AESWT PoC 2

Matteo Bordignon,
28/03/2024

Background

The system is vulnerable to **bit-flipping attacks** [1]. It is an attack on a cryptographic cipher in which the attacker can change the ciphertext in such a way as to result in a predictable change of the plaintext, although the attacker is not able to learn the plaintext itself. In the case of AES-CBC [2][3], the attacker can modify specific bits in the ciphertext to flip the corresponding bits in the decrypted plaintext.
In AES CBC, **each block of plaintext is XORed with the previous block of ciphertext** before being encrypted. This means that **if an attacker modifies a bit in the ciphertext of one block**, it will result in the corresponding **bit in the decrypted plaintext of the next block being flipped**. This is because the modified ciphertext block will be XORed with the next block of plaintext during decryption, resulting in the bit flip.
The attack is possible since **there is no integrity** and authentication on the data. A MAC or an HMAC can be used to prevent this such as AES-CBC-HMAC if the CBC mode is a must to use [4].
In other cases, it is better to use modern encryption schemes. The authenticated encryption with Associated Data (AEAD)[5] which provides confidentiality, integrity, and authenticity.

Vulnerability

The vulnerability is present in the application since the AES-CBC is used to encrypt the information provided by the user to create a token that then is used for user authentication without any implemetation of MAC or HMAC.

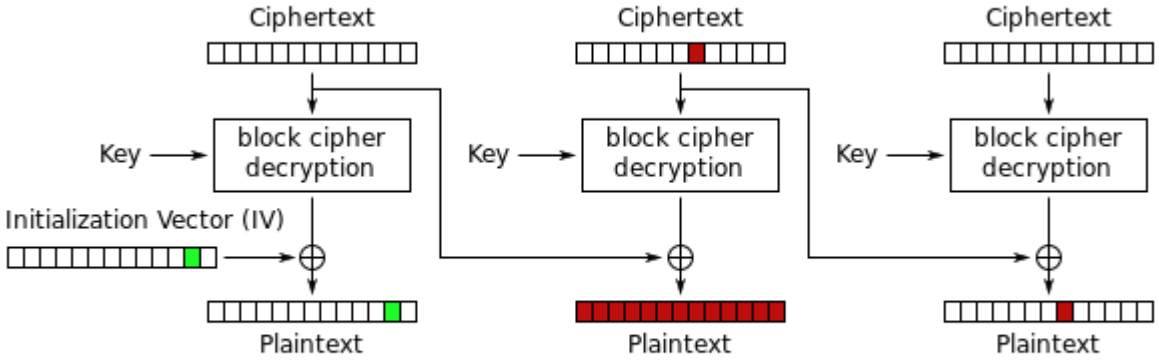
Solution

The AES-CBC decryption process is perfomed as

$P_1 = Dec_k(C_i) \oplus IV$

$P_i = Dec_k(C_i) \oplus C_{i-1}$

This means that if we know how the plaintext should be to authenticate us as **admin** we can modify the cipher text of the previous block in order to obtain our desidered plaintext. Note that when a ciphertext block is modified, the corresponding plaintext block will be completly corrupted since AES-CBC is very sensitive to changes.



Cipher Block Chaining (CBC) mode decryption

First step: sign up

Let's create a token:

```
# token generation
r.recvuntil(b"Login with a token")
r.sendlineafter(b"> ", bytes("1".encode("utf-8")))
r.recvuntil(b"Choose a username")
r.sendlineafter(b"> ", bytes("bdmin?desc?I am a boss".encode("utf-8")))
r.recvuntil(b"Insert a description of yourself")
r.sendlineafter(b"> ", bytes("xxxxxxxxxx".encode("utf-8")))
r.recvuntil(b': ')
ciphertext =  bytes.fromhex(r.recvline(keepends=False).decode("utf-8"))
```

The application will encrypt this text:

"desc=xxxxxxxxxx&user=bdmin?desc?I am a boss\0x05\0x05\0x05\0x05\0x05"

And return the encrypted token such as:

```
6c3fbe7aaa93862c5a592b6acd59a7168c76e9664ae1baea5864f99332013c70558d6e
204abf447885ae32b5c5337602add00946091f4baa362e2bff7bfdcab6
```

Second step: alteration of the ciphertext

We obtained a ciphertext composed of three blocks plus the IV each of 16 bytes:

- IV: xxxxxxxxxxxxxxxx
- Block 1: desc=xxxxxxxxx&
- Block 2: user=bdmin?desc?
- Block 3: I am a boss\0x05\0x05\0x05\0x05\0x05

We have to modify the ciphertext of the first block to change the plaintext of the target block (the second one). To do so we use this snippet of the python script:

```
# this is the original string in the first block
plaintext =  b"user=bdmin?desc?"

# this is the string that we want to obtain
goal =  b"user=admin&desc="

# modification of the ciphertext
block1 = ciphertext[16:32]
iv = pwn.xor(block1, plaintext, goal)

# since we changed the ciphertext of the first block we can replace the iv
# and the old block1 ciphertext with the new block1 ciphertext
new_token = iv + ciphertext[32:]
```

Third step: sending the new token

Now we send the forged token for authentication and print the flag, using this snippet:

```
r.recvuntil(b"Login with a token")
r.sendlineafter(b"> ", bytes("2".encode("utf-8")))
r.recvuntil(b"Insert the token (hex)")
r.sendlineafter(b"> ", new_token.hex().encode())
flag = r.recvline(keepends=False).decode("utf-8")
print(flag)
```

References

1. Bit-flipping attack wikipedia
2. AES-CBC RFC-3602
3. Bit-flipping The CTF recipes
4. Stack exchange Bit Flipping Attack on CBC Mode
5. AEAD wikipedia
6. Full solution python code