INFOSEC

HACKING

CBC byte flipping attack—101 approach

August 22, 2013 by Daniel Regalado

Share:

As usual, there are some explanations about this attack out there (see references at the end), but some knowledge is required to understand it properly, so here I will describe, step by step, how to perform this attack.

Purpose of the attack

To change a byte in the plaintext by corrupting a byte in the ciphertext.

Why?

To bypass filters by adding malicious chars like a single quote, or to elevate privileges by changing the ID of the user to admin, or any other consequence of changing the plaintext expected by an application.

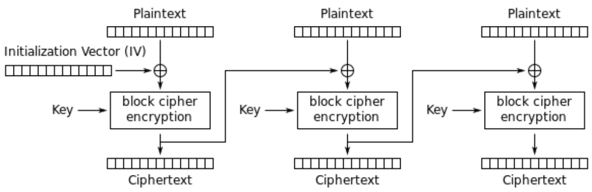
In this Series

First of all, let's start understanding how CBC (cipher-block chaining) works. A detailed explanation can be found here:

http://en.wikipedia.org/wiki/Block_cipher_mode_of_operation#Cipher-block_chaining_.28CBC.29

But I will only explain what is needed to understand the attack.

Encryption process



Cipher Block Chaining (CBC) mode encryption

Plaintext

The data to be encrypted.

IV: A block of bits that is used to randomize the encryption and hence to produce distinct ciphertexts even if the same plaintext is encrypted multiple times.

Key: Used by symmetric encryption algorithms like AES, Blowfish, DES, Triple DES, etc.

Ciphertext: The data encrypted.

An important point here is that CBC works on a fixed-length group of bits called a block. In this blog, we will use blocks of 16 bytes each.

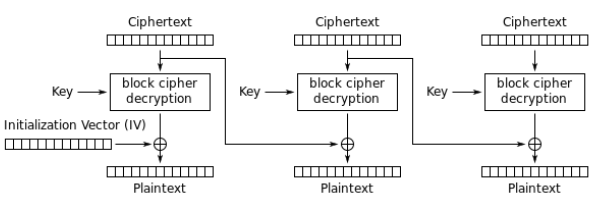
Since I hate mathematical formulas, below are mine:

Ciphertext-0 = Encrypt(Plaintext XOR IV)—Just for the first block.

Ciphertext-N= Encrypt(Plaintext XOR Ciphertext-N-1)—For second and remaining blocks.

Note: As you can see, the ciphertext of the previous block is used to generate the next one.

Decryption process



Cipher Block Chaining (CBC) mode decryption

Plaintext-0 = Decrypt(Ciphertext) XOR IV—Just for the first block.

Plaintext-N= Decrypt(Ciphertext) XOR Ciphertext-N-1—For second and remaining blocks.

CBC byte flipping attack—101 approach

How to crack a password: Demo and video walkthrough

Inside Equifax's massive breach: Demo of the exploit

Wi-Fi password hack: WPA and WPA2 examples and video walkthrough

How to hack mobile communications via Unisoc baseband vulnerability

How to build a hook syscall detector

Top tools for password-spraying attacks in active directory networks

NPK: Free tool to crack password hashes with AWS

Tutorial: How to exfiltrate or execute files in compromised machines with DNS

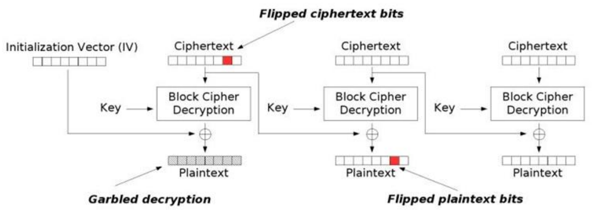
Top 19 tools for hardware hacking with Kali Linux

20 popular wireless hacking tools [updated 2021]

13 popular wireless hacking tools [updated 2021]

Man-in-the-middle attack: Real-life example and video

Note: The Ciphertext-N-1 is used to generate the plaintext of the next block; this is where the byte flipping attack comes into play. If we change one byte of the Ciphertext-N-1 then, by XORing with the net decrypted block, we will get a different plaintext! You got it? Do not worry, we will see a detailed example below. Meanwhile, below is a nice diagram explaining this attack:



Modification attack on CBC

Example: CBC blocks of 16 bytes

Let's say we have this serialized plaintext:

a:2:{s:4:"name";s:6:"sdsdsd";s:8:"greeting";s:20:"echo 'Hello sdsdsd!'";}

Our target is to change the number 6 at "s:6″ to number "7". The first thing we need to do is to split the plaintext into 16-byte chunks:

Block 1: a:2:{s:4:"name";
Block 2 s:6:"sdsdsd";s:8 <<<—–target data-blogger-escaped-div="" data-blogger-escaped-here="">
Block 3: :"greeting";s:20:
Block 4: "echo 'Hello sd
Block 5: sdsd!'";}

So our target character is located at block 2, which means that we need to change the ciphertext of block 1 to change the plaintext of the second block.

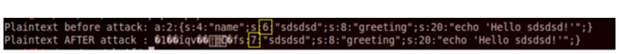
A rule of thumb is that the byte you change in a ciphertext will ONLY affect a byte at the same offset of next plaintext. Our target is at offset 2:

[0] = s
[1] = :
[2] = 6

Therefore we need to change the byte at offset 2 of the first ciphertext block. As you can see in the code below, at line 2 we get the ciphertext of the whole data, then at line 3 we change the byte of block 1 at offset 2, and finally we call the decryption function.

1. $v = "a:2:{s:4:"name";s:6:"sdsdsd";s:8:"greeting";s:20:"echo 'Hello sdsdsd!'";}";
2. $enc = @encrypt($v);
3. $enc[2] = chr(ord($enc[2]) ^ ord("6") ^ ord ("7"));
4. $b = @decrypt($enc);

After running this code, we are able to change number 6 to 7:



But, how did we change the byte to the value we wanted at line 3?

Based on the decryption process described above, we know that A = Decrypt(Ciphertext) is XOR with B = Ciphertext-N-1 to finally get C = 6. Which is equal to:

C = A XOR B

So the only value we do not know is A (block cipher decryption); with XOR we can easily get that value by doing:

A = B XOR C

And finally, A XOR B XOR C is equal to 0. With this formula, we can set our own value by adding it at the end of the XOR calculation, like this:

A XOR B XOR C XOR "7" will give us 7 in the plaintext at offset 2 on the second block.

Below is the PHP source code so that you can replicate it:

```
[php]
define('MY_AES_KEY', "abcdef0123456789″);
function aes($data, $encrypt) {
$aes = mcrypt_module_open(MCRYPT_RIJNDAEL_128, '',
MCRYPT_MODE_CBC, '');
$iv = "1234567891234567";
mcrypt_generic_init($aes, MY_AES_KEY, $iv);
return $encrypt ? mcrypt_generic($aes,$data) :
mdecrypt_generic($aes,$data);
}

define('MY_MAC_LEN', 40);

function encrypt($data) {
return aes($data, true);
}

function decrypt($data) {
$data = rtrim(aes($data, false), "\0");
return $data;
}
$v = "a:2:{s:4:"name";s:6:"sdsdsd";s:8:"greeting";s:20:"echo
'Hello sdsdsd!'";}";
echo "Plaintext before attack: $vn";
$b = array();
$enc = array();
$enc = @encrypt($v);
$enc[2] = chr(ord($enc[2]) ^ ord("6") ^ ord ("7"));
$b = @decrypt($enc);
echo "Plaintext AFTER attack : $bn";
[/php]
```

Try changing the character from "7" to "A" or something else to see how it works.

Exercise 2