\xAB524\xF2\xF5UA\xE8\x1A\x29\xD4\xCB\xFA\xF6\xB3" \
8
"\x95h\x2B\x0D\xF4\xB9\xC8\xDB\xF8n\xB9o\xBES\x11d\xA3" \
9
"9\xA3c\x3Fi\xE7\xFA\x1C\xD0\xDBk\xDD\xD2_6\x06"
10
11     original_cookie = original_cookie.encode('hex')
12     iv = original_cookie[0:32]
13     first_16_byte_block = original_cookie[32:64]
14     print "IV: %s" % iv
15     print "First Block: %s" % first_16_byte_block
16
17     #Plain text of first 16 byte block.
18     plaintext ='{"username":"b",'
19
20     encrypted_iv = int(plaintext.encode('hex'),16) ^
int(first_16_byte_block,16)
21     encrypted_iv = hex(encrypted_iv)[2:-1]
22     print "Encrypted IV: %s" % encrypted_iv
23
24     #The text I want to forge in the first block.
25     forge_text = '{"admin": true }'
26
27     print 'Encrypting payload...'
28
29     payload = int(forge_text.encode('hex'),16) ^
int(encrypted_iv,16)
30     payload = hex(payload)[2:-1]
31     payload = iv + payload
32     print "PAYLOAD: %s" % payload
```

```
33
34      cookie = {"auth": payload.decode("hex")}
35      r = requests.get("http://52.69.244.164:51913/",
cookies=cookie)
36      print "Flag: %s" % r.text
37
38 if __name__ == "__main__":
39      main()
```

Running the script gives us:

```
.../CTF/hitcon python simple.py
IV: e96189ecc77cbc1592adf817f8407756
First Block: ab353234f2f55541e81a29d4cbfaf6b3
Encrypted IV: d017474797873b20857f0beee998d49f
Encrypting payload...
PAYLOAD: e96189ecc77cbc1592adf817f8407756ab352623faee5502bf5f7f9c9cfdf4e2
Flag: You're admin! The flag is hitcon{WoW_CFB_m0dE_5o_eAsY}
```

And we have our flag: **hitcon{WoW_CFB_m0dE_5o_eAsY}**

#ctf #hitconctf #crypto #aes

Jan 25th, 2016