Crypto 400 – Write Up

El reto final de la categoría de criptografía consiste en un CBC Bit flipping attack en una implementación de AES. El titulo del reto fl1pp3r da un indicio de lo que se trata-

El texto del reto dice lo siguiente:

Los delfines son animales muy inteligentes, o eso se dice, asi que le pedimos a uno ayudandarnos a implementar un sistema de tickets para facilitar la autenticación. Estamos empezando a dudar de su conocimiento en implementaciones criptográficas, pues varios usuarios se han quejado de alertas de inicio de sesión no reconocidas en sus cuentas.

Por alguna razón a todo nuevo usuario se le entrega, una llave "AAAAAAAAAAAAAAA" a la que se le dice quizás incorrectamente "llave de inicialización", se cuenta con capacidad para 6 IDs de usuarios invitados (0-5), siendo el usuario con id 6 el administrador del sistema.

Nos preocupa que nuestro administrador pueda ser comprometido. Puedes ayudarnos a descubrir el error en la implementación criptográfica?

Además de que se nos entrega el código fuente del reto.

Analizando el reto vemos la siguiente ejecución:

```
f@ubuntu:~/Documents/hackdef/400$ nc 127.0.0.1 3102

                DEVELOPED BY FL1PP3R THE DOLPHIN
                                  _
                              _.-~  )
                   _..--~~~~,'   ,-/     _
                .-'. . . .'   ,-','    ,' )
              ,'. . . _   ,--~,-'__..-'  ,'
            ,'. . .  (@)' ---~~~~      ,'
           /. . . . '~~             ,-'
          /. . . . .             ,-'
         ; . . . .  - .        ,'
        : . . . .       _     /
       . . . . .          `-.:
      . . . ./  - .          )
     .  . . |  _____..---.._/ ____
~---~~~~~~~~~~~~~~~~           ~~

Bienvenido humano, tu token de autenticacion es: +JzEeGgG1cH/d/59JAWzTw== para el flipid=1
***Si tienes un token previo usalo para autenticarte a continuacion.
1) Ingresar
2) Salir
Eleccion:
```

Por si solo, el reto parece un sistema de autenticación valido. Sin embargo haciendo referencia al texto de descripción del programa vemos que nuestro objetivo es hacer un hijacking del token de sesión de usuario 6.

```
Ingresa tu token: +JzEeGgG1cH/d/59JAWzTw==
Ingresa la llave de inicializacion: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

El token pertenece al usuario: flipid=1
```

SI se inserta una “llave de inicialización ” incorrecta el padding se verá modificado.

```
***Si tienes un token previo usalo para autenticarte a continuacion.
1) Ingresar
2) Salir
Eleccion: 1
Ingresa tu token: +JzEeGgG1cH/d/59JAWzTw==
Ingresa la llave de inicializacion: BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

El token pertenece al usuario: Padding Incorrecto
```

Analizando el código fuente del reto podemos notar lo siguiente:

La generación de la llave se realiza de manera aleatoria con la función:

```
def gen_random_key(N):
    return ''.join(secrets.choice(string.ascii_uppercase + string.digits) for _ in range(N))
```

La cual se invoca con un tamaño de 16. Comprendiendo esto, podemos saber que la “llave de inicialización” no es la llave de cifrado. Las funciones que realizan el cifrado, no parecen tener ningún problema de implementación, entonces volvamos al control que se le da al usuario.

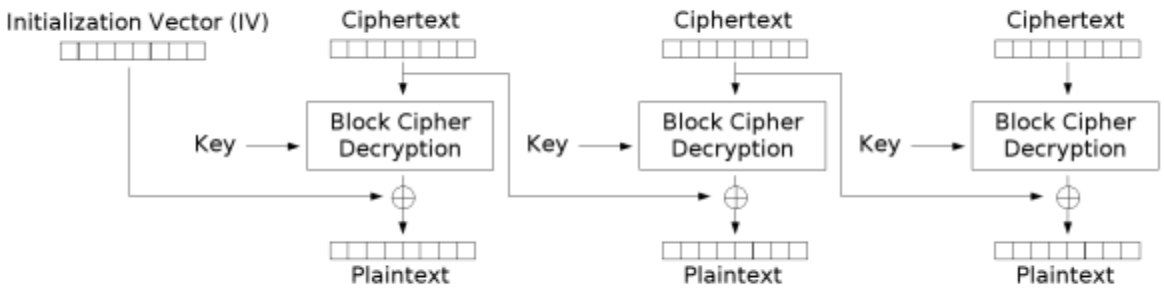
```
def delfin_encrypt(iv,key,msg):
    cipher = AES.new(bytes(key,"utf-8"),AES.MODE_CBC,bytes(iv,"utf-8"))
    los_bytes = cipher.encrypt(pad(bytes(msg,"utf-8"),AES.block_size,style="pkcs7"))
    return base64.b64encode(los_bytes)

def delfin_decrypt(iv,key,msg):
    try:
        ciphertext = base64.b64decode(msg)
        cipher = AES.new(bytes(key,"utf-8"),AES.MODE_CBC,iv)
        el_string = unpad(cipher.decrypt(ciphertext),AES.block_size,style="pkcs7")
        return str(el_string,"ascii")
    except:
        return "Padding Incorrecto"
```

Específicamente en la sección que valida el token de usuario:

```
self.request.send(b"Ingresa tu token: ")
token = self.request.recv(1024).strip()
self.request.send(b"Ingresa la llave de inicializacion: ")
llave = self.request.recv(1024).strip()
if len(llave) < 16:
    self.request.send(b"\r\nLa llave de inicializacion no cumple con el tamano requerido\r\n")
id = delfin_decrypt(llave[:16],key,token)
```

Se considera como llave al vector de inicialización, por lo que tenemos control sobre el bloque de código descifrado con una simple función XOR. Analizando como es aplicado el descifrado en AES-CBC, podemos ver que el Vector de Inicialización aplica solamente la función XOR en el texto tras pasar por la función AES, permitiéndonos manipular los bits del primer bloque de cifrado.



Cipher Block Chaining (CBC) mode decryption

El proceso puede realizarse manualmente. Ya que 'A' XOR 'F' = 6. Dado que solo queremos cambiar un byte de la cadena, tenemos que modificar el byte que hace referencia al ID, es decir, la octava posición.

```
El token pertenece al usuario: flipid=1
Quieres seguir intentando? (S/N): S
Ingresa tu token: +JzEeGgG1cH/d/59JAWzTw==
Ingresa la llave de inicializacion: AAAAAAABAAAAAAAA

El token pertenece al usuario: flipid=2
Quieres seguir intentando? (S/N): S
Ingresa tu token: +JzEeGgG1cH/d/59JAWzTw==
Ingresa la llave de inicializacion: AAAAAAACAAAAAAAA

El token pertenece al usuario: flipid=3
Quieres seguir intentando? (S/N): S
Ingresa tu token: +JzEeGgG1cH/d/59JAWzTw==
Ingresa la llave de inicializacion: AAAAAAAFAAAAAAAA

Felicidades: hackdef{_nunC4_dej3s_3l_c0ntR0l_d3l_IV_a_uN_usuAr1o_qU3_3nt1end3_3L_m0d0_d3_op3r4ci0n_CBC_}
```