Laboratorio 1

Ataque Padding Oracle

Profesores: Alejandro Hevia y Eduardo Riveros
Auxiliares: Sergio Rojas
Ayudantes: Darlene Sobarzo y Tomás Alvarado

- Trabajo personal o en parejas
- Entrega: viernes 4 de abril a las 11:45 hrs.

En este laboratorio programaremos el ataque *Padding Oracle*, el cual consiste en aprovecharse del *padding* agregado a los textos planos para obtener el mensaje a partir de su texto cifrado, sin necesidad de conocer la llave usada para cifrar.

Puede usar el código disponible en U-Cursos (sección material docente) como base para construir su solución.

Deben entregar el código realizado en cada etapa, junto a un breve informe recopilando la información.

1. Resumen de Ataque Padding Oracle

1.1. Padding

El modo AES-CBC requiere que el texto plano a encriptar sea múltiplo de 16 bytes. Si no lo es, se necesita agregar un padding, el cual sigue el estándar PKCS7.

En el padding PKCS7, se llenan todos los bytes restantes del último bloque con un mismo byte X, el cual se escoge para representar el número de bytes que «sobran» para completar un bloque completo. Por ejemplo:

$$pkcs7(\text{hola mundo}) = \text{hola mundo} + [0\text{x}06] * 6$$

$$pkcs7(\text{mensaje completo}) = \text{mensaje completo} + [0\text{x}10] * 16$$

Donde $[0\text{x}10] = 16$ porque está en hexadecimal. Notar que si el mensaje es del tamaño un bloque, se añade un nuevo bloque para incluir el padding; en otras palabras, siempre se realiza el padding, aún si el texto ya es múltiplo de 16.

1.2. El ataque

Al ejecutar AES-CBC sobre un texto plano $p = [p_1, p_2, ...p_n]$ se obtiene un texto cifrado $c = [c_1, c_2, ..., c_n]$ junto a un IV mediante:

$$c_i = encrypt(p_i \oplus c_{i-1})$$

Con $c_0 = IV$. Despejando obtenemos

$$p_i = decrypt(c_i) \oplus c_{i-1}$$

Notar que si conocemos c, solo nos falta obtener $decrypt(c_i)$ para poder conseguir el texto plano. Gracias al padding, si tenemos un oráculo que nos avisa si el texto cifrado que le entregamos proviene de un texto plano que cumple el padding, podemos obtener información suficiente para conseguir p_i.

El algoritmo empieza obteniendo el último byte del último bloque. Para obtener el último byte de c_i ($c_i[15]$), procedemos de la siguiente forma:

Hacemos un ciclo para ir modificando $c_{i-1}[15]$ de [0x00] a [0xff], y en cada iteración mandamos al oráculo el nuevo c'; si el oráculo nos señala que el padding está correcto (lo cual indica que c' desencripta a un p' con padding correcto), sabemos exactamente que:

$$p_i[15]' = \text{0x01} = decrypt(c_i)[15] \oplus c_{i-1}[15]'$$

Entonces

$$decrypt(c_i)[15] = \text{0x01} \oplus c_{i-1}[15]'$$

Con lo que podemos obtener

$$p_i[15] = decrypt(c_i)[15] \oplus c_{i-1}[15] = \text{0x01} \oplus c_{i-1}[15]' \oplus c_{i-1}[15]$$

Notar que cuando el oráculo nos indica que la desencriptación fue correcta podemos deducir que $p_i[15]' = \text{0x01}$, ya que estamos modificando el último byte.

Ahora, para obtener $p_i[14]$, tenemos que hacer algo parecido, pero ya como estaremos modificando el segundo byte, debemos imponer que el padding sea con [0x02]. Para eso primero seteamos:

$$c_{i-1}[15]' = decrypt(c_i)[15] \oplus \text{0x02}$$

Y procedemos a modificar $c_{i-1}[14]$ de [0x00] a [0xff], y cuando el padding sea el correcto, es porque

$$p_i[14]' = \text{0x02}$$

Se procede de la misma forma para obtener

$$p_i[14]$$

. Para obtener $p_i[13]$, primero se setea

$$c_{i-1}[15]' = decrypt(c_i)[15] \oplus \text{0x03}$$

$$c_{i-1}[14]' = decrypt(c_i)[14] \oplus \text{0x03}$$

Y se sigue la iteración. De esta forma, podemos conseguir todo el bloque de texto plano.

Cuando se consiga todo p_i, se realiza el mismo procedimiento para p_{i-1}, recordar que el padding se realiza siempre en el último bloque, por lo que para trabajar con p_{i-1} se requiere quitar el último bloque de c (el cual ya fue completamente utilizado para obtener el último bloque de p).

2. Servicios a atacar

Se dejaron dos servicios accesibles corriendo en **cc5327.hackerlab.cl**. El servicio A (puerto 5312) y el servicio B (puerto 5313). Con el código base proporcionado pueden conectarse fácilmente a estos servicios.

- Servicio A: Al recibir un mensaje de ustedes m, les entregará un texto en hexadecimal correspondiente a un texto cifrado c en AES-CBC, el cual su texto plano corresponde al mensaje que enviaron junto a una contraseña. Es decir, $c = Encrypt(m + key)$, entendiendo $+$ como la concatenación de los textos.

- Servicio B: Al recibir un texto en hexadecimal correspondiente a un mensaje cifrado c, el servidor les entregará la desencriptación del mensaje incluido en c, o un error en caso de desencriptación incorrecta (cuando el padding no es el correcto). Esto es, en un funcionamiento correcto, si $M = decrypt(c)$, $M = m + key$, se retornará m.

El objetivo es entonces, poder obtener el valor de key mediante un ataque Padding Oracle.

3. Repartición de puntajes

a) **[1.0 pts]** Pruebe los servicios con distintos tipos de entrada (distintos largos, modificaciones de bytes, etc). Documente el análisis exploratorio realizado y sus conclusiones.
b) **[1.0 pts]** Cree un programa basado en el código base que envía un mensaje m al servidor A y le envía la respuesta del servidor A al servidor B. Documente lo observado.
c) **[1.0 pts]** Responda brevemente, ¿cómo podría conocer en un contexto genérico, el tamaño del bloque del cifrador utilizado, sin conocer el cifrador? Solo responda teóricamente, ya que el tamaño del bloque en AES es conocido.
d) **[1.0 pts]** Cree una función que permita descifrar el último carácter del texto cifrado.
e) **[1.0 pts]** Modifique la función anterior para descifrar un bloque completo.
f) **[1.0 pts]** Ejecute satisfactoriamente el ataque descrito para obtener key, registre su valor en el informe del laboratorio.