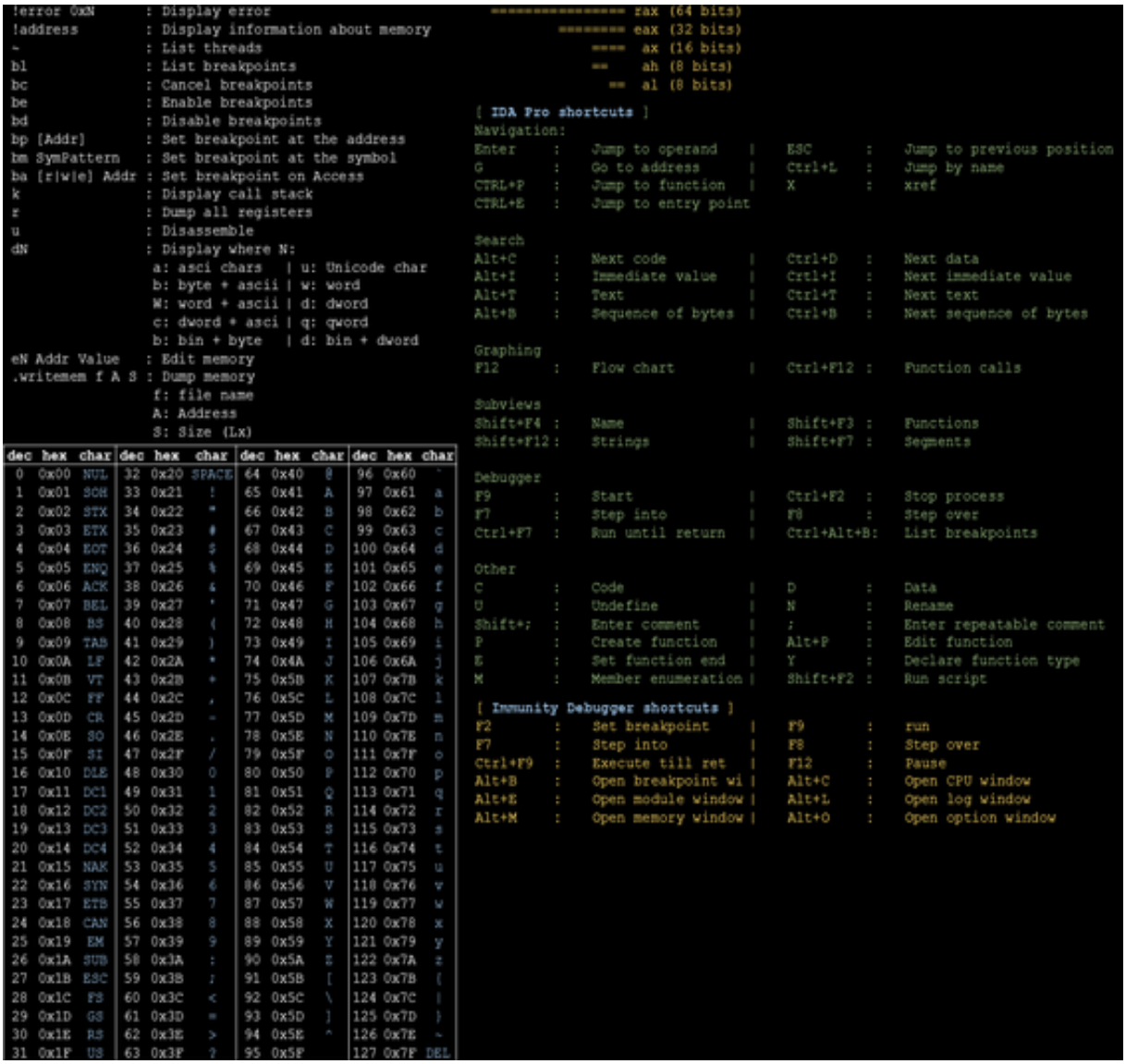


# Máster en Análisis de Malware, Reversing y Bug Hunting

ENIIT INNOVA IT BUSINESS SCHOOL

Campus Internacional CIBERSEGURIDAD

UCAM UNIVERSIDAD CATÓLICA DE MURCIA

## MÓDULO 3 – Análisis de Código Fuente

https://campusciberseguridad.com

# 1. INTRODUCCIÓN

Esta es la segunda y última evaluación de la asignatura de análisis de código fuente. En ella, nos centraremos en los dos últimos temas, los cuales versan sobre el uso de todas las técnicas y definiciones que hemos estado viendo en los temas anteriores de forma práctica.

- El primer ejercicio está basado en código de bajo nivel (ensamblador x86)
- El segundo ejercicio está basado en código de alto nivel (lenguaje C)

## 1.1. OBJETIVOS DE APRENDIZAJE

El aprendizaje de esta práctica os mete de lleno en el análisis de código, tanto fuente como procedente de un desemsamblado.

Esta práctica no solo se asemeja, sino que es prácticamente parte del proceso que una persona que analiza malware de forma profesional efectúa en su día a día.

# 2. EVALUACIÓN

Cada uno de los dos ejercicios está dividido en varias tareas con dificultad ascendente. La puntuación es idéntica para cada uno de los ejercicios y para cada una de las tareas dentro de cada ejercicio.

**Criterios de evaluación**

- Claridad en las explicaciones. Incluso una solución que está mal se puntuará de forma parcialmente positiva si se ha explicado e ilustrado de forma correcta y apropiada. Ya en vuestras calificaciones se os indicará que está mal.

- No se tendrán en cuenta respuestas que no razonen los hallazgos encontrados. Ejemplo:

> "*Esta función parece que cifra un archivo*"
>
> vs
>
> "*La función toma como parámetro un archivo (su descriptor) y lee el contenido el cual va almacenando en un buffer que se recorre en un bucle for. Este bucle va iterando sobre cada byte cifrando el contenido con RC4. La función no retorna nada, ya que su objetivo es modificar lo que le es pasado por parámetro*."

- No es necesario que os expliquéis de manera formal. Se os anima a hacerlo, puesto que es la forma de expresión que se os pedirá de manera profesional cuando redactéis un informe.

- **En ningún caso se puntua negativamente**, solo positivo. Si algo está mal se os dirá y corregirá en vuestra evaluación personal.

## 3. FORMATO DE ENTREGA

Podéis usar el formato en el que os encontréis más cómodos siempre y cuando sea un formato conocido: Word, OpenOffice, PDF, etc.

Idealmente, el formato sería en PDF.

## 4. AVISO

**IMPORTANTE IMPORTANTE IMPORTANTE**

**NO USEIS VUESTRO SISTEMA**

**USAD LA MÁQUINA VIRTUAL**

**DESCONECTAD LA RED DE LA MÁQUINA VIRTUAL**

**DESECHAD EL ESTADO DE LA MÁQUINA VIRTUAL UNA VEZ FINALIZADOS LOS EJERCICIOS**

**IMPORTANTE IMPORTANTE IMPORTANTE**

# 5. EJERCICIO 1

Para este ejercicio os descargaréis un ejecutable (**NO ES MALWARE**) creado específicamente para vosotros.

Se trata de un ejecutable para sistemas Linux (32 bits) basados en arquitectura x86. Para ayudaros, se le han incluído los símbolos de depuración.

Al ejecutar el programa, **este examinará ciertas condiciones** y no se ejecutará de forma real sino se cumplen. Es lo que haría un malware en un ambiente real: comprobar que se dan las circunstancias oportunas para ejecutar la parte maliciosa.

Podéis utilizar cualquier herramienta. Lo fundamental en esta prueba es que aprendáis a desenvolveros con un desensamblador e interpretar el código.

**NOTA**: El archivo de encuentra en el área de descargas de la asignatura. Se trata de un archivo ZIP cuya clave es "**infected"**. Esta clave es un convenio (o estándar de facto) en la industria. Los analistas se suelen intercambiar muestras de esta manera y por convención la clave siempre suele ser esa.

De nuevo, no se trata de un programa malicioso.

## 5.1. TAREA 1

Descubrid a través de la lectura del código de la función 'main' desensamblada como ejecutar el programa.

Es decir, **NO ES NECESARIO MANIPULAR EL PROGRAMA**. Simplemente entendiendo como funciona 'main' es posible adivinar como evadir las protecciones y ejecutar el programa.

Se pide:

- Detallad y explicad mostrando el código ensamblador cuales son las condiciones para que el programa se ejecute.

- Demostrad (mediante capturas de pantalla) la ejecución del programa.

## 5.2. TAREA 2

Detallad que hace el programa realmente.

Se pide:

- El programa, además de "main", solo tiene una única función. Encontradla y explicad **que hace y como lo hace** esta función mostrando el código ensamblador correspondiente.

### 5.3. PISTAS

- El ensamblador es x86 32 bits
- Comenzar desensamblando el código (ajustad la sintáxis de x86 a vuestro gusto: Intel o AT&T)
- Comenzad por 'main', es el punto de entrada de todos los programas, es fácil localizarlo
- ¿Qué va haciendo 'main'? ¿Se están cumpliendo todas las condiciones que inspecciona el programa?
- Además de 'main' el programa contiene otra función, el resto son llamadas a funciones externas.
- Podéis usad todas las herramientas que queráis, pero de nuevo, se resuelve con un buen desensamblador y estudiando el código parte a parte. Es decir, analizando.

## 6. EJERCICIO 2

Para este ejercicio emplearemos el código fuente del ransomware Babuk, el cual **NO DEBEIS COMPILAR NI MUCHO MENOS EJECUTAR**.

Babuk es un ransomware que tuvo un significativo auge a comienzos de 2021 y cuyo código fue liberado de forma súbita por sus creadores. Este malware es multiplataforma, existiendo versiones para Linux, Windows, routers y varios dispositivos IoT. Su código está escrito en los lenguajes Go y C.

El código fuente está disponible en el siguiente repositorio:

https://github.com/Hildaboo/BabukRansomwareSourceCode

### 6.1. TAREA 1

Una de las características del malware es que ejecuta una rutina o función que detiene ciertos procesos del sistema para evitar su detección por parte de estos. En otras ocaciones suele detectar la presencia de herramientas de análisis y detiene su propia ejecución o la camufla para evitar ser analizada.

Esta tarea consiste en la localización de la función (ojo, no donde esta es llamada) que detiene los procesos en **la versión para sistemas Windows** del malware Babuk.

Es decir:

- Localizad el código de la función (de nuevo, NO DONDE ES LLAMADA).
- Realizad un análisis de código de la función.

- Enumerar los procesos que tiene en cuenta esta función para que sean detenenidos.

Documentad profusamente el proceso que habéis realizado (capturas de pantalla, evidencias).
Cuanto más detallada (en calidad) esté hecho el análisis mayor será la puntuación.

## 6.2. TAREA 2

Aquí, se encuentra la función “main” del código del cifrador escrito en Go **de la versión para dispositivos NAS** de Babuk.

Esa función representa el punto de entrada del ejecutable. Dicho ejecutable es el que utilizan los cibercriminales para cifrar un sistema. Es decir, explotan un sistema, filtran el ejecutable y este se encarga de ir cifrando todos los archivos de la víctima en ese sistema.

Comentad el funcionamiento de la función referenciada con todo el detalle posible.
Si no entendéis algunas partes saltadlas y proseguid o al menos comentad lo que creeis que es.

Código:
https://github.com/Hildaboo/BabukRansomwareSourceCode/blob/main/nas/enc/main.go#L810