Lab 1 Seguridad Computacional: Padding Oracle Attack

Intergrantes: Edgar Morales, Jean Paul Duchens

a) Estudio de ambos servicios

Tenemos que el servicio 1 cifra y el 2 descifra. Probando diferentes largos de inputs para el servicio 1: -"H" (texto plano de largo 1) -> clave de largo 112 bytes -"Ho" (texto plano de largo 2) -> clave de largo 112 bytes . . . -texto plano de largo 9 -> clave de largo 128 bytes

Concluyendo que los bloques son de largo de 16 bytes por la manera en la que opera PKCS7 rellenando un nuevo bloque con padding.

Además pudimos observar que tenemos un largo límite del texto plano que se puede ingresar y por ende un largo límite del texto plano que nos da al descifrar, pues dando como texto plano la secuencia "aaaaa....." (256 bytes) logramos que el descifrador se desfazara y, que para nuevas peticiones se entregaran respuestas pasadas de bloques que no se lograron procesar a tiempo.

b) Realizar el programa que envia un mensaje al servidor de encriptacion y responde al servidor de desencriptacion

Código adjunto en mainb.py, logramos encriptar y desencriptar en una sola funcion.

c) Respuesta teorica sobre la estimacion del tamaño de bloques para un cifrador

Según lo analizado en la parte a), podemos inferir el tamaño del bloque del cifrador enviando mensajes de longitud creciente, agregando un byte a la vez, y observando los largos del texto cifrado. Cuando se detecte un incremento repentino en la longitud del texto cifrado, ese salto indica el tamaño del bloque utilizado.

Main_god.py

Decidimos explicar el punto f) solamente, este consta de los puntos d) y e), pues es el padding oracle attack completo

d)

Código adjunto. Funciona perfectamente.

e)

Código adjunto. Funciona perfectamente.

f)

Comenzamos dividiendo el texto cifrado en bloques de tamaño 16, luego iteramos desde el penúltimo bloque ahsta el primero, ya que el primer bloque es el IV y no se descifra como parte del mensaje.

Para cada bloque debemos almacenar el bloque descifrado y los valores intermedios (resultado de aplicar XOR entre el padding y el byte anterior al descifrado). Se seleccionan el bloque actual y previo (o IV si es el primero), y creamos una copia del bloque previo, con esta copia queremos lograr detectar los bytes del texto plano.

Dentro de cada bloque recorremos cada byte desde el final hasta el inicio y para cada uno se prueban los 256 valores posibles, modificando ese byte en la copia del bloque previo. El bloque modificado y el actual se unen y se envian al oraculo, si el padding es correcto calculamos el valor del padding esperado y, con eso, se deduce el byte intermedio. Luego, podemos calcular el byte real del texto plano mediante el XOR entre el byte intermedio y el byte original del bloque previo.

Después de descubrir un byte, se actualizan los bytes siguientes del bloque modificado para simular un padding válido más largo, lo cual permite descubrir el próximo byte en la siguiente iteración. Repetimos este proceso hasta descifrar todo el bloque. Una vez terminado un bloque, se antepone a la cadena de bloques descifrados y se continua con el que sigue.

Finalmente la clave encontrada fue la siguiente:

83c804bf74f04b841b9c186342c62a1797cd87b1fe6ce7bf54d32c4a16390020