# Análisis de documentos maliciosos – Parte 03 – Documentos de Microsoft Office

Español

Después de verificar cómo configurar máquinas virtuales como entornos seguros y de presentar un flujo de trabajo introductorio para analizar documentos PDF sospechosos, estamos listos para continuar con los formatos de archivo de Microsoft Office.

## Problemas de seguridad con los documentos de Microsoft Office

En general, los documentos de Office no son peligrosos por sí mismos si sólo contienen la información para la que han sido creados: páginas con texto y otros elementos imprimibles para MS Word, celdas con valores y fórmulas para MS Excel, diapositivas con elementos observables en MS PowerPoint, etc.

Sin embargo, entre las muchas funciones disponibles en el ecosistema de MS Office para añadir funcionalidades adicionales a los documentos, hay una que resulta particularmente interesante desde una perspectiva de seguridad digital: la posibilidad de incrustar objetos en los documentos. El tipo de objetos que podemos incrustar es muy variado, como notaciones matemáticas, elementos multimedia, otros documentos, etc. Y entre todos ellos, hay uno especialmente poderoso porque permite la ejecución de código personalizado, que podría ser utilizado de manera malintencionada para dañar al usuario: la macro.

## Macros

El caso de uso inicial de las macros en documentos MS es ejecutar tareas repetitivas fácilmente al "grabarlas" una vez y luego "reproducirlas" repetidamente. Se pueden crear macros sin tener conocimientos de programación simplemente grabando los clics en los botones y los atajos de teclado, entonces MS Office traducirá la grabación en una serie de comandos que se ejecutarán como un "pequeño programa" alojado dentro de nuestros documentos.

Cuando empezamos a profundizar en cómo funcionan las macros, nos damos cuenta de cómo pueden utilizarse con fines maliciosos y por qué son tan populares en los ataques de phishing para infectar computadoras comparadas con otras técnicas. Para empezar, las macros se almacenan como código escrito en el lenguaje de programación Visual Basic para Aplicaciones (VBA), el cual está bien documentado, es sencillo de escribir y muy potente. En general, Visual Basic también se utiliza para escribir programas independientes completos, por lo que tiene capacidad para hacer cosas fuera del alcance del documento, como descargar y ejecutar archivos y alterar la configuración del sistema, por ejemplo. La mayoría de los comandos asociados para estas tareas también están disponibles en las macros de MS Office, por lo que podemos escribir macros que aprovechen los comandos avanzados

disponibles y utilizarlos para ejecutar tareas dañinas, como descargar y ejecutar malware más evolucionado, eliminar archivos, etc.

Por otra parte, ejecutar macros desde documentos es realmente sencillo. Los creadores de documentos pueden configurarlas para que se ejecuten automáticamente al abrir el archivo, al hacer clic en un botón, enlace o cualquier otro elemento, entre otros desencadenantes. En el caso de archivos desconocidos, MS Office nos advertirá de que sus macros pueden ser peligrosas y las bloqueará, pero normalmente estamos a uno o dos clics de desactivar esta protección y ejecutar las macros de todos modos. Esta situación resulta muy atractiva para que los actores maliciosos utilicen macros y nos convenzan de que es seguro ejecutarlas mediante argumentos persuasivos, propios de cada campaña de phishing.

Comparando los archivos PDF con los documentos de MS Office que contienen macros para actividades maliciosas, los documentos de MS Office ofrecen más posibilidades de ejecución de comandos en los dispositivos que los abren, lo que los hace más potentes, así como más populares, que los PDF. Además, esa flexibilidad es la razón por la que nos enfocamos en la utilización maliciosa de macros permitidas dentro de documentos en lugar de otras formas de utilización maliciosa de documentos de MS Office. Si está interesado en conocer otras formas de utilizar estos archivos para vulnerar a los usuarios de versiones antiguas de Office, al final le proporcionamos enlaces a otras referencias.

**Vulnerabilidades de Office**
Existen muchas formas documentadas de explotar macros o documentos de Office en general, sin embargo, muchas de las más creativas no pueden explotarse en instancias completamente actualizadas de MS Office.

Como cualquier otro programa, MS Office puede tener vulnerabilidades conocidas o desconocidas que podrían permitir comprometer un dispositivo, incluso sin utilizar macros.

## Los formatos de documentos "antiguos" y "nuevos" de MS Office

Desde 2003, Microsoft Office cambió la forma en que se crean los documentos de manera predeterminada, incluyendo nuevas extensiones de archivo, de modo que los nuevos documentos de MS Word se almacenan con la extensión ".docx" en lugar de ".doc" y así sucesivamente. Aunque la estructura interna de estos dos formatos de archivo sea diferente, las macros se almacenan de manera similar, por lo que las directrices proporcionadas en este material se aplican tanto a los formatos de archivo antiguos como a los nuevos.

Si le interesa profundizar más en las convenciones para almacenar macros y muchos otros tipos de objetos, puede investigar más sobre Vinculación e incrustación de objetos (u OLE), que sigue utilizándose y adaptándose a los nuevos formatos de archivo, incluidos los documentos de MS Office

## Analizar documentos de MS Office

Para comenzar a explorar las formas de detectar cuándo se incluyen macros en los documentos de MS Office, utilizaremos oledump.py, una herramienta de Python desarrollada por Didier Stevens, el mismo autor de las herramientas propuestas anteriormente en la sección anterior dedicada a los PDF. También utilizaremos una serie de archivos de ejemplo para mostrar cómo funcionan los documentos de MS Office y cómo se pueden detectar y analizar las macros, también extraídos de los materiales de capacitación de Didier Stevens.

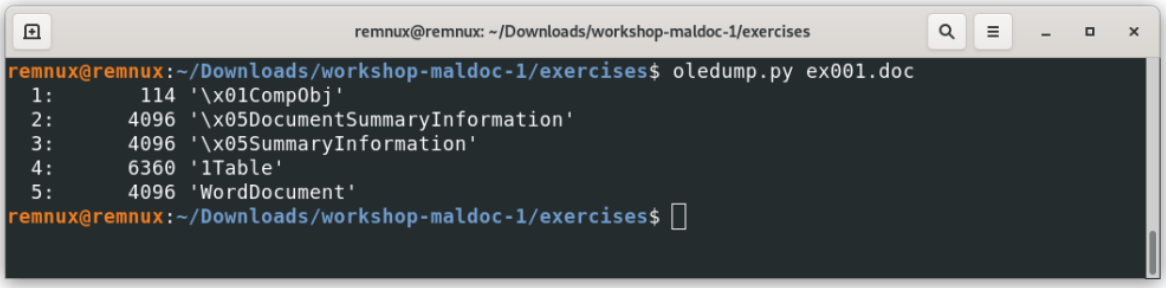
El flujo de trabajo para iniciar el análisis de los documentos de MS Office es muy similar al que utilizamos en los PDF. Primero, enumeramos los distintos elementos presentes en el archivo, identificamos los objetos interesantes en términos de seguridad y, luego, intentamos obtener el contenido real de esos elementos para determinar si hay algo dañino en ellos.

## Oledump.py

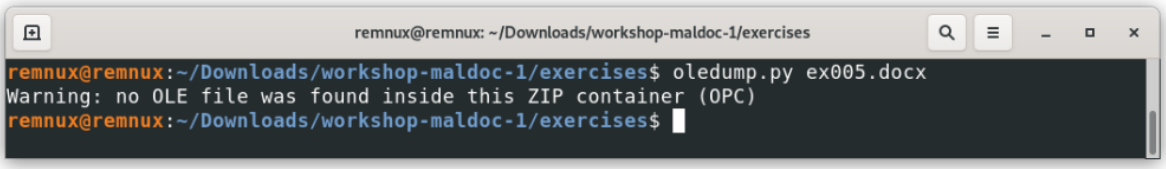
El uso principal de esta herramienta es listar cualquier objeto OLE incluido en un archivo específico, y mostrar su contenido. El uso básico de la herramienta es el siguiente:

```
oledump.py ex001.doc
```

Donde ex001.doc es el nombre del documento que queremos analizar



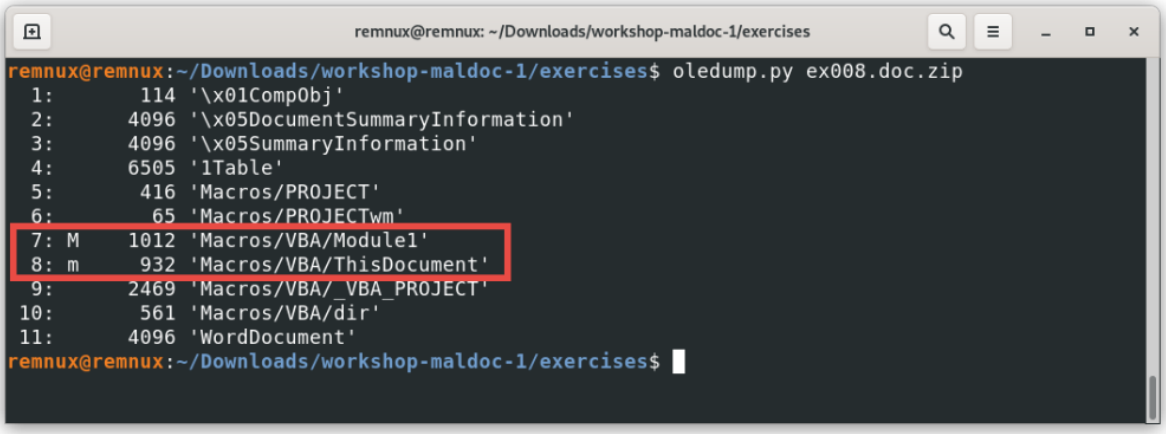

En este caso podemos ver todos los elementos para un archivo sin macros ni otros objetos inusuales incrustados en un archivo de tipo "antiguo" de MS Office, haciendo el mismo experimento para un archivo con el "nuevo" formato (posterior a MS Office 2003), obtendremos algo como esto.




Aquí podemos ver que más elementos en el ejemplo .doc están almacenados como objetos OLE, mientras que en el ejemplo .docx, tenemos un documento funcional sin utilizar objetos OLE. Esto es debido a que los documentos .docx están empaquetados como un archivo .zip que contiene principalmente archivos .xml (incluso puede intentar cambiar un archivo

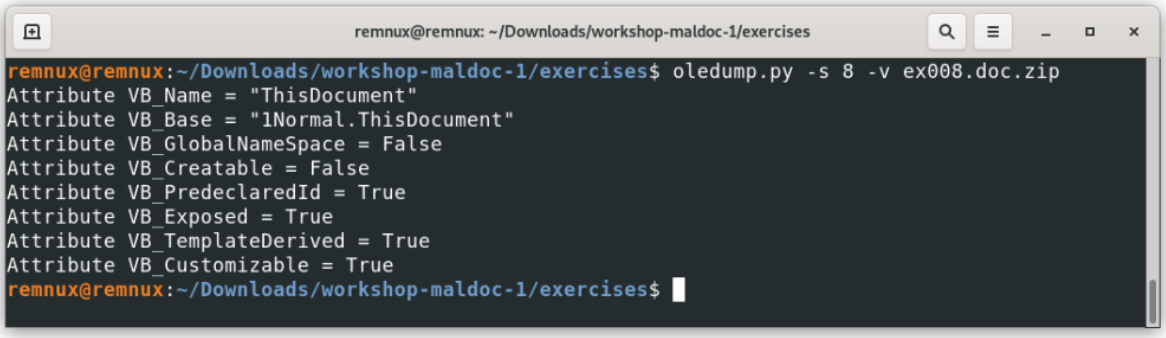
.docx|.xlsx|.pptx seguro a la extensión .zip y abrirlo), y sólo utilizan datos con formato OLE cuando es necesario.

Cuando abramos un archivo con macros, el resultado incluirá nuevos elementos:




Si observamos bien este ejemplo, podemos ver que el archivo que pasamos al comando es un archivo zip. En este caso, se trata de un archivo .zip que contiene un documento de MS word. Además, el archivo .zip está protegido con la contraseña "infectado". Esta es una práctica común en la comunidad de análisis de malware, y oledump.py lo considera como una entrada válida y gestiona toda la descompresión, y nos pasa el documento para su análisis automáticamente.

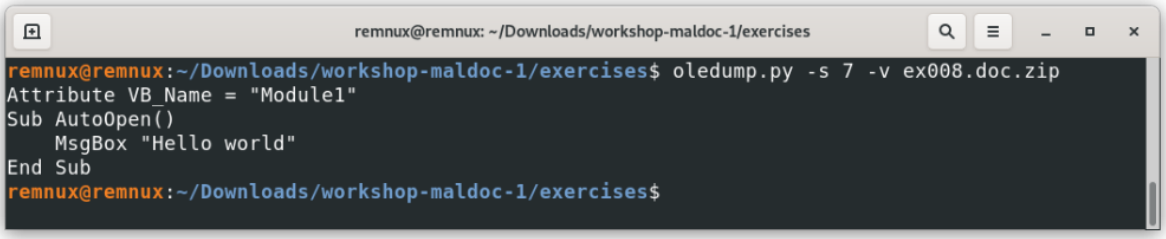
En esta salida, vemos 2 objetos que son diferentes, y se identifican con la letra "m" o "M". Esto significa que esos objetos específicos contienen macros. Utilicemos el comando `-s` en oledump.py para ver el contenido de esos streams, empezando por el objeto (o stream 8)




Usamos el comando `-s` para seleccionar el objeto identificado con el número 8, y el comando `-v` para descomprimir el contenido porque VBA puede comprimir código por defecto, por lo que es una práctica segura incluir este comando cuando se solicitan objetos con macros.

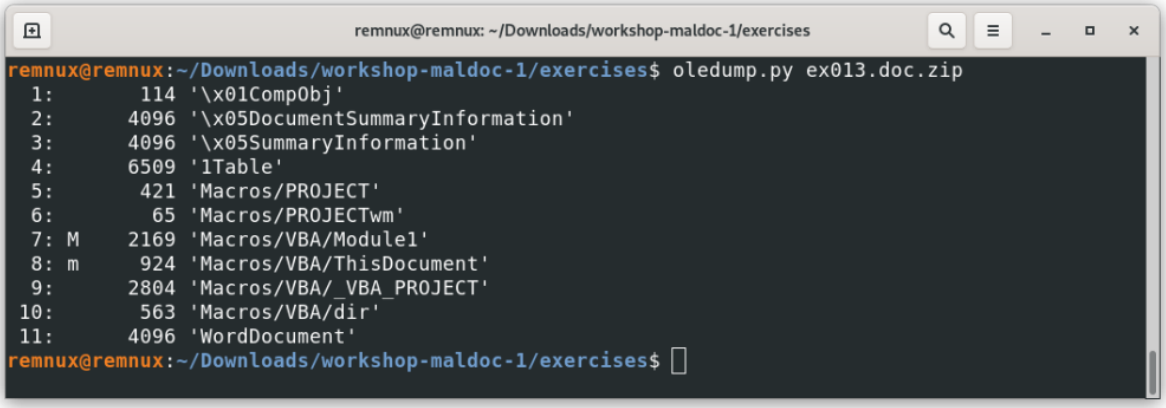
Ahora bien, mirando el contenido, vemos algunas declaraciones de atributos, estas son realizadas por defecto por VBA y ni siquiera son visibles para el creador del documento, luego, este código no se considera personalizado o perjudicial a nuestros efectos. Dicho esto, el código que la herramienta considera inofensivo se identifica con una "m" minúscula,

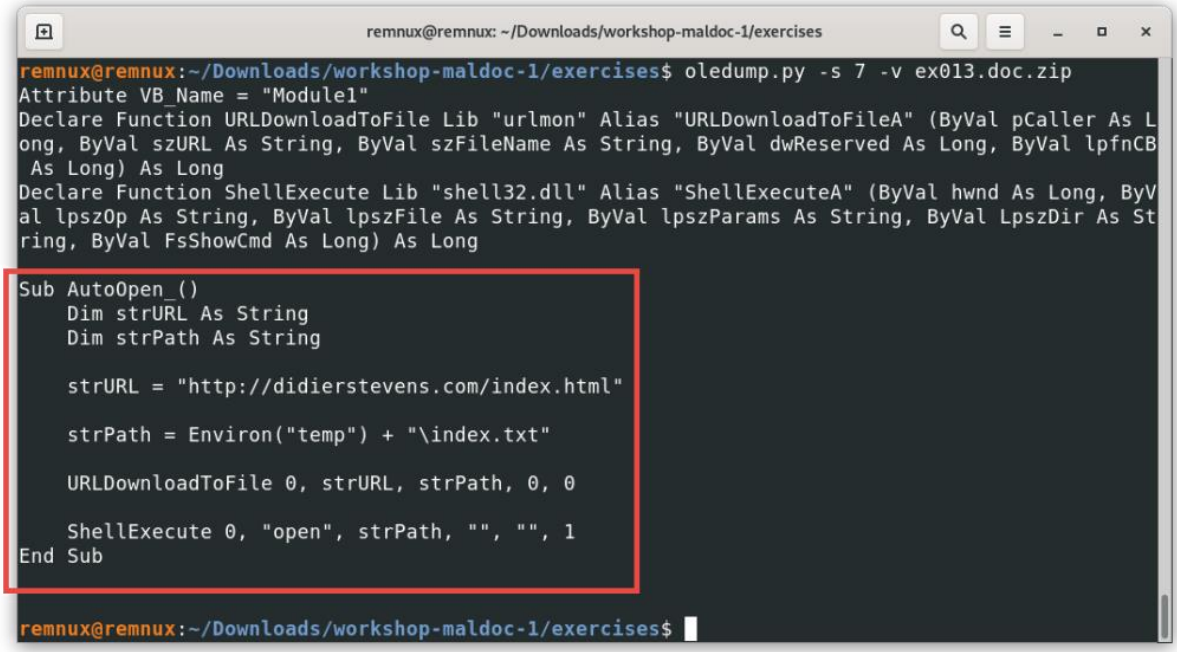
asumiéndolo como seguro y menos interesante para un análisis posterior. Vamos a analizar el otro objeto que contiene una macro.




Para dar un poco de contexto, esta macro utiliza el comando MsgBox que lanza una ventana de diálogo con el mensaje "Hello world" en este caso. Además, AutoOpen() indica al programa que esta macro debe ejecutarse automáticamente al abrir el archivo. En la práctica, un documento desconocido que intente ejecutar una macro AutoOpen() activará una alerta de seguridad. Sin embargo, dependiendo del contexto, el usuario podría ser engañado para pasar por alto la advertencia y ejecutar la macro de todos modos.

Para explorar la forma en que esto podría utilizarse indebidamente, verifiquemos el siguiente archivo

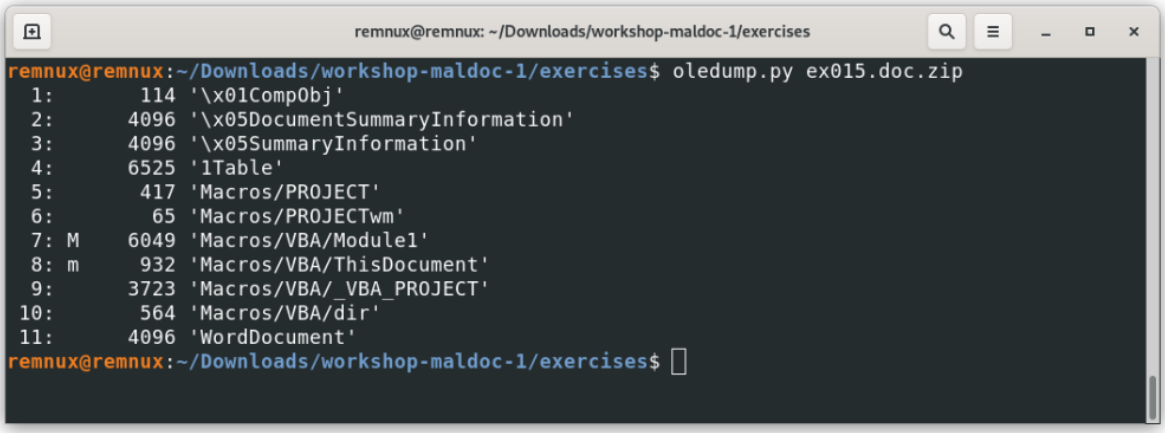


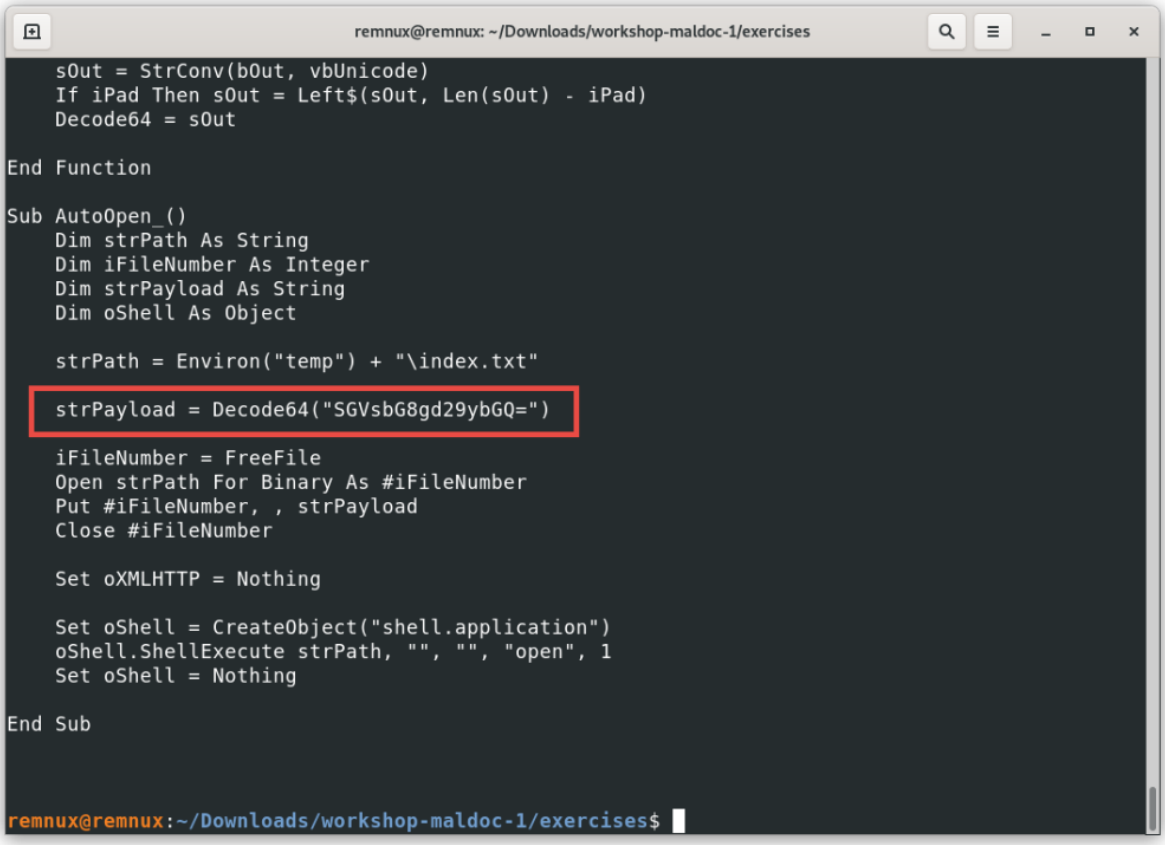



Aquí, la macro de interés es un poco más compleja que una ventana de diálogo con un mensaje de texto. Podemos ver que, una vez más, se utiliza la función AutoOpen() para ejecutar el código siguiente al abrir el documento. Analizando el código, y con un poco de ayuda para verificar los comandos utilizados, podemos deducir que la macro intenta descargar el contenido de una URL en un archivo dentro del directorio temporal de nuestra máquina y ejecutar el archivo que se haya descargado.

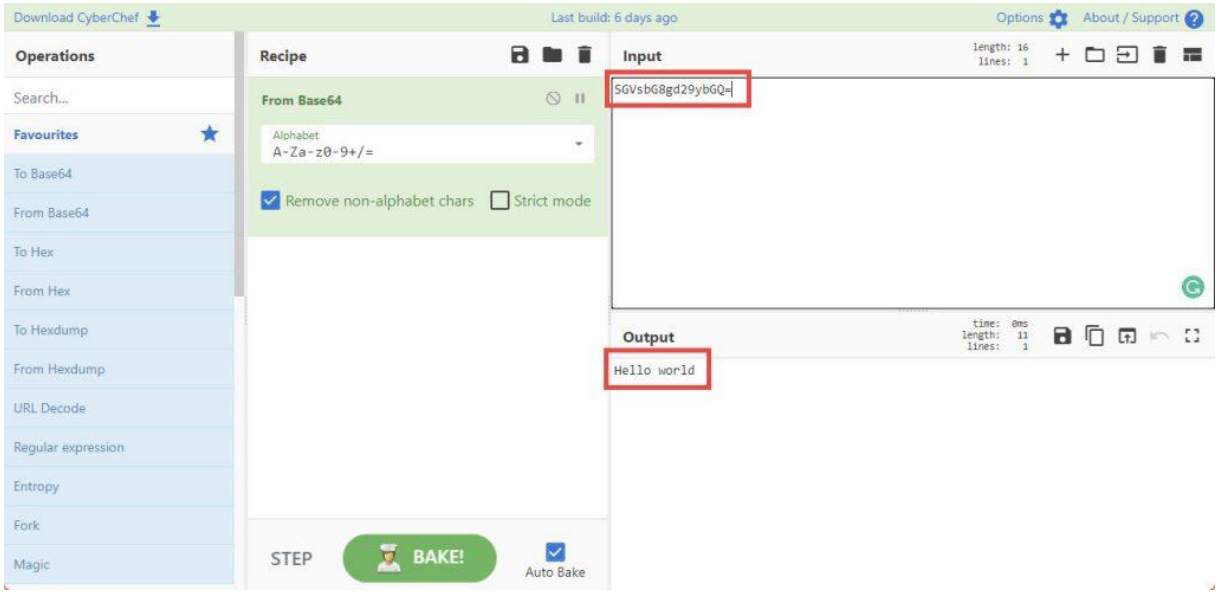
En este caso, parece que el archivo está llenando un archivo de texto, que debería ser inofensivo. Sin embargo, con la URL correcta y el tipo de archivo correcto utilizado para descargar el contenido, una macro como ésta puede escribir y ejecutar programas u otros artefactos dañinos sin mucha interacción del usuario o incluso sin su conocimiento. Además, vale la pena mencionar que esta macro es una versión simplificada de amenazas más reales, que suelen ofuscar su código para evitar ser detectadas por el software antivirus y pueden ejecutar acciones más elaboradas, como por ejemplo agregar el malware descargado a programas de inicio o tareas programadas, haciendo que el malware sea persistente en el tiempo, entre otras acciones.

En ocasiones, analizar macros más complejas requerirá habilidades no cubiertas en este material, como descifrar código y descubrir cómo, mediante el uso de diferentes comandos y estructuras de datos, podemos obtener el código final ejecutado para poder entender lo que hace (desofuscación). Un ejemplo aún muy sencillo que muestra una técnica común para ofuscar contenidos se encuentra en el siguiente archivo.




Aquí, en lugar de usar texto plano, el creador de la macro utilizó el esquema de codificación base64, el cual dificulta la lectura de la carga útil que se intenta ejecutar. Para este ejemplo, hay muchas herramientas que nos pueden ayudar a decodificar esa variable, una de ellas es CyberChef, una aplicación web donde podemos ingresar datos y ejecutar operaciones sobre ellos para obtener un resultado. En este caso, tenemos:

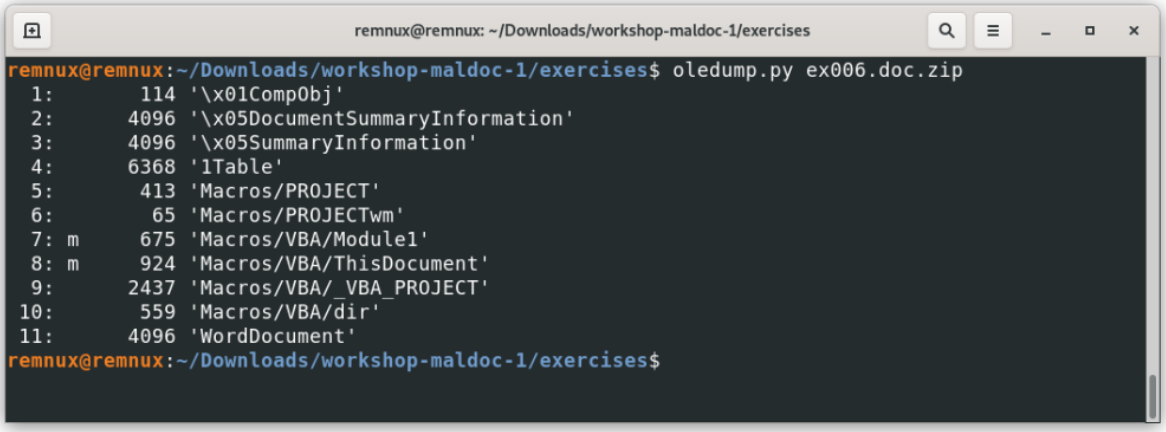



Con estos ejemplos y referencias, deberíamos ser capaces de distinguir si un archivo tiene macros incrustadas usando oledump.py, si un archivo es interesante para analizar más a fondo, y en caso de que su código sea lo suficientemente simple, qué intenta hacer la macro. En el caso de encontrar documentos con macros muy complejas, el consejo es buscar ayuda para analizar el archivo con mayor profundidad y nunca intentar ejecutar el archivo en nuestros entornos porque podría tener efectos terribles en nuestros dispositivos en caso de que se infecten.

Ahora que hemos aprendido algunos flujos de trabajo introductorios para analizar archivos PDF y de MS Office, estamos listos para revisar algunas estrategias defensivas para protegernos de documentos maliciosos en la siguiente y última parte.

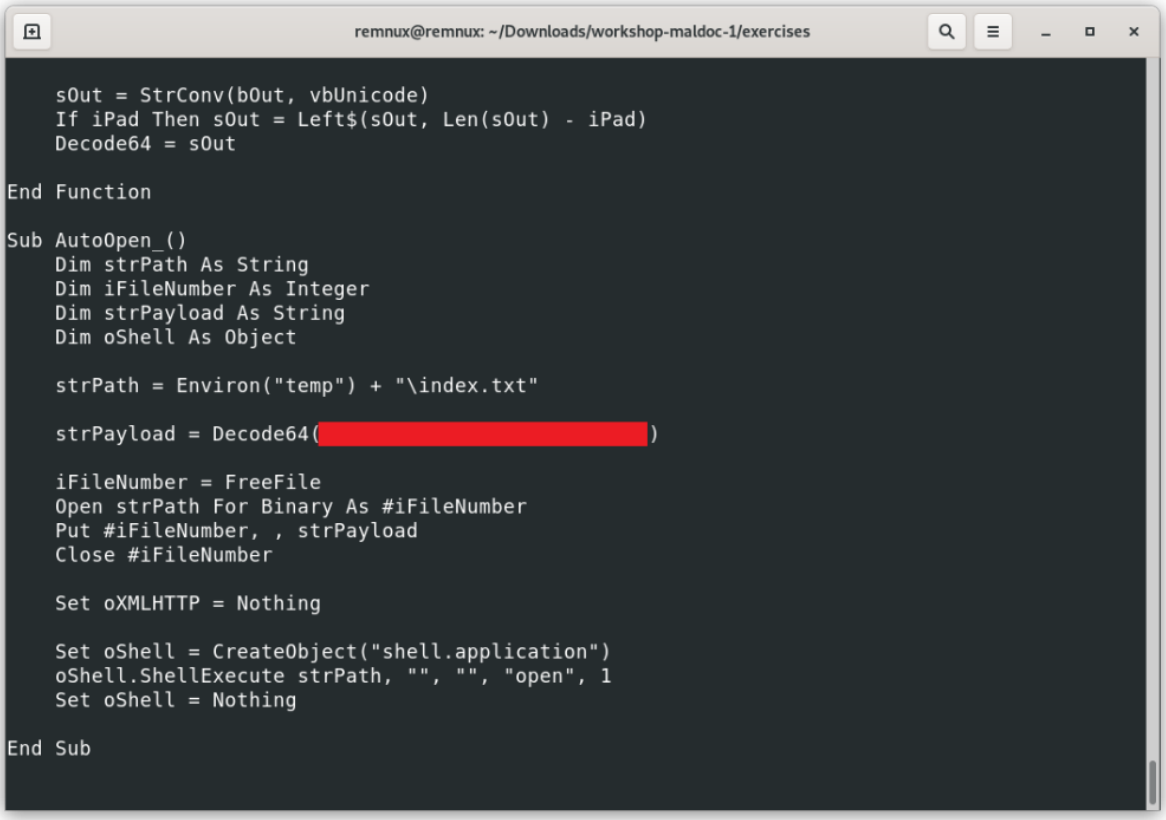
## Desafíos

**Pregunta:** aplicando el mismo flujo de trabajo al archivo ex006.doc.zip obtenemos este resultado. ¿Cuál de las siguientes hipótesis se confirma con la información obtenida hasta ahora?



- 1. El archivo no tiene macros
- 2. El archivo tiene macros creadas a medida como cualquiera de los otros ejemplos tratados
- 3. De alguna manera, el archivo tiene macros inofensivas para el sistema, pero no macros creadas a medida

**Pregunta:** aplicando el mismo flujo de trabajo al archivo ex016.doc.zip vemos una macro similar a la última tratada anteriormente, donde la macro decodifica algo en base64. ¿Qué se pasa a la función Decode64? (pista: aaaaaaaaaaaaa.aaaaaaa.aaaa)




- Quiero ver la respuesta

# Y ahora, ¿qué sigue?

Después de tener una mejor idea de cómo se pueden evaluar documentos PDF y MS Office en busca de código malicioso, podemos entender mejor cómo proponer medidas defensivas, y también, podemos revisar los consejos finales sobre cómo llevar a cabo este tipo de evaluación inicial.

# Bonus: Lecturas adicionales sobre la seguridad de MS Office

- Oletools: otra herramienta muy conocida para analizar archivos de MS Office
- Lista de vulnerabilidades conocidas de Microsoft Office (la mayoría no implican macros)
- CVE-2022-30190 o nombre en clave "Follina": una vulnerabilidad reciente y popular que abusa de los documentos para interactuar con funciones de solución de problemas de Windows.
- "Uncompromised: Unpacking a malicious Excel macro", un caso interesante que explora un archivo malicioso paso a paso.
- Analyzing Malicious Documents Cheat Sheet, una guía rápida de Lenny Zeltser para analizar documentos sospechosos.