Encriptación para Discord

Miguel Angel Lama Carrasco

***Abstract*—Este trabajo de investigación presenta un algoritmo para la aplicación Discord, con la intención de encriptar los mensajes enviados en esta plataforma.**

I. Introduction

En los últimos años han surgido una variedad de *exploits* en Discord que han llevado a muchos mensajes a ser expuestos al publico [1]. Debido a esto, hoy en día Discord no se puede considerar un espacio privado o seguro. Por lo que, es necesario desarrollar métodos para ocultar la información para así mantener la privacidad dentro de la plataforma. Con este fin este trabajo de investigación presenta un algoritmo que encripta los mensajes enviados atravez de la plataforma utilizando el método de encriptación asimétrica RSA.

II. Marco teórico

El método RSA genera una llave privada d y una llave publica e y n. Luego, encripta un mensaje m utilizando:

$$c = m^e mod(n)$$

donde c corresponde al mensaje cifrado que se desea enviar. Para desencriptar el mensaje utilizamos la llave privada d que tiene la propiedad de:

$$m = c^d mod(n).$$

Donde n es el producto de dos primos grandes, aleatorios y bien espaciados p y q. Para generar e primero calculamos un $\theta = (p-1)(q-1)$ y luego encontramos un e aleatorio que cumpla las siguientes propiedades:

$$1 < e < \theta$$

$$gcd(\theta, e) = 1$$

Por ultimo, para calcular d utilizamos $de = 1 mod \theta$.

El método AES-CBC utiliza la operación XOR para encriptar un mensaje en bloques. Requiere de un IV inicial que se utiliza para generar variabilidad en la salida. Si los datos que se utiliza no caben dentro del bloque final se le añade un padding.

El método SHA-256 se utiliza para firmar un documento y verificar la autenticidad del usuario. Genera una firma en base a una llave privada que se puede verificar con una llave publica.

M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

Revisado por José Carlos Pazos para el curso de ciberseguridad en la Universidad de Ingenieria y Tecnologia.

Enviado Octubre, 2024; Revisado Diciembre, 2024.

III. Diseño y Implementación

Primero, para autenticar que un usuario sea realmente quien dice ser, se utilizarán GitHub y GitLab como servicios externos para almacenar las llaves públicas. Para garantizar la identidad del usuario, realizaremos una verificación presencial antes de autorizar la publicación de su llave pública en la plataforma. De este modo, se previenen casos de suplantación de identidad.

A continuación, desarrollaremos un bot que cada usuario podrá utilizar para encriptar, enviar, recibir y desencriptar mensajes. Este bot funcionará en una aplicación externa a Discord instalada en la computadora del usuario y tendrá acceso únicamente a la llave privada de la persona que esté utilizando la instancia del bot. Esto facilitará el uso del sistema RSA de manera segura y sencilla.

El programa generará una llave privada RSA para cada usuario con una longitud de dos kilobits. Esta llave se almacenará de forma local en la computadora del usuario, protegida dentro de un archivo cifrado mediante el método AES-CBC (Cipher Block Chaining). Para garantizar que solo el usuario pueda acceder a su llave privada, la clave simétrica utilizada para el cifrado AES será generada aleatoriamente y protegida con una contraseña que el usuario definirá. De esta forma, se asegura que las llaves privadas permanezcan protegidas incluso si alguien accede a la máquina local.

Para preservar la integridad de los mensajes enviados y recibidos, se empleará el algoritmo SHA-256 (Secure Hash Algorithm). Antes de encriptar un mensaje con RSA, se generará un hash utilizando SHA-256. Este hash acompañará al mensaje encriptado y será verificado por el receptor tras desencriptar el mensaje en su computadora, asegurando que el contenido no haya sido alterado durante la transmisión.

Finalmente, todos los mensajes enviados a través de un canal de Discord estarán encriptados con RSA. La encriptación y desencriptación de estos mensajes se realizará localmente en la computadora de cada usuario, garantizando así la privacidad y seguridad del contenido.

El programa se desarrollará en JavaScript utilizando Node.js, Express, y las bibliotecas de criptografía node-rsa y crypto. Además, será publicado como código abierto en GitHub.

IV. Amenazas

Debido a que el programa está diseñado para ser ejecutado en el computador del usuario, las amenazas comunes en otras páginas web son de menor preocupación en este proyecto. Sin embargo, existen ciertos riesgos si la computadora del usuario es comprometida.

La principal amenaza radica en el posible acceso a la llave privada y la contraseña del usuario. Con estos dos elementos,

sería posible desencriptar los mensajes dirigidos al usuario y firmar mensajes, haciéndose pasar por él. Para mitigar este riesgo, se añadió una advertencia en GitHub recomendando no compartir la llave privada ni la contraseña. No obstante, un atacante que se haga pasar por un experto podría engañar a un usuario menos experimentado y robar su llave privada.

Amenaza	Probabilidad	Daño
Exploits de memoria	Baja	Alto
Robo de computador	Baja	Alto
Romper cifrado	Baja	Alto
Error de usuario	Alta	Alto
Ataques de inyección	Baja	Alto
Alteración de código	Baja	Alto

V. Resultados

Se realizaron pruebas de funcionalidad, determinando que es posible: encriptar la comunicación, firmar los mensajes, detectar cambios en las llaves públicas y utilizar Discord como medio de mensajería seguro.

VI. Conclusión

Para futuros trabajos, es necesario investigar posibles exploits en JavaScript, las bibliotecas utilizadas y el trabajo propuesto.

Appendix A
Repositorio del trabajo

https://github.com/Skinde/EncriptacionDiscord

References

[1] PrivacyHawk, "Discord's massive data breach: Over 4 billion messages leaked and sold in april 2024." [Online]. Available: https://privacyhawk.com/resources/discords-massive-data-breach-over-4-billion-messages-leaked-and-sold-in/