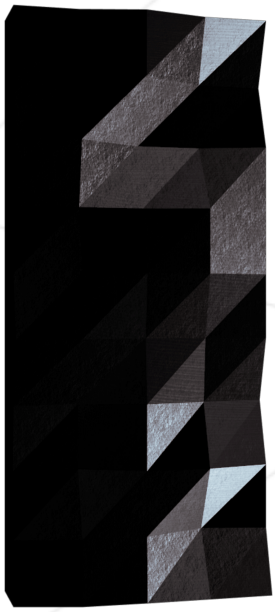

# Bootcamp Ciberseguridad | 42 Madrid

## coRSAir

*Resumen: Bloque de criptografía: cifrados vulnerables*

*Versión: 1*

# Índice general

# Capítulo I

# Introducción

Este proyecto introduce conceptos específicos sobre la fortaleza del algoritmo RSA y sus vulnerabilidades potenciales. Si bien el algoritmo es considerado suficientemente fuerte para la potencia computacional de los dispositivos actuales, ciertas formas de utilizarlo pueden llevar a graves problemas de seguridad.

# Capítulo II

# Prólogo

Había estado cuidando de los pájaros del Capitán Burt, y hablando con él de ellos. Así, nos habíamos hecho bastante amigos. Para mí todavía era Señor, y me llevaba la mano a la frente y todo eso. Pero le había demostrado que se podía relajar conmigo y que aun así podía levantarme de un salto cuando me daba una orden. Por todo esto contestaba a mis preguntas, cuando no eran demasiadas, y me enseñó a usar el astrolabio. Esencialmente, lo que estaba haciendo era medir el ángulo del sol a mediodía. Una vez que sabes eso y la fecha, sabes la latitud. Cuando más al norte estás, más al sur sale el sol y más bajo está al mediodía en invierno. Si sabes la fecha, la tabla te da la latitud. Algunas estrellas pueden ser usadas del mismo modo.
Existen muchos problemas con este sistema, como pude ver. En primer lugar, es difícil conseguir una buena medición a menos que dé la casualidad de que estás sobre una roca. Cuando el mar está en calma, tomas tres mediciones y calculas la media. Cuando está picado, ya puedes olvidarte.
Y eso no es todo. Cuando hace mal tiempo y no se puede ver el sol, no se puede medir. Y además, tu brújula está apuntando al norte magnético, no al real. También hay tablas para la desviación de la brújula, pero tienes que saber tu posición para usarlas. Así que lo que solía hacer (estoy yendo demasiado deprisa otra vez) era comprobar la orientación magnética con la estrella polar. Si esto empieza a sonar complicado, ya verá más adelante. Solo le he dado los puntos relevantes.
Una vez has averiguado tu latitud, todavía necesitas saber tu longitud, y lo único que podemos hacer nosotros es medir nuestra velocidad con la barquilla y registrarla en el cuaderno cada hora. La barquilla tiene un cabo con nudos para medir la velocidad. La arrojas detrás del barco, miras el pequeño reloj de arena y cuentas los nudos.

Por supuesto, todo cambia cuando la tierra está cerca. Te orientas con los objetos de la carta de navegación, que te da tu posición siempre y cuando la carta sea correcta y no hayas elegido mal la isla, la montaña, o lo que sea. Para cuando hube aprendido la mitad de todo esto, ya estábamos muy, muy lejos de Veracruz. Así que... ¡buenas noches!

Extracto de “Confesiones de un Pirata”, Gene Wolfe, 2007.

# Capítulo III

# Instrucciones generales

Para este proyecto, debes usar C como lenguaje. La lista de funciones pemitidas es la siguiente:

- Todas las funciones de `<math.h>`
- Todas las funciones de `<string.h>`
- La librería *openssl*
- Todo lo puesto en cabecera del ejercicio.

# Capítulo IV

# Parte Obligatoria

La seguridad en la criptografía asimétrica usando claves RSA se basa en la premisa de que es muy difícil computacionalmente factorizar los dos factores primos de un número. *"Multiplicar dos números primos p y q para obtener n"* es una operación sencilla y su complejidad no aumenta drásticamente cuando los números crecen:

$$[1736640013 \cdot 1230300287 = 2136588706409583731]$$

En cambio, la operación inversa, *"dado un número n obtener sus dos factores primos"*, es una operación que se vuelve computacionalmente inviable cuando los números involucrados son lo suficientemente grandes.

Para generar la pareja de claves, el algoritmo RSA crea una clave pública y privada usando este concepto. Simplificando la generación de las claves, los números primos elegidos aleatoriamente $p$ y $q$ se multiplican para crear el módulo n que se usará tanto en la clave privada como en la pública. Este módulo $n$ es público pero los factores primos $p$ y $q$ no.

$$[? \cdot ? = 2136588706409583731]$$

Si tenemos dos certificados generados en un sistema cuyo generador de números aleatorios estaba configurado al mínimo y por tanto en el que la entropía era mínima... Esto puede haber dado pie a repeticiones de números primos durante distintas generaciones, se podría haber dado el caso en el que dos módulos compartan el mismo número p o q.

$$[n1 = p1 \cdot q1]\ [n2 = p1 \cdot q2]$$

# Capítulo V

# Ejercicio 1 - coRSAir

| | |
|---|---|
| **Nombre de función** | `corsair` |
| **Archivos a entregar** | `*.c, *.h` |
| **Funciones autorizadas** | `printf, snprintf, write, read, open, close, malloc, free` |
| **Descripción** | `Bloque de criptografía: cifrados vulnerables` |

Con esta información, crearás una herramienta que:

- Lea la clave pública de estos certificados y obtenga el módulo y exponente. Calcular el resto de los datos necesarios.
- Construya la clave privada a partir de dos primos y su producto, y de ahí saque la clave simétrica cifrada con él.
- ¡Descifre el mensaje!

# Capítulo VI

# Parte Bonus

La evaluación de los bonus se hará `SI Y SOLO SI` la parte obligatoria es `PERFECTA`. De lo contrario, los bonus serán totalmente `IGNORADOS`.

Puedes mejorar tu proyecto con las siguientes características:

- Documentación detallada y clara de todos los fundamentos teóricos detrás del proyecto
- Implementación propia de una librería o conjunto de funciones en C para operar con enteros de gran tamaño.
- Todo lo que se te ocurra... podrás justificarlo todo durante la evaluación.

# Capítulo VII

# Evaluación por pares

Este proyecto será corregido por tus compañeros. Entrega los archivos en el repositorio Git y asegúrate de que todo funciona como se espera.