Emanuel Rodríguez Oviedo

c.2022108678

# Proyecto 1: Identificación de debilidades mediante análisis estático de código C/C++

Seguridad del Software

IC-8071

I Semestre 2024

# Contenido



# Glosario

CWE: Common Weakness Enumeration

VP: Verdadero Positivo

FP: Falso Positivo

VN: Verdadero Negativo

FN: Falso Negativo

LoC: Lines of Code

# Descripción del problema:

En el siguiente proyecto se pretende analizar una base de código a partir de la cual se logren encontrar posibles debilidades asociadas a patrones CWE. Esto se deberá lograr a partir del uso de una serie de herramientas disponibles para la distribución Kali Linux en su versión Kali Rolling 2024.1 de 64 bits, al ejecutar por medio de la terminal cada aplicación aportando la base de código y una serie de opciones de configuración adaptadas a obtener los mejores resultados posibles según el criterio del desarrollador del proyecto.

## Selección de programa y archivos:

Primeramente, se realiza un proceso de búsqueda exhaustivo en los repositorios de paquetes de Ubuntu/Debian ejecutando el comando:

$ apt-get source <biblioteca u aplicación>

A partir de ello se escoge la herramienta Redis en su versión 7.0.15. Por medio de la exploración de su archivo fuente y al utilizar dos scripts de shellcode (véase los scripts de nombre “scriptFindFilesOnSize.sh” y el script “filterFilesOnIncludes.sh” se escogen los archivos “t_set.c” y “bitops.c”, encontrados en el código fuente del programa, por sus características (líneas de código mayor o igual a 1000 y cantidad pequeña de directivas de preprocesamiento #include).

## Configuración de las herramientas:

Para lograr llegar a conseguir los mejores resultados posibles (esto es, los resultados de usar las herramientas que se presentarán a continuación logran aportar una cantidad de advertencias que se aproxima lo más cercanamente a la cantidad de verdaderos positivos en el código para la capacidad de la herramienta, mientras se mantiene una baja cantidad de falsos positivos y falsos negativos.

**Flawfinder:**

Para confirar flawfinder, se utilizaron los parámetros:

- --context: aporta x y x
- --minlevel=0: establece el mínimo nivel de las advertencias por mostrar, es decir, muestra cualquier tipo de advertencia que el programa pueda dar.

El comando completo es:

**Cppcheck**:

Con respecto a cppcheck, se obtuvo lo siguiente:

- --enable=all: permite a cppcheck realizar un análisis exhaustivo del código a procesar.
- --force:
- -D BYTE_ORDER=LITTLE_ENDIAN: necesario para que el procesamiento funcione, esta macro se define por medio del archivo <stdint.h> perteneciente a la configuración de reddis.
- -D BIG_ENDIAN=4321 -D LITTLE_ENDIAN=1234: se definenen ambos valores para probar el buen funcionamiento de la definición de la macro, sus valores reales no interesan en este momento pues esto está fuera del alcance del análisis de cppcheck
- -I /home/kali/redis-7.0.15/src \ -I /home/kali/redis-7.0.15/deps/hdr_histogram/ : ambos includes son necesarios para que cppcheck pueda comprender correctamente el funcionamiento del archivo pues son parte de las directivas de preprocesamiento include que se establecen en el archvio.
- suppress=missingIncludeSystem: se utiliza para suprimir los mensajes de advertencia sobre includes faltantes que son irrelevantes al análisis, pues cppcheck no necesita analizar el contenido de los mismos ni saber a profundidad que definen para generar un análisis.
- Path del archivo: path para los archivos de prueba.

cppcheck --enable=all --force --inconclusive \

-D BYTE_ORDER=LITTLE_ENDIAN \

-D BIG_ENDIAN=4321 -D LITTLE_ENDIAN=1234 \

-I /home/kali/redis-7.0.15/src \

-I /home/kali/redis-7.0.15/deps/hdr_histogram/ \

--suppress=missingIncludeSystem /home/kali/redis-7.0.15/src/t_set.c

## Análisis de Resultados

Con el fin de lograr determinar la calidad y veracidad de resultados, mientras al mismo tiempo documentar las advertencias adquiridas gracias al uso de estas herramientas, se decide adaptar los resultados y colocarlos en una tabla con un formato estándar, lo que permite la siguiente visualización:

**bitops.c (1265 LoC)**

| Línea | CWE | Descripción breve | Justificación | Flawfinder | Cppcheck | Manual |
|---|---|---|---|---|---|---|
| 44 | CWE-120 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues | **FP**: El array está apropiadamente inicializado, siendo un array estático en el cual solo se realizan operaciones de lectura y su tamaño siempre se da entre los índices adecuados (0-255) | ✓ | | |
| 210 | CWE-398 | The scope of the variable 'bit' can be reduced. [variableScope] uint64_t byte, bit, byteval, bitval, j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza.. | | ✓ | |
| 210 | CWE-398 | The scope of the variable 'bitval' can be reduced. [variableScope] uint64_t byte, bit, byteval, bitval, j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 210 | CWE-398 | The scope of the variable 'byteval' can be reduced. [variableScope] uint64_t byte, bit, byteval, bitval, j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 210 | CWE-398 | The scope of the variable 'byteval' can be reduced. [variableScope] uint64_t byte, bit, byteval, bitval, j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 225 | CWE-197 | Numeric truncation error: presente en el casteo de int64_t a un valor uint64_t. | **FN:** la transformación de los valores puede ser crítica puesto que asume complemento a dos, lo que implica transformación de data que podría explotarse o manipularse erróneamente. | | | ✓ |
| 229 | CWE-665 | Parameter 'p' can be declared as pointer to const | **VP:** El valor p debería mejor ser declarado como const unsigned char* pues existe la posibilidad | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | [constParameterPointer]<br>uint64_t getUnsignedBitfield(unsigned char *p, uint64_t offset, uint64_t bits) { | de que se modifique maliciosamente p en la función o que un mal diseño de API pueda llegar a mandar valores const y no const que podrían requerir casteos que podrían llegar a operaciones no seguras. | | | |
| 230 | CWE-398 | The scope of the variable 'bit' can be reduced. [variableScope]<br>uint64_t byte, bit, byteval, bitval, j, value = 0; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 230 | CWE-398 | The scope of the variable 'bitval' can be reduced. [variableScope]<br>uint64_t byte, bit, byteval, bitval, j, value = 0; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 230 | CWE-398 | The scope of the variable 'bitval' can be reduced. [variableScope]<br>uint64_t byte, bit, byteval, bitval, j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 230 | CWE-398 | The scope of the variable 'byteval' can be reduced. [variableScope]<br>uint64_t byte, bit, byteval, bitval, j, value = 0; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 382 | CWE-561 | The function 'printBits' is never used. [unusedFunction]<br>void printBits(unsigned char *p, unsigned long count) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 382 | CWE-561 | La descripción del función menciona que solo se usa en procesos de debugging | **FN:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código. | | | |
| 382 | CWE-665 | Parameter 'p' can be declared as pointer to const [constParameterPointer]<br>void printBits(unsigned char *p, unsigned long count) { | **VP:** El valor p debería mejor ser declarado como const unsigned char* pues existe la posibilidad de que se modifique maliciosamente p en la función o que un mal diseño de API pueda llegar a mandar valores const y no const que podrían requerir casteos que podrían llegar a operaciones no seguras. | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 383 | CWE-398 | The scope of the variable 'byte' can be reduced. [variableScope] unsigned long j, i, byte; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 388 | CWE-134 | If format strings can be influenced by an attacker, they can be exploited. | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza, pero no representa un riesgo real | ✓ | | |
| 389 | CWE-134 | If format strings can be influenced by an attacker, they can be exploited. | **VP:** El valor 'count' podría declararse como un valor más grande que el tamaño del Sting apuntado por el puntero. | ✓ | | |
| 391 | CWE-134 | If format strings can be influenced by an attacker, they can be exploited. | **FP:** La llamada a la función printf contiene un carácter de nueva línea, no ningún input que pueda ser controlado por el usuario. | ✓ | | |
| 416 | CWE-665 | Variable 'err' can be declared as pointer to const [constVariablePointer] char *err = "bit offset is not an integer or out of range"; | **VP:** El valor err debería mejor ser declarado como const unsigned char* pues existe la posibilidad de que se modifique maliciosamente p en la función o que un mal diseño de API pueda llegar a mandar valores const y no const que podrían requerir casteos que podrían llegar a operaciones no seguras. | | ✓ | |
| 430 | CWE-20 | Improper Input Validation: | **FN:** No se realiza la adecuada validación para el valor de los bits con números no negativos antes de usarlo en cálculos podría llevar a lograr comportamientos inesperados | | | ✓ |
| 433 | CWE-433 | Integer Overflow or Wraparound | **FN:** números La multiplicación de loffset *= bits y su operación de shift son candidatos a integer overflow. | | | ✓ |
| 451 | CWE-665 | Variable 'p' can be declared as pointer to const [constVariablePointer] char *p = o->ptr; | **VP:** El valor p debería mejor ser declarado como const unsigned char* pues existe la posibilidad de que se modifique maliciosamente p en la función o que un mal diseño de API pueda llegar a mandar valores const y no const que podrían requerir casteos que podrían llegar a operaciones no seguras. | | ✓ | |
| 452 | CWE-665 | Variable 'err' can be declared as pointer to const [constVariablePointer] char *err = "Invalid bitfield type. Use something like i16 u8. Note that u64 is not supported but i64 is." | **VP:** El valor err debería mejor ser declarado como const unsigned char* pues existe la posibilidad de que se modifique maliciosamente p en la función o que un mal diseño de API pueda llegar a mandar valores const y no const que podrían | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | requerir casteos que podrían llegar a operaciones no seguras. | | | |
| 464 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **VP:** Al no verificar que se está usando un valor de p que contenga una terminación, la función strlen(p+1) podría llevar a leer más allá del límite de p. | ✓ | | |
| 488 | CWE-908 | lookupStringForBitCommand Function | FN: Si se crea un nuevo objeto sdsnewlen, sin explícitamente inicializar la memoria, se podría tener problemas si se manipula ese Sting de memoria sin asegurar su correcta inicialización. | | | |
| 532 | CWE-561 | The function 'setbitCommand' is never used. [unusedFunction] void setbitCommand(client *c){ | VP: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 578 | CWE-561 | The function 'getbitCommand' is never used. [unusedFunction] void getbitCommand(client *c){ | VP: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 581 | CWE-120 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues | **FP:** Esta advertencia es un falso positivo pues la función ll2string que acepta a llbuf, comprueba el tamaño de llbuf de tal modo que asegura que nunca se llegue a manipular datos más allá de su límite. | ✓ | | |
| 607 | CWE-561 | The function 'bitopCommand' is never used. [unusedFunction] void bitopCommand(client *c){ | VP: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | prevención de la ejecución de código muerto o no. | | | |
| 689 | CWE-120 | Does not check for buffer overflows when copying to destination | **FP:** el código asegura por medio de condicionales que los tamaños de los valores copiados al buffer no generen un overflow | ✓ | | |
| 690 | CWE-120 | Does not check for buffer overflows when copying to destination | **FP:** el código asegura por medio de condicionales que los tamaños de los valores copiados al buffer no generen un overflow | ✓ | | |
| 796 | CWE-561 | The function 'bitcountCommand' is never used. [unusedFunction] void bitcountCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 799 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **FP:** Error de asignación de advertencia, línea de código no contiene operaciones con Strings. | ✓ | | |
| 801 | CWE-120 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues ( | VP: Si el valor asignado por la macro llega a ser inesperado y no se comprueba correctamente los tamaños al almacenar data en llbuf, se podría causar un overflow. | ✓ | | |
| 802 | CWE-398 | The scope of the variable 'isbit' can be reduced. [variableScope] int isbit = 0; | **VP**: El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza | | ✓ | |
| 808 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **VP:** Al no verificar que se está usando un valor de p que contenga una terminación, la función strlen(p+1) podría llevar a leer más allá del límite de p. | ✓ | | |
| 812 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **FP:** No se da buffer overflow porque el valor detectado hace referencia a la función strlen() pero el código se refiere a una variable con ese mismo nombre. | ✓ | | |
| 864 | CWE-120 | Statically-sized arrays can be improperly | **FP:** No se da buffer overflow porque el array es estático de tamaño dos y se le asignan dos | ✓ | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | restricted, leading to potential overflows or other issues. | valores *unaigned char*. Esto no es motivo de buffer overflow | | | |
| 877 | CWE-561 | The function 'bitposCommand' is never used. [unusedFunction] void bitposCommand(client *c) { ^ | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 880 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **FP:** No se da buffer overflow porque el valor detectado hace referencia a la función strlen() pero el código se refiere a una variable con ese mismo nombre. | ✓ | | |
| 882 | CWE-120 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues ( | VP: Si el valor asignado por la macro llega a ser inesperado y no se comprueba correctamente los tamaños al almacenar data en llbuf, se podría causar un overflow. | ✓ | | |
| 903 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **VP:** Al asignar p no se está asegurando que contenga una terminación \0, con lo que al usar la variable se podría leer más allá de su tamaño si no se maneja apropiadamente. | ✓ | | |
| 907 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **FN:** Se refiere únicamente a asignar un tamaño a otra variable, pero si totlen se fuera a usar para leer buffers sin chequear el tamaño dado podría ser posible que se dé un overflow, aunque no es probable. | ✓ | | |
| 1120 | CWE-120 | Statically-sized arrays can be improperly restricted, leading to potential overflows or other issues ( | **FP**: No tiene la posibilidad de suceder un buffer overflow, pues la función solo envía un String que es un literal a definido. | ✓ | | |
| 1137 | CWE-398 | Variable 'thisop' can be declared as pointer to const [constVariablePointer] struct bitfieldOp *thisop = ops+j; | **VP:** El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza. | | ✓ | |
| 1217 | CWE-120 | Statically-sized arrays can be improperly | **FP**: La advertencia hace referencia a la declaración de | ✓ | | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | restricted, leading to potential overflows or other issues ( | un array de elementos tipo *unsigned*. Esta declaración no representa ninguna debilidad por sí sola. | | | |
| 1223 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **VP:** Al llamar a la función getObjectReadOnlyString se podría dar un overflow si el parámetro aportado como llbuf tiene un largo que no coincide con el parámetro de la función que indica su tamaño con lo que se podría leer más allá del stringñ | ✓ | | |
| 1233 | CWE-126 | Does not handle strings that are not \0-terminated; if given one it may perform an over-read (it could cause a crash if unprotected) | **VP**: Un *over-read* puede ocurrir si se da el caso de que strlen sea menor que el tamaño de src. | ✓ | | |
| 1259 | CWE-561 | The function 'bitfieldCommand' is never used. [unusedFunction] void bitfieldCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1263 | CWE-561 | The function 'bitfieldroCommand' is never used. [unusedFunction] void bitfieldroCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |

**t_set.c (1245 LoC)**

| Línea | CWE | Descripción breve | Justificación | Flawfinder | Cppcheck | Manual |
|---|---|---|---|---|---|---|

| | | | | | | |
|---|---|---|---|---|---|---|
| 125 | CWE-824 | Access of Uninitialized Pointer | **FN:** No se inicializan explícitamente los fields la estructura setTtpeIterator alocada. Si el encoding no es OBJ_ENCODING_HT o OBJT_ENCODING_INTSET entonces si->di y si->ii no se inicializan, lo que podría suponer que se realicen accesos a campos no inicializados | | | ✓ |
| 247 | CWE-398 | The scope of the variable 'si' can be reduced. [variableScope]<br>setTypeIterator *si; | **VP**: El valor encontrado en la advertencia podría ser reducido en scope a el loop donde se utiliza | | ✓ | |
| 258 | CWE-120 | memcpy: Does not check for buffer overflows when copying to destination | **VP**: El código debe asumir que insetBlobLen(is) retorne el tamaño de los datos en is, si se manipula este valor es posible explotar un error de overflow. | ✓ | | |
| 272 | CWE-561 | The function 'setTypeDup' is never used. [unusedFunction]<br>robj *setTypeDup(robj *o) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 290 | CWE-665 | Variable 'd' can be declared as pointer to const [constVariablePointer]<br>dict *d = o->ptr; | **VP**: Es una buena práctica inicializar el puntero a const para evitar que pueda ocurrir modificaciones al puntero en compilación que puedan ser peligrosas. | | ✓ | |
| 303 | CWE-561 | The function 'saddCommand' is never used. [unusedFunction]<br>void saddCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | | |
| 326 | CWE-561 | The function 'sremCommand' is never used. [unusedFunction]<br>void sremCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 354 | CWE-561 | The function 'smoveCommand' is never used. [unusedFunction]<br>void smoveCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 393 | CWE-476 | NULL Pointer Dereference | **FN:** Si sucede que setTypeCreate o dbAdd fallen por algún motivo (fallo de malloc) y no se manejan adecuadamente estos errores, algunas operaciones de dstset podrían tratar de desreferenciar punteros nulos. | | | ✓ |
| 409 | CWE-561 | The function 'sismemberCommand' is never used. [unusedFunction]<br>void sismemberCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | prevención de la ejecución de código muerto o no. | | | |
| 421 | CWE-561 | The function 'smismemberCommand' is never used. [unusedFunction] void smismemberCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 440 | CWE-561 | style: The function 'scardCommand' is never used. [unusedFunction] void scardCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 598 | CWE-561 | The function 'spopCommand' is never used. [unusedFunction] void spopCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 740 | CWE-770 | Allocation of Resources Without Limits or Throttling | **FN:** Los condicionales deben del valor count de entrada del usuario, sin especificar un máximo de elementos que puede ser procesado, lo que podría explotarse para causar uso excesivo de memoria. | | | ✓ |

| | | | | | | |
|---|---|---|---|---|---|---|
| 811 | CWE-561 | The function 'srandmemberCommand' is never used. [unusedFunction]<br>void srandmemberCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 846 | CWE-665 | Variable 'o1' can be declared as pointer to const [constVariablePointer]<br>robj *o1 = *(robj**)s1, *o2 = *(robj**)s2; | **VP**: Es una buena práctica inicializar el puntero a const para evitar que pueda ocurrir modificaciones al puntero en compilación que puedan ser peligrosas. | | ✓ | |
| 846 | CWE-665 | Variable 'o2' can be declared as pointer to const [constVariablePointer]<br>robj *o1 = *(robj**)s1, *o2 = *(robj**)s2; | **VP**: Es una buena práctica inicializar el puntero a const para evitar que pueda ocurrir modificaciones al puntero en compilación que puedan ser peligrosas. | | ✓ | |
| 1011 | CWE-561 | The function 'sinterCommand' is never used. [unusedFunction]<br>void sinterCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1016 | CWE-561 | The function 'sinterCardCommand' is never used. [unusedFunction]<br>void sinterCardCommand(client *c) {<br>^ | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | ejecución de código muerto o no. | | | |
| 1048 | CWE-561 | The function 'sinterstoreCommand' is never used. [unusedFunction]<br>void sinterstoreCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1218 | CWE-561 | The function 'sunionCommand' is never used. [unusedFunction]<br>void sunionCommand(client *c) {<br>^ | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1223 | CWE-561 | The function 'sunionstoreCommand' is never used. [unusedFunction]<br>void sunionstoreCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1228 | CWE-561 | The function 'sdiffCommand' is never used. [unusedFunction]<br>void sdiffCommand(client *c) { | **VP:** En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si | | ✓ | |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | realmente existe la prevención de la ejecución de código muerto o no. | | | |
| 1233 | CWE-561 | The function 'sdiffstoreCommand' is never used. [unusedFunction]<br>void sdiffstoreCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |
| 1237 | CWE-561 | The function 'sscanCommand' is never used. [unusedFunction]<br>void sscanCommand(client *c) { | **VP**: En el contexto dado, se podría dar que un atacante mantenga acceso a este archivo y logre saltar en ejecución a la función sin que esta se use normalmente en el código, se declara como verdadero positivo pues el alcance del proyecto no permite determinar si realmente existe la prevención de la ejecución de código muerto o no. | | ✓ | |

## Reflexión:

Gracias al análisis realizado por medio de las herramientas solicitadas, se logró obtener un conocimiento mucho más amplio de las posibles debilidades que se encuentran en el código de programas de gran tamaño. En el proceso de la configuración se tuvo que explorar y buscar entre las anotaciones y archivos necesarios que se solicitaban con las directivas de preprocesamiento *include*. A pesar de resultar como un limitante, puesto que la utilización de muchas de estas en algunos archivos llevaba a tener que descartarlos por la complejidad de solucionar errores en la ejecución de los análisis, esto también resultó enriquecedor al enseñar cómo funcionan las dependencias de códigos.

Otra limitación encontrada fue el nivel de complejidad de la configuración de ciertas herramientas como *splint* y *frama*-c, las cuales requerían de un alto grado de conocimiento para configurarlas adecuadamente. Desafortunadamente esto llevó a tener que limitar el alcance previsto a solo las herramientas *cppcheck* y *flawfinder*. En cuanto a estas últimas, sus limitaciones se encontraban en el análisis dinámico del código, así como el nivel de customización para análisis exhaustivo.

Sin embargo, fueron muchas sus ventajas, pues gracias a su uso se lograron detectar errores en puntos clave como el uso de memoria y buffers, o en la declaración de código muerto e incluso de uso de punteros y correcto manejo de tipados y *casting*. Además, fayudan como primer acercamiento para decidir y contextualizar en el análisis manual de código *a posteriori*

En conclusión, se puede decir que las herramientas utilizadas presentan una gran ventaja en el reconocimiento y prevención de la producción de código susceptible a ataques maliciosos. Gracias a estas se pueden identificar, clasificar y priorizar los cambios necesarios para mitigar muchos de los riesgos asociados a la creación del software. Sin embargo, se debe tener cuidado, pues estas herramientas no son perfectas y pueden tener dificultad en realizar análisis adecuados si no se tiene buen conocimiento de como configurarlas. Incluso, es sumamente necesario el realizar un post-proceso de análisis manual para determinar la veracidad de los resultados y atrapar posibles detalles que no hayan sido identificados por las mismas.

## Recomendaciones:

1. Utilizar los recursos de documentación aportados en las páginas web oficiales de cada una de las herramientas usadas.
2. Investigar independientemente los diferentes CWE que existen, especialmente los más comunes pues hay una gran cantidad de los mismos, con el fin de poder facilitar el posterior análisis manual de los resultados del uso de las herramientas.
3. Leer detenidamente las anotaciones proporcionadas por los desarrolladores en el código fuente, ya que estas ayudan a comprender con mayor claridad la intención de las funciones y el programa en general.
4. Realizar varias pruebas con parámetros distintos de tal modo que se compruebe a criterio propio y análisis cuales son los más indicados según el esfuerzo requerido.
5. Realizar un análisis manual posterior al uso de las herramientas para comprobar la veracidad y completitud de los análisis realizados, esto incluso puede conllevar comparar los resultados de diferentes herramientas.

## Anexos

**Código del script scriptFindFilesOnSize.sh**

```
# Directorio de inicio para la búsqueda
START_DIR="/home/kali/redis-7.0.15"

# Buscar archivos con extensión .c
#find "$START_DIR" -type f \( -name "*.c" -o -name "*.cpp" -o -name "*.h" \) -exec wc -l {} + | awk '{if ($1 > 500) print }'
find "$START_DIR" -type f \( -name "*.c"  \) -exec wc -l {} + | awk '{if ($1 > 1000) print } '
#find "$START_DIR" -type f \( -name "*.c" -o -name "*.cpp"  \) -exec wc -l {} + | awk '{if ($1 > 500) print }'
```

```
#find "$START_DIR" -type d -path "*test*" -prune -o \( -type f \(  -name "*.c" -o -name "*.cpp" -o -name "*.h" \) \) -exec wc -l {} + | awk '{if ($1 > 500) print }'
```

**Código del script filterFilesOnIncludes.sh**

```
#!/bin/bash
while read -r line; do
    file_path=$(echo "$line" | awk '{print $2}')
    include_count=$(grep -c '^\#\s*include' "$file_path")
    if [ "$include_count" -le 3 ]; then
        echo "El archivo $file_path utiliza $include_count #include(s)."
    else
        echo "sin suerte."
    fi
    echo
done
```