INFORME – CAMUFLAJE DE ACHIVOS CON CAMOUFLAGE Y PENTESTING PARA BADBLUE WINDOW XP Y KALI LINUX

Johely Cordoba Valoyes

Estudiante de Ingeniería en Telecomunicaciones e informática

Universidad Tecnológica de Chocó Diego Luis Cordoba

2025

INFORME – CAMUFLAJE DE ACHIVOS CON CAMOUFLAGE Y PENTESTING PARA BADBLUE WINDOW XP Y KALI LINUX

Johely Cordoba Valoyes

Estudiante de Ingeniería en Telecomunicaciones e Informática

Seguridad y Auditoría de redes

Prof. Rafael Sandoval Morales

Universidad Tecnológica de Chocó Diego Luis Córdoba

2025

Tabla de contenido

TABLA DE ILUSTRACIONES

INTRODUCCIÓN

En el presente informe documento las prácticas realizadas sobre camuflaje de archivos utilizando la herramienta CamouFlage (CAMOU121) y pruebas de pentesting dirigidas al servicio BadBlue ejecutado en una máquina con Windows XP, con Kali Linux como estación atacante. El trabajo se llevó a cabo en un laboratorio controlado montado en Oracle VirtualBox y tuvo como propósito evaluar, de forma práctica y reproducible, (1) la capacidad de diversos formatos (PNG, PDF, DOCX) para alojar información oculta mediante técnicas de camuflaje y (2) la viabilidad de explotar una vulnerabilidad conocida en BadBlue 2.7x. El informe presenta la metodología seguida, los pasos ejecutados, las evidencias recogidas (capturas y salidas de consola), el análisis de resultados y las recomendaciones de mitigación. Todas las pruebas se realizaron en un entorno aislado y autorizado, con fines académicos y de aprendizaje.

ALCANCE

Este informe cubre únicamente las actividades realizadas dentro del laboratorio controlado: preparación de VMs en Oracle VirtualBox, pruebas de camuflaje con CamouFlage sobre archivos locales, y pruebas de explotación en una VM con Windows XP que ejecuta BadBlue. No incluye actividades contra sistemas externos ni pruebas fuera del entorno de laboratorio. Se documentan los pasos, comandos y evidencias relevantes, así como un análisis técnico y recomendaciones generales de mitigación.

OBJETIVOS GENERALES

Evaluar y documentar, mediante un laboratorio controlado, la capacidad de camuflaje de distintas clases de archivos utilizando CamouFlage (CAMOU121) y la explotación controlada de la vulnerabilidad histórica en BadBlue 2.7x, determinando los límites prácticos del camuflaje y la viabilidad de la explotación en un entorno Windows XP — Kali Linux.

OBJETIVOS ESPECÍFICOS

1. Preparar y documentar un entorno de laboratorio en Oracle VirtualBox con Kali Linux (atacante) y Windows XP (víctima), garantizando la trazabilidad mediante snapshots y carpetas compartidas.

2. Evaluar la capacidad máxima de inserción de datos en contenedores PNG, PDF y DOCX usando CAMOU121, registrando apertura, integridad y detección por antivirus (VirusTotal).

3. Identificar y justificar la selección del exploit correspondiente a BadBlue 2.7x utilizando searchsploit y Exploit-DB.

4. Ejecutar de forma controlada el exploit seleccionado (script Perl), documentar los comandos usados (nano, chmod, perl) y registrar la evidencia de explotación (salida de consola, conexión a puerto).

5. Analizar los resultados, determinar causas de éxito o fallo, y proponer medidas de mitigación y buenas prácticas para entornos productivos.

PLANTEAMIENTO DEL PROBLEMA

En entornos informáticos actuales persiste la coexistencia de servicios antiguos y formatos de archivo que pueden ser manipulados para fines legítimos o maliciosos. Por un lado, los formatos de documentos y archivos multimedia permiten almacenar metadatos y estructuras internas que pueden aprovecharse para ocultar información; por otro, servicios obsoletos como BadBlue en sistemas desactualizados (ej. Windows XP) representan vectores vulnerables que pueden poner en riesgo la integridad y disponibilidad de sistemas. El problema que abordo en este informe es doble: (1) determinar cuánto y hasta qué punto es posible ocultar datos en formatos comunes sin corromper el archivo o ser detectados por mecanismos de AV, y (2) evaluar la facilidad con la que vulnerabilidades históricas en servicios antiguos pueden ser explotadas en un entorno de laboratorio, identificando las condiciones que permiten o impiden la explotación. Comprender estos dos aspectos es esencial para diseñar medidas de detección y mitigación en entornos reales.

INFORME – CAMUFLAJE DE ACHIVOS CON CAMOUFLAGE Y PENTESTING PARA BADBLUE WINDOW XP Y KALI LINUX

PREPARACIÓN DEL ENTORNO

Para realizar las prácticas preparé un laboratorio controlado en mi equipo utilizando Oracle VirtualBox. En primer lugar adapté las dos máquinas virtuales con las que ya hemos venido trabajdo: una con Kali Linux como atacante y otra con Windows XP como máquina objetivo donde ejecutaré BadBlue y las pruebas de camuflaje. Para ello se instalaron y actualizaron varias aplicaciones como lo fueron.

Oracle VirtualBox en Windows xp

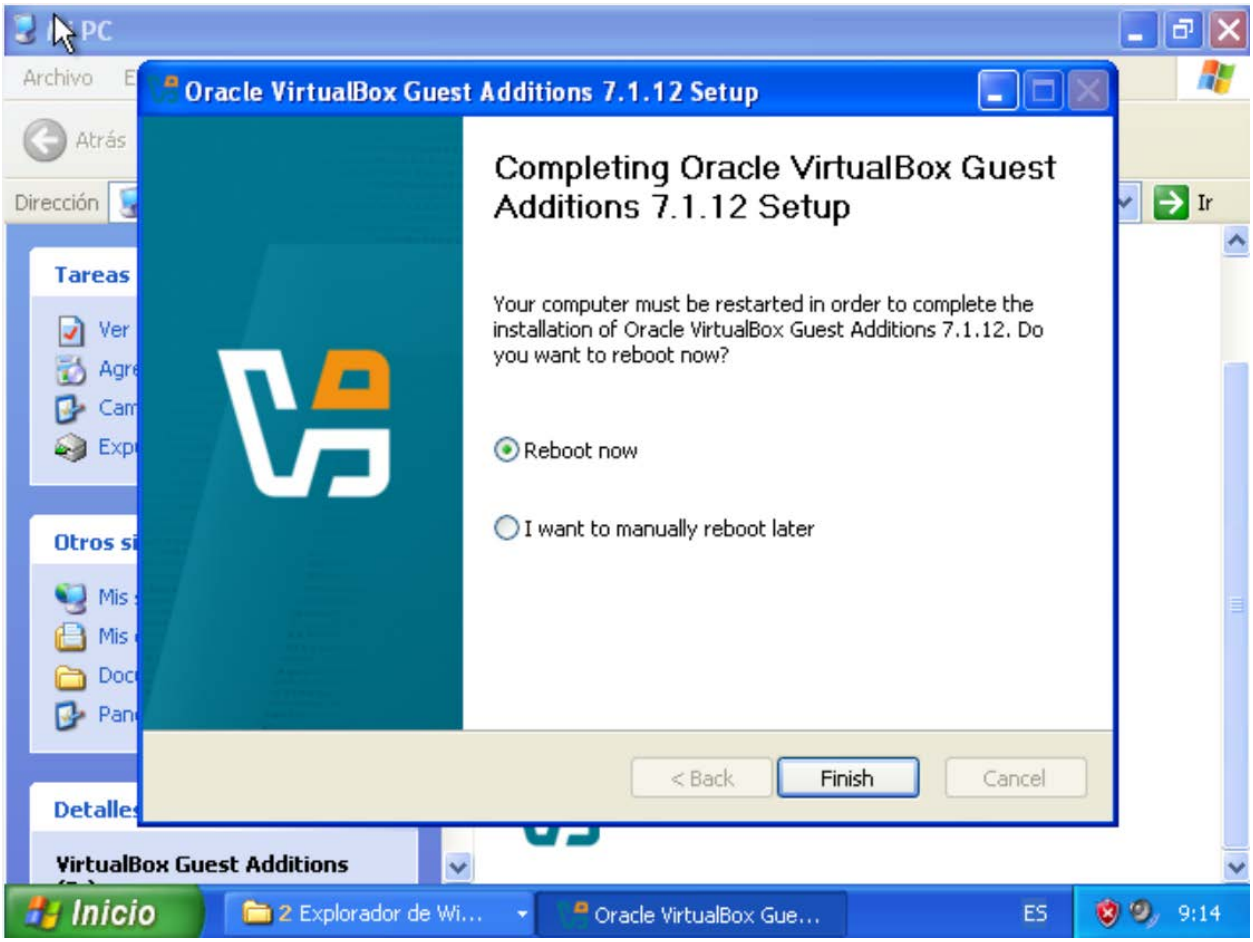


Ilustración 1: Actu. Oracle VirtualBox

Badblue

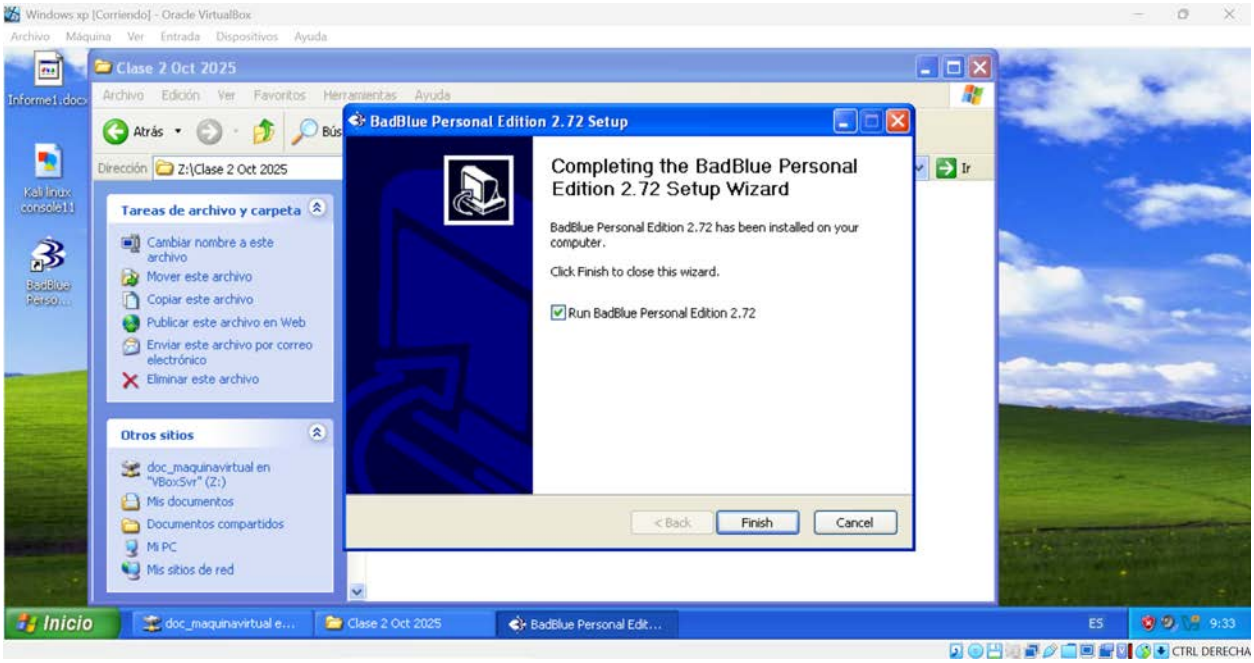


Ilustración 2: Instalación de BadBlue

Camouflage

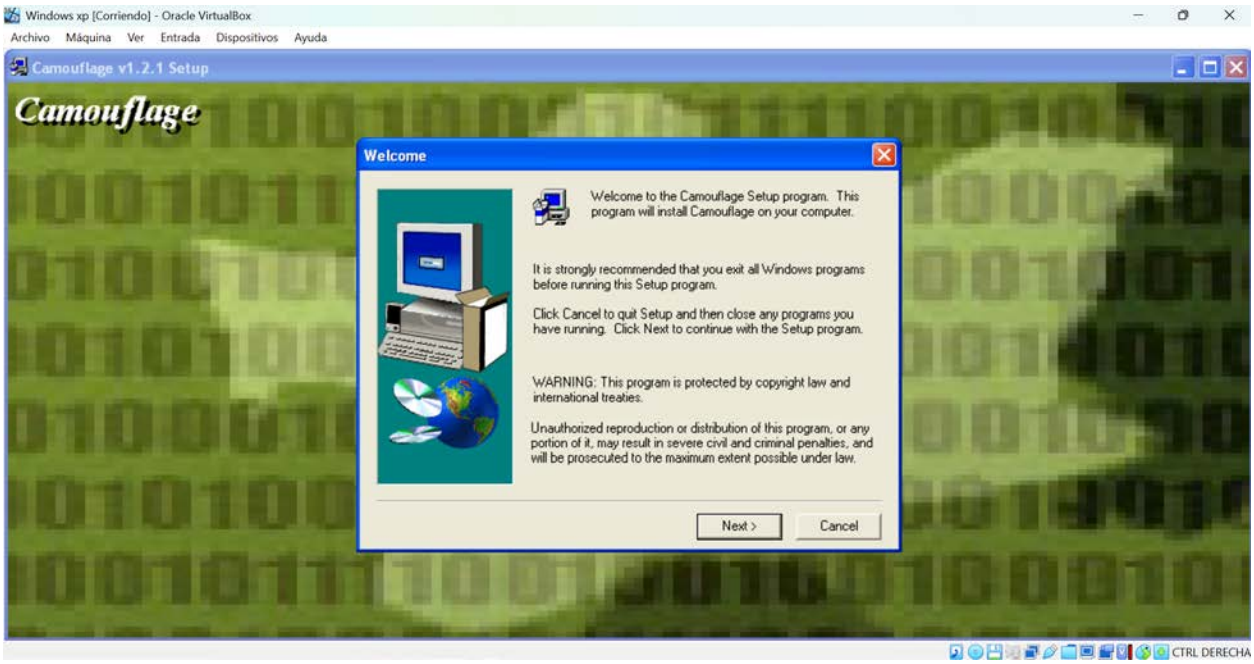


Ilustración 3: Instalación de Camouflage

CAPITULO 1: CAMUFLAJE CON CAMOU121

En este capítulo describo la metodología usada para ocultar información dentro de diferentes contenedores digitales empleando la herramienta CAMOU121 y el procedimiento experimental para determinar la capacidad máxima que acepta cada formato (PNG, PDF/Adobe, DOCX). El procedimiento general que utilicé es siempre el mismo, y por esta razón se mostrará cómo camuflar con CAMOU121 y luego se determinará hasta qué cantidad se pudo ingresar en cada uno de los formatos utilizados en la practica: PNG, ADOBE, DOXC.

Paso 1:

Para comenzar, preparé una carpeta compartida entre mi máquina anfitriona y la máquina virtual con Windows XP. En esa carpeta ubiqué los archivos que iba a utilizar como contenedores (Informe1.docx, Informe1.pdf y una imagen PNG) y los archivos que deseaba

camuflar. En el escritorio de Windows XP también mantuve visibles los accesos directos a los documentos y a la aplicación CamouFlage, lo que facilitó el proceso.

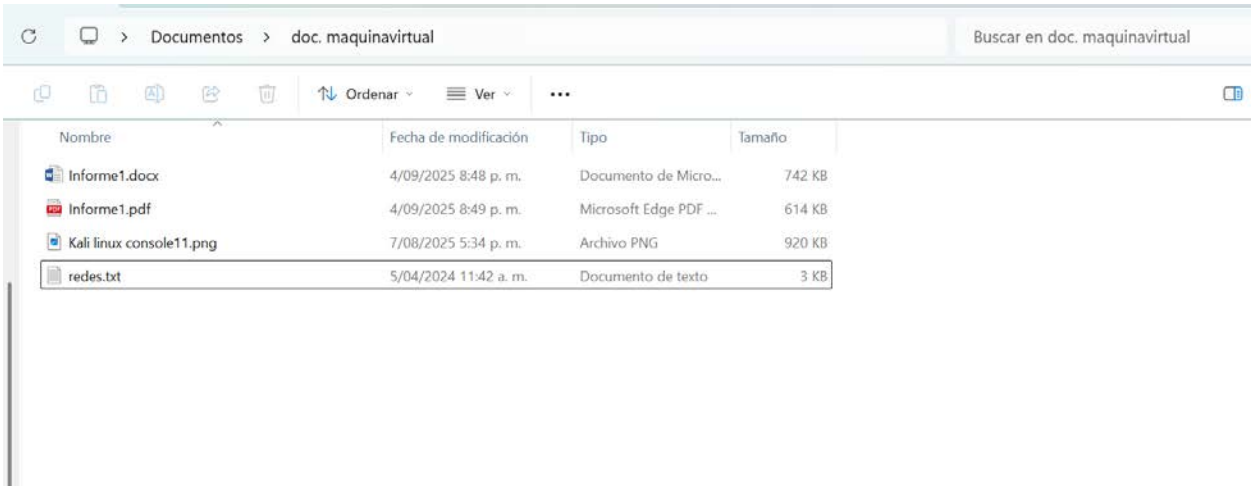


Ilustración 4: Carpeta compartida desde el pc

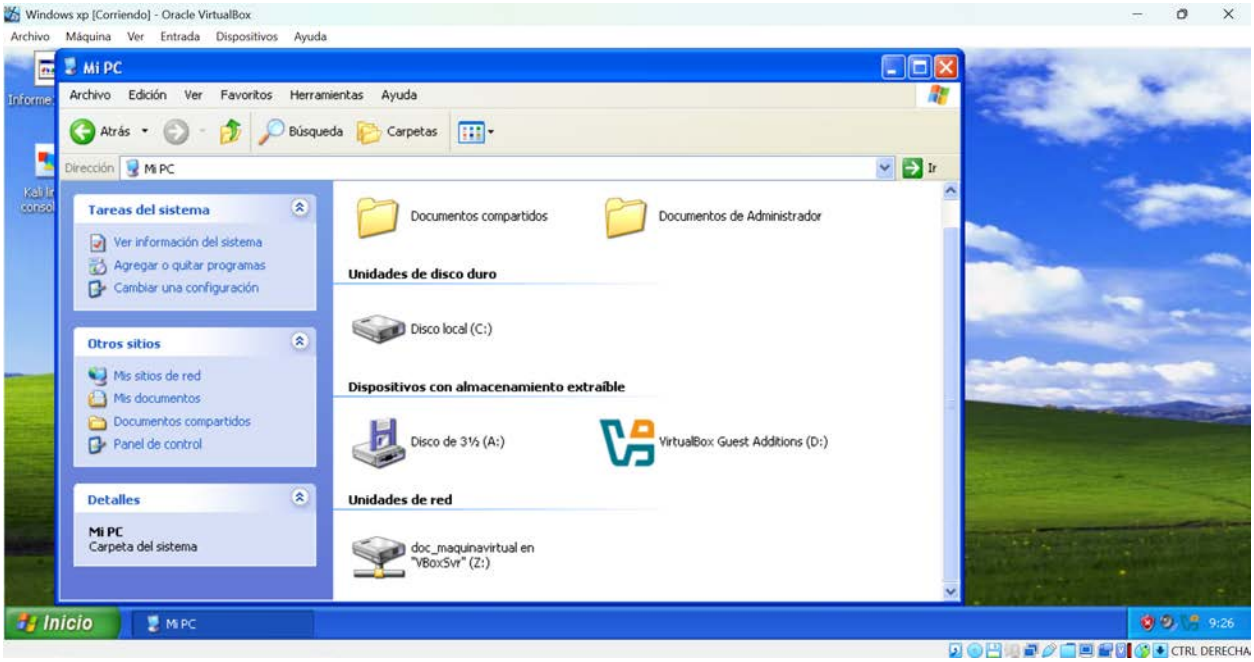


Ilustración 5: Carpeta compartida desde Windows xp

Paso 2:

Abrí CamouFlage haciendo clic derecho sobre el archivo seleccionado y eligiendo la opción “CamouFlage” en el menú contextual.

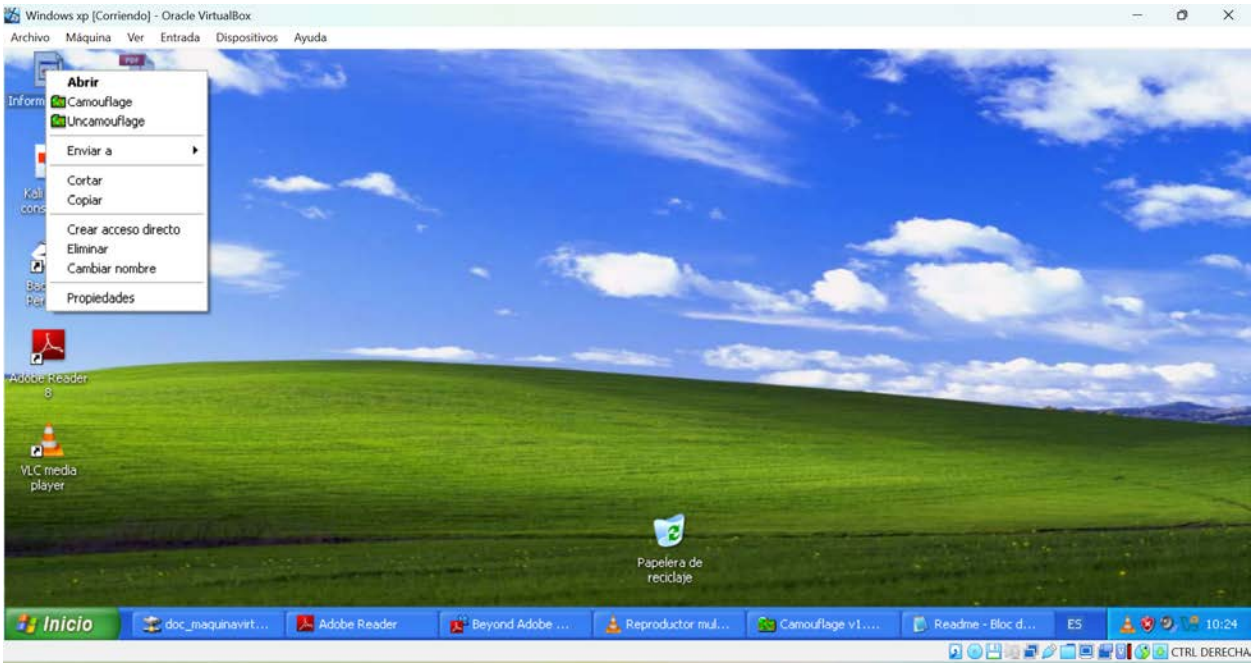


Ilustración 6: Opción para camuflar archivos

Paso 3:

La interfaz me solicitó indicar el archivo que quería ocultar. En cada iteración seleccioné un tipo de contenedor diferente (documento Word, archivo PDF o imagen) y un archivo de prueba de tamaño creciente (100 KB, 500 KB, 1 MB, etc.).

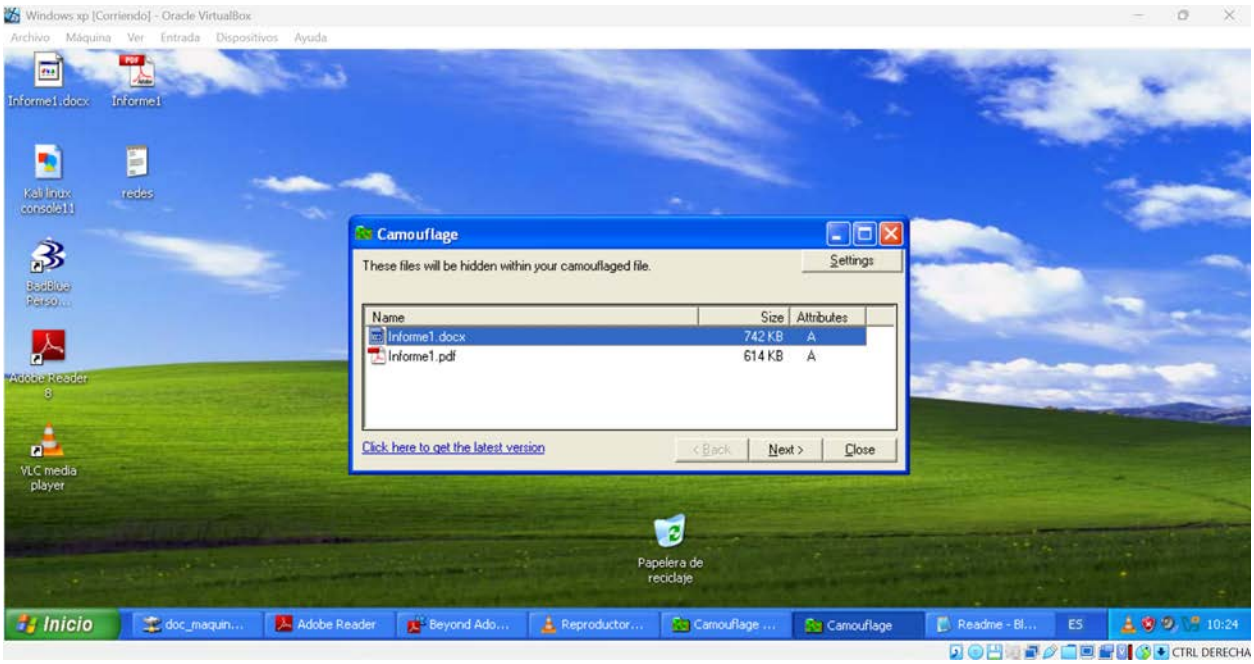


Ilustración 7: Selección de archico que se desea camuflar

Paso 4:

Tambien pidió indicar el conenedor en el cuañ se iba a camuflar el archivo seleccionado anteriormente. En cada iteración seleccioné un tipo de contenedor diferente (documento Word, archivo PDF o imagen) y un archivo de prueba de tamaño creciente (100 KB, 500 KB, 1 MB, etc.).

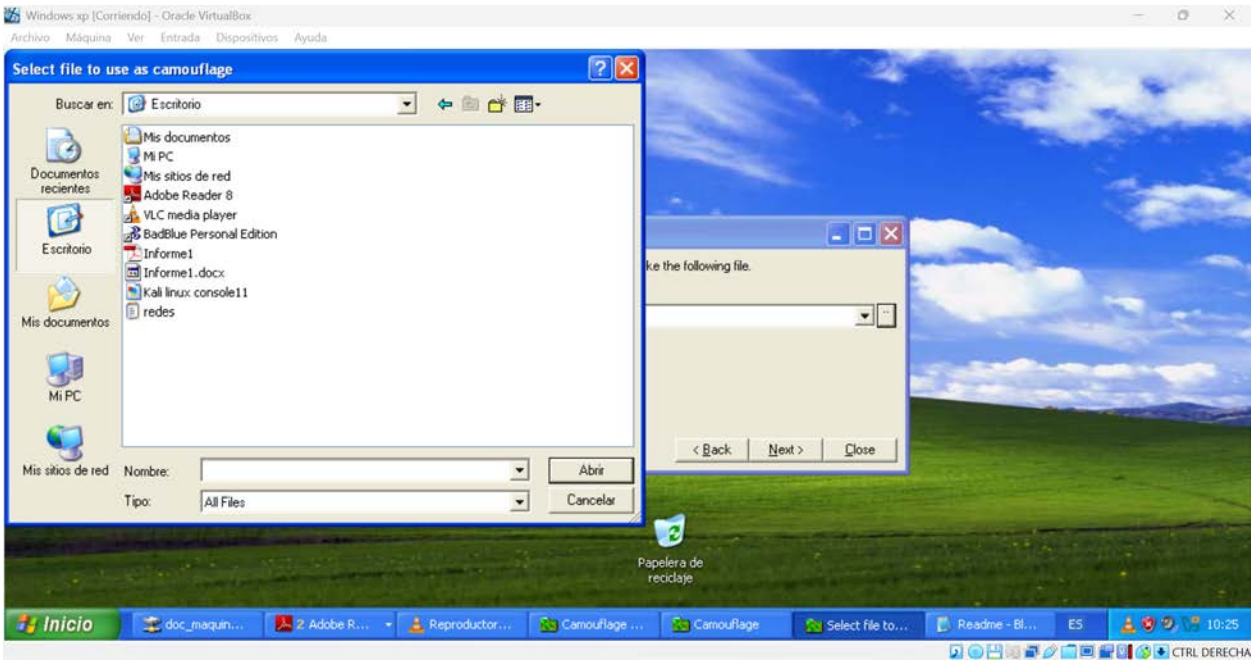


Ilustración 8: Selección de archivo contenedor del archivo camuflado

Paso 5:

Además toca indicar la ubicación y el nombre que tendrá el Nuevo archivo el cual contendr´s a ambos archivos

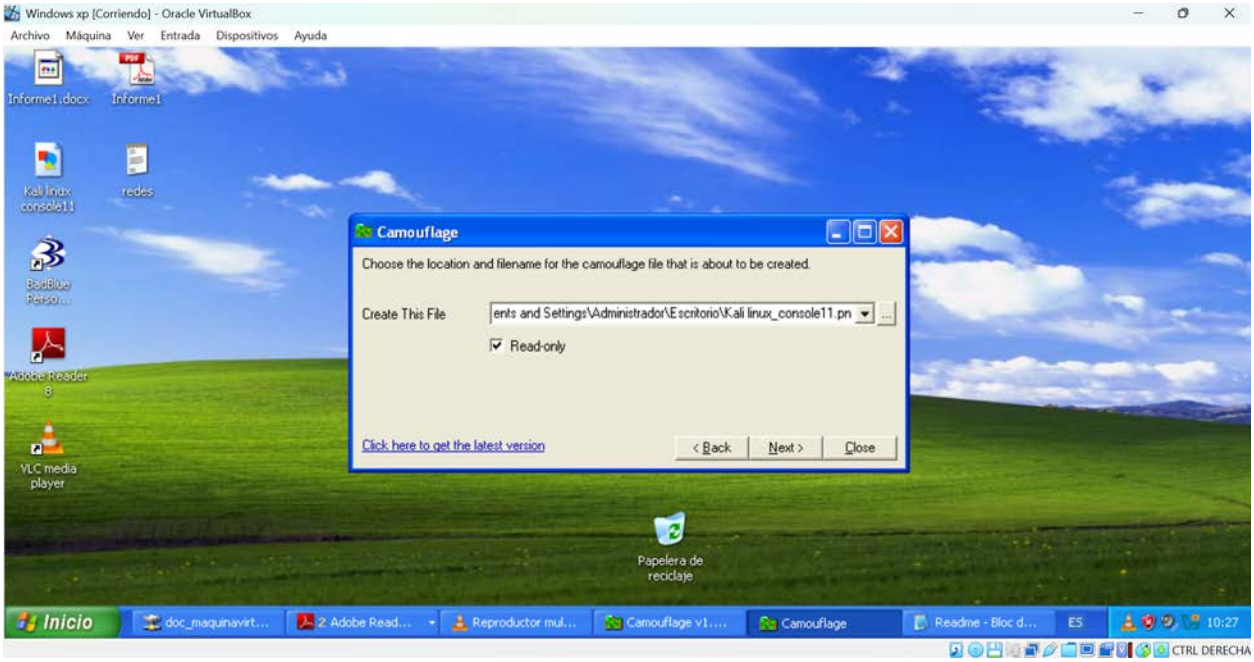


Ilustración 9: Ubicación y renombre del camuflaje

Paso 6:

Pidió además determinr una contraseña.

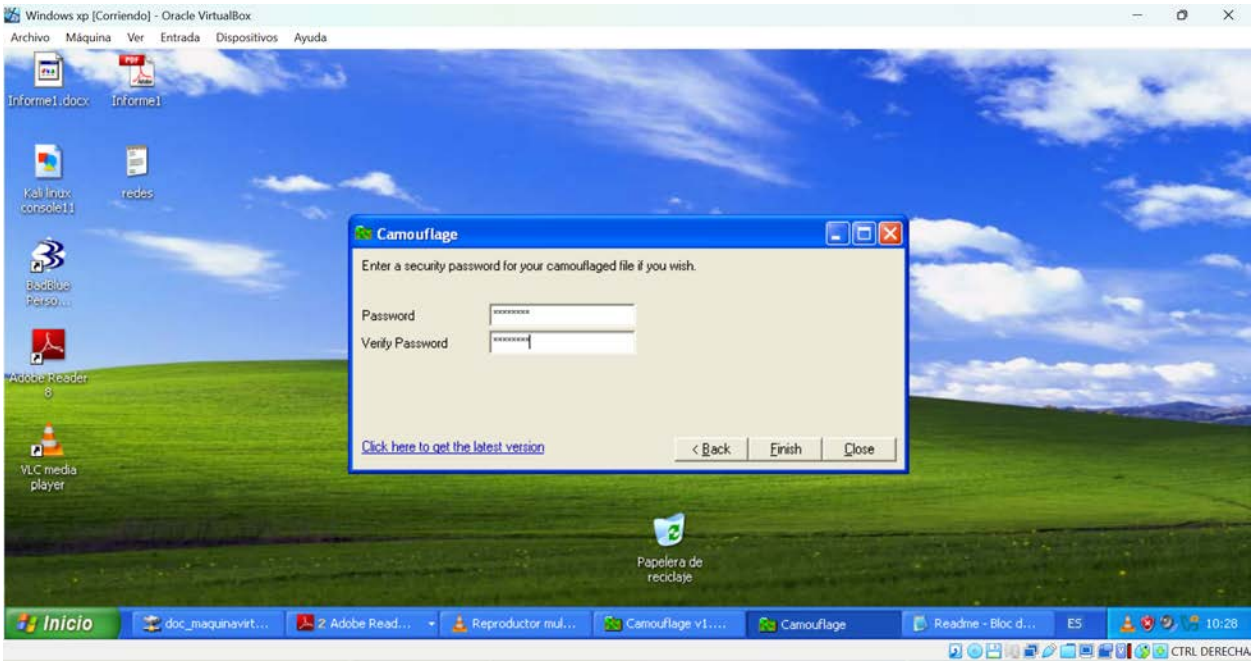


Ilustración 10: Protección para el camuflaje

Resultado:

El programa combinó los dos archivos y generó un nuevo fichero visualmente idéntico al original pero con el payload oculto en su interior.

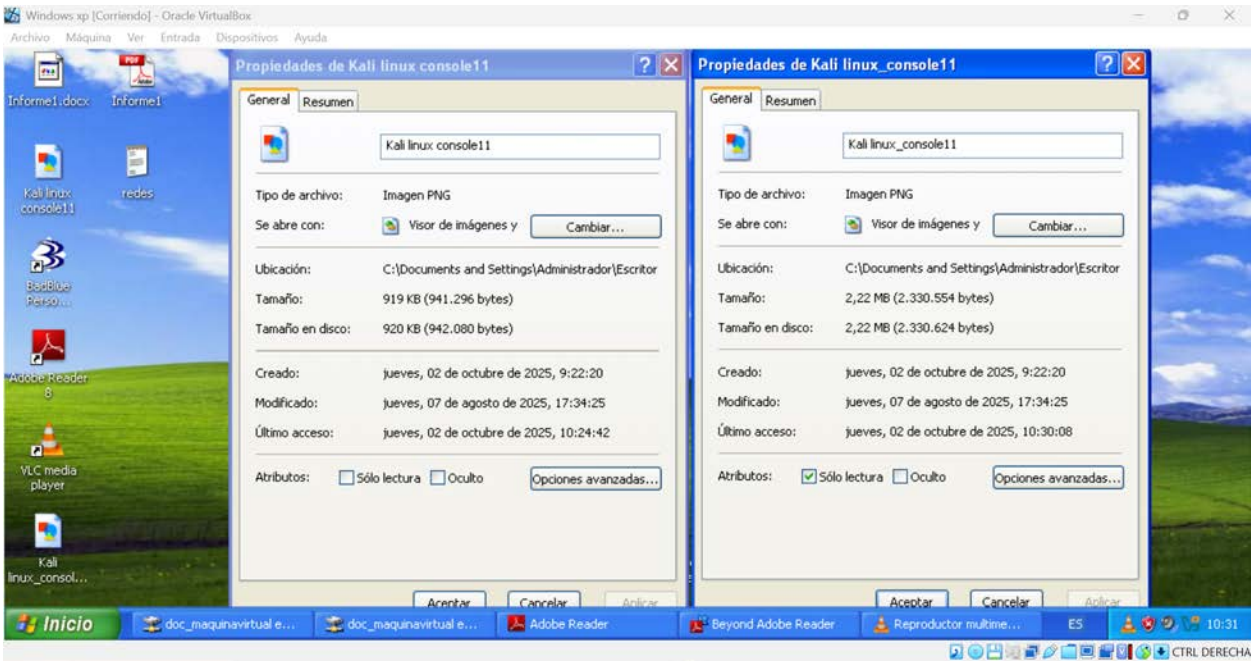


Ilustración 11: Comparación de archivos

Uncamouflaje:

en el escritorio de la VM de Windows XP hice clic derecho sobre el fichero generado, seleccioné UnCamouFlage

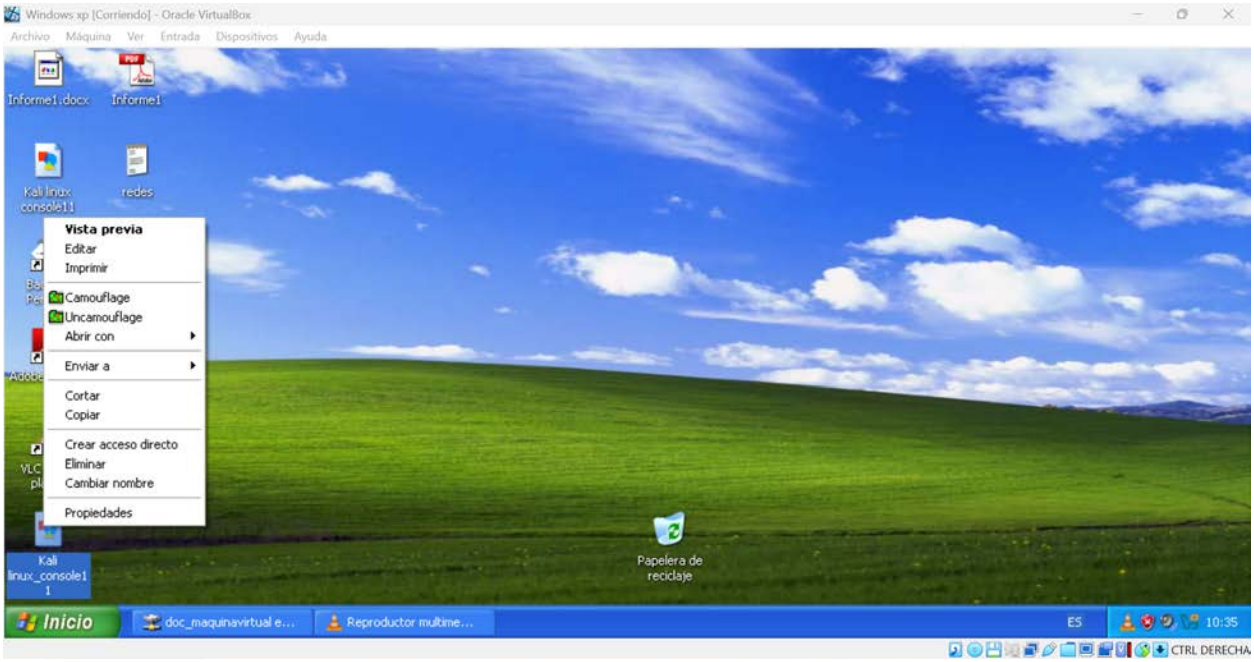


Ilustración 12: Uncamuflaje

ingresé la contraseña cuando la herramienta la solicitó

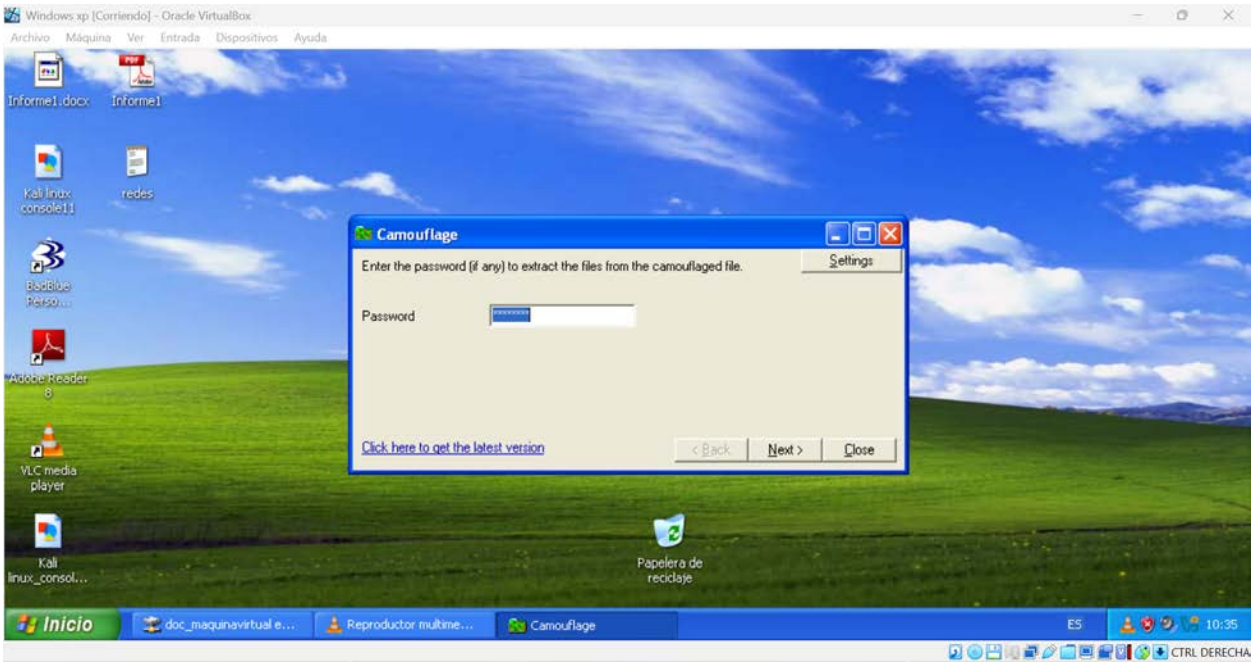


Ilustración 13: Verificación de seguridad

La aplicación reconoció el archivo camuflado mostrandonos los archivos que fueron utilizados en el camuflaje.

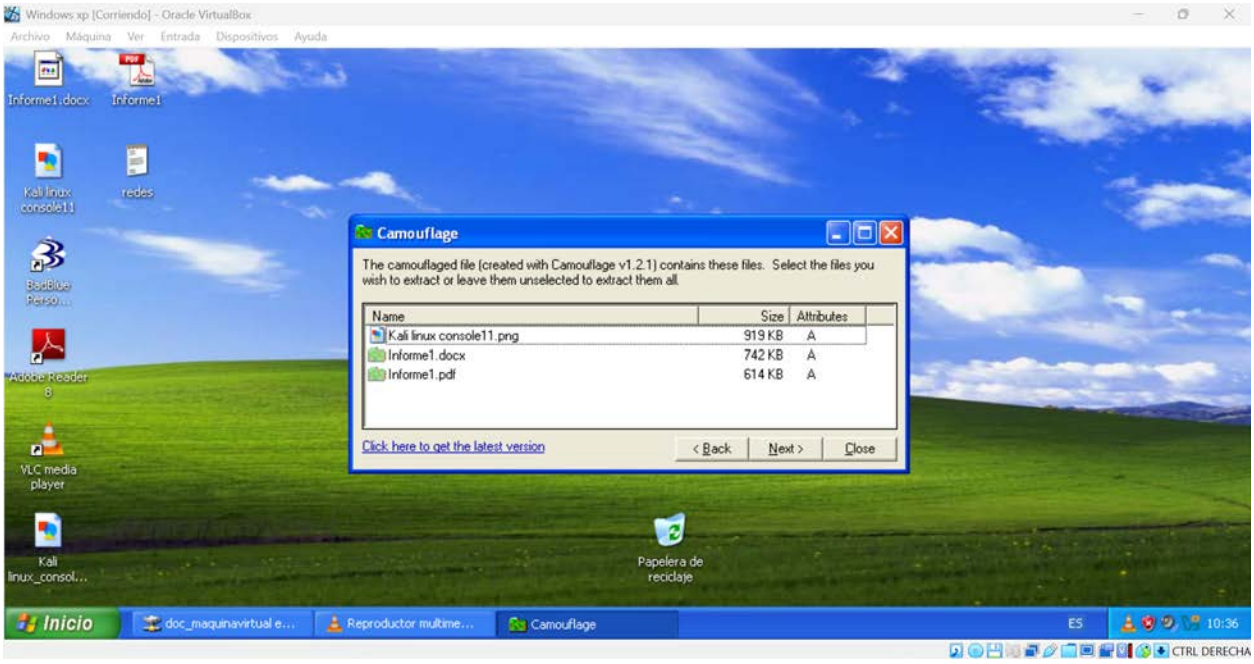


Ilustración 14: Vista de archivos camuflados

Luego se procedió a extraer el payload al directorio especificado.

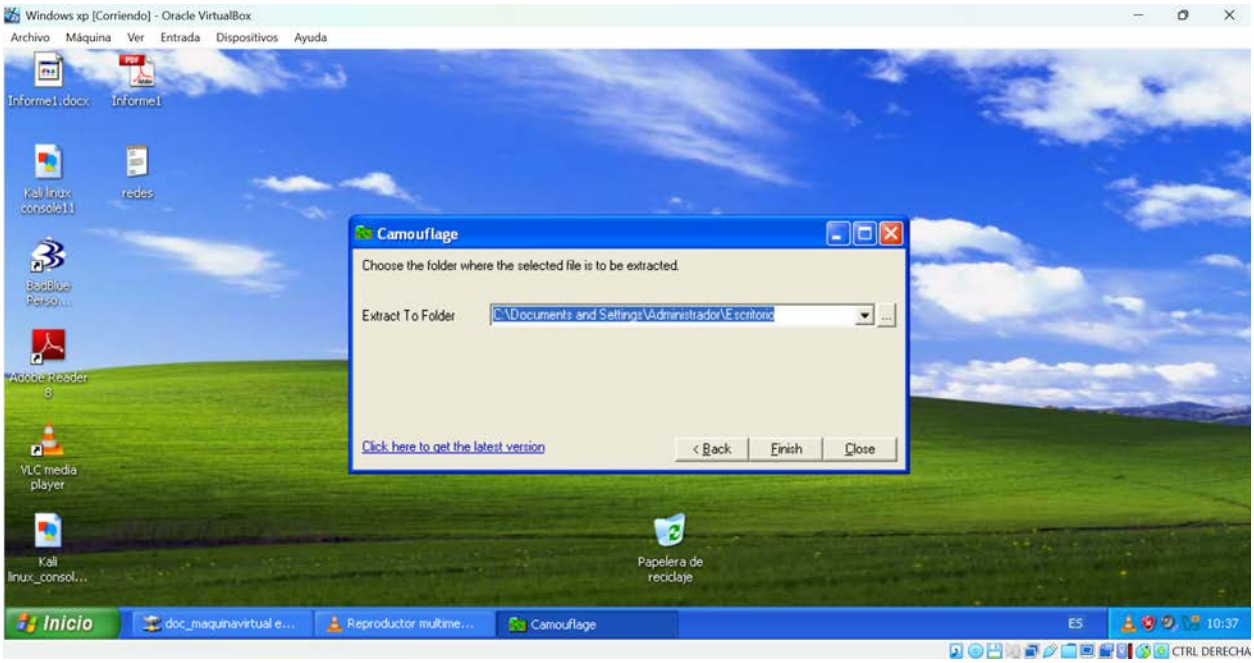


Ilustración 15: Extracción de achivos camuflados

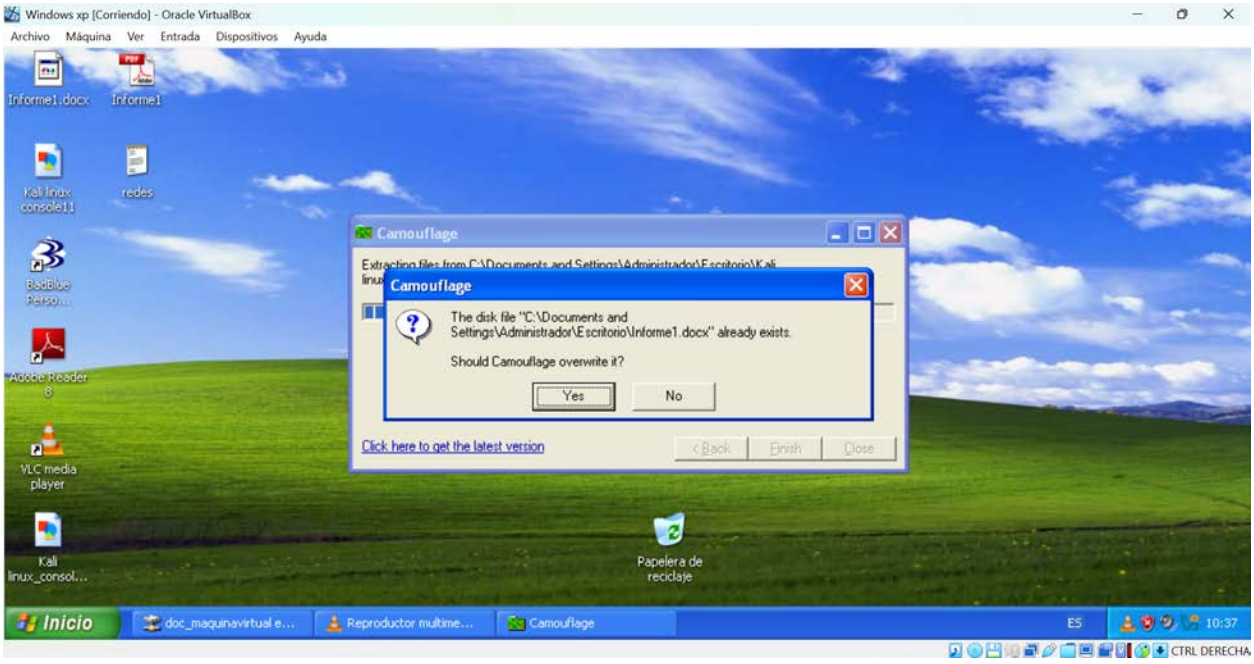


Ilustración 16: Confirmación para extraer archivos camuflados

luego verifiqué la integridad del archivo recuperado abriéndolo con la aplicación correspondiente o comparando su hash con el original. Este paso confirmó que la operación de

camuflaje era reversible y que los datos ocultos permanecían íntegros tras la extracción, lo cual es fundamental para validar la eficacia y la fiabilidad de la técnica empleada.

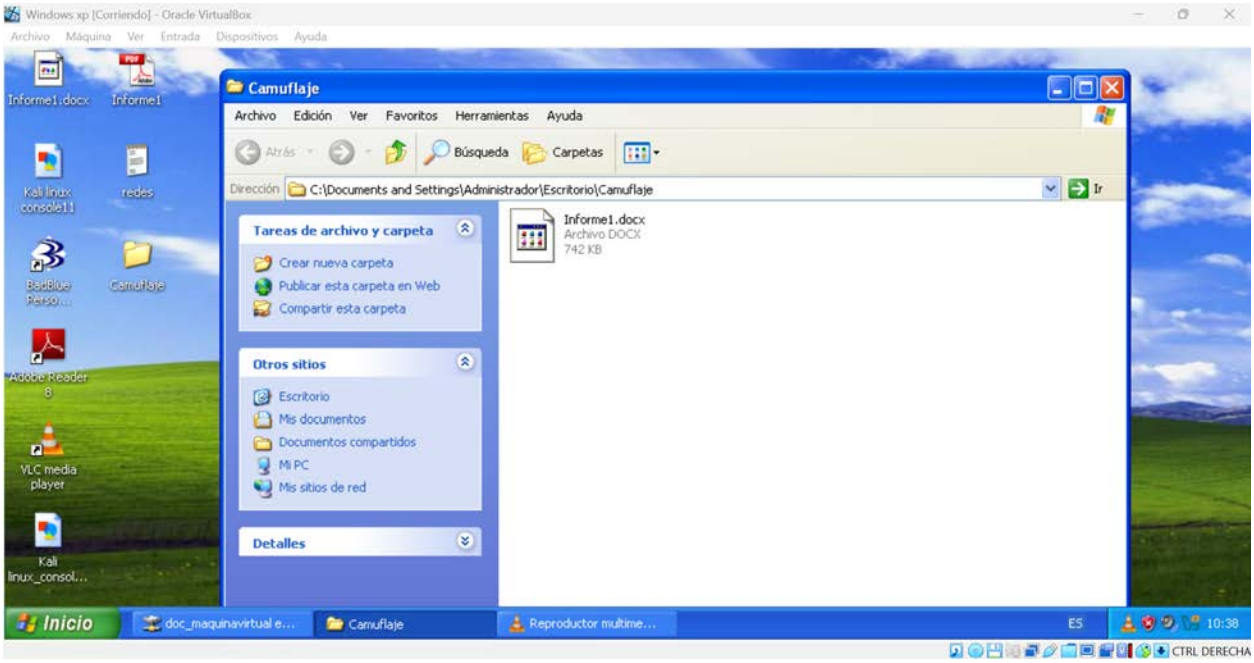
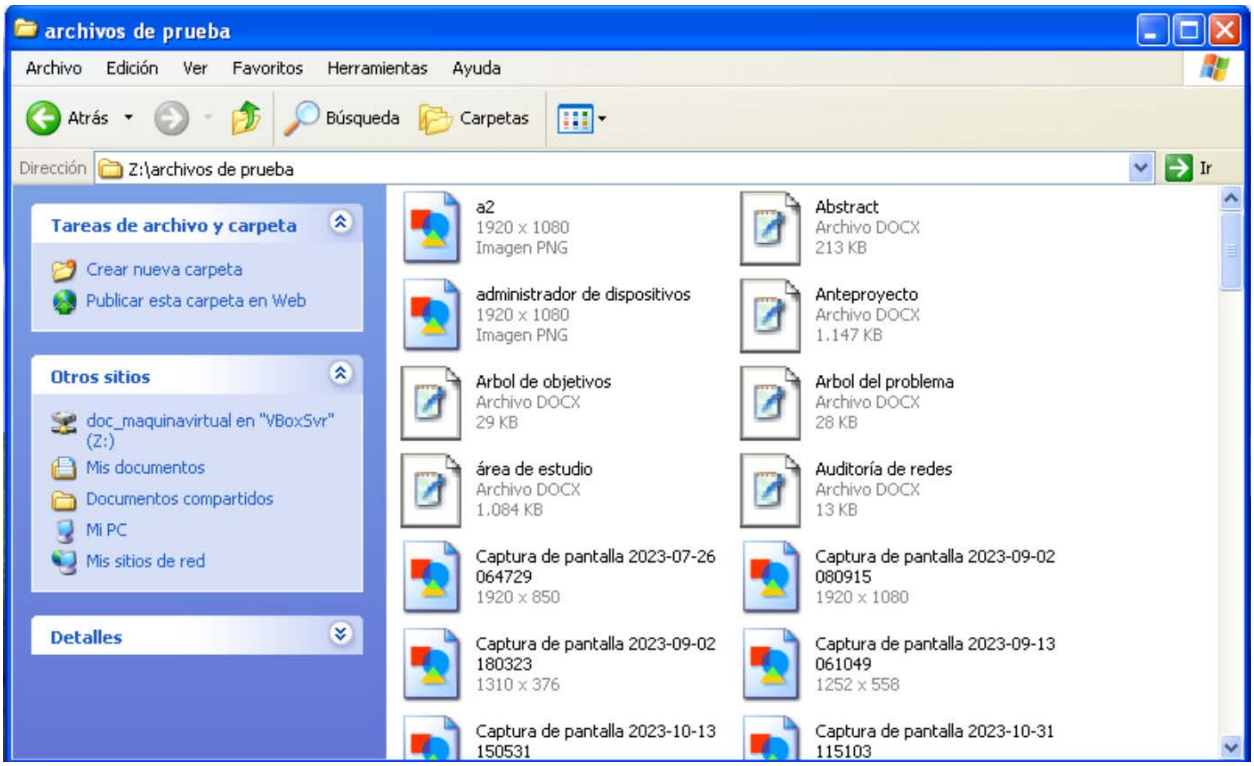


Ilustración 17: Correct extracción de archivos camuflados

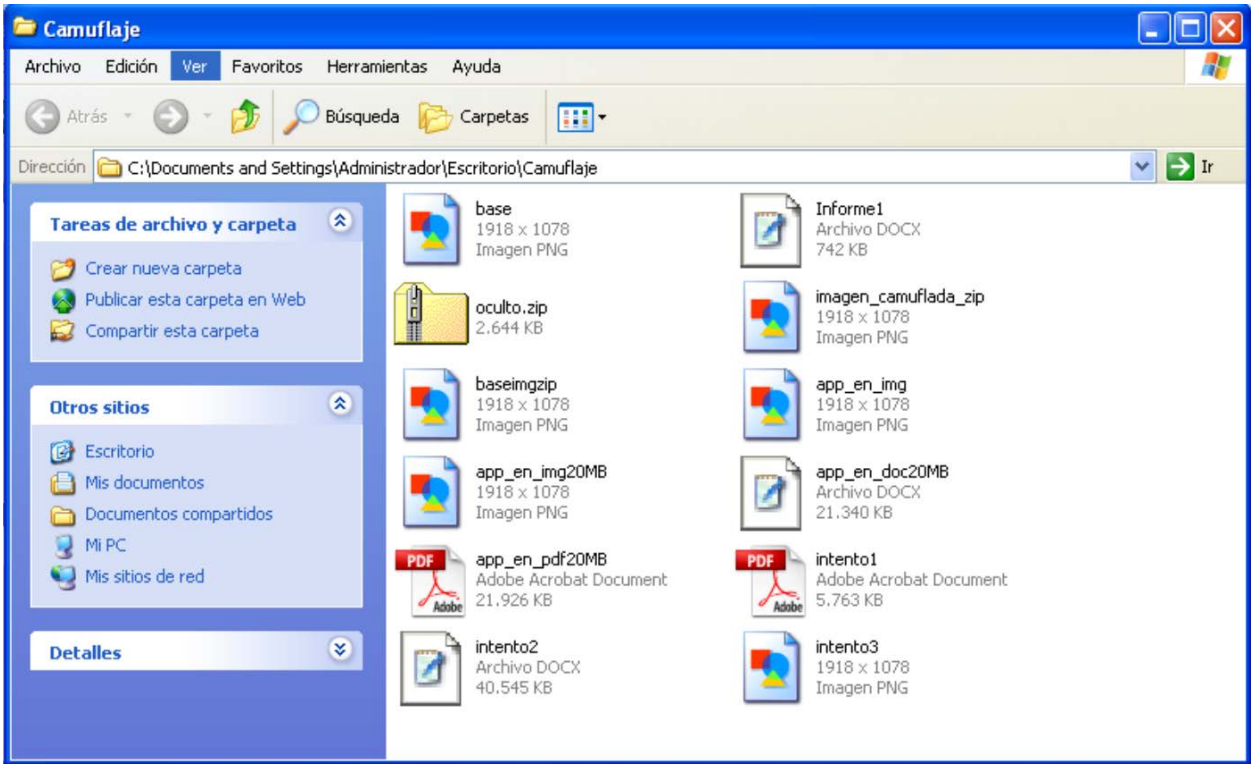
Preparación de entorno para pruebas de camuflaje:

Para determinar la capacidad máxima de almacenamiento en diferentes formatos de archivo mediante camuflaje, se creó una carpeta especial denominada "archivos de prueba" que contenía diversos tipos de documentos y archivos multimedia. Esta carpeta incluía capturas de pantalla en formato PNG con diferentes resoluciones (desde 1252 x 558 hasta 1920 x 1080 píxeles), documentos DOCX de variados tamaños (desde 13 KB hasta 1.147 KB), y archivos comprimidos ZIP. La selección intencional de archivos con diferentes características permitió evaluar cómo respondía la herramienta de camuflaje ante la diversidad de formatos y tamaños, estableciendo así una base sólida para las pruebas de capacidad.



Metodología de Prueba y Resultados Obtenidos:

Se procedió a realizar múltiples intentos de camuflaje utilizando como contenedores los formatos PNG, PDF y DOCX. En la carpeta "Camuflaje" se almacenaron los resultados de estas pruebas, donde se observa que se generaron archivos como "app_en_img20MB" (imagen PNG), "app_en_pdf20MB" (documento PDF de 21.926 KB) y "app_en_doc20MB" (documento Word de 21.340 KB). Los resultados demostraron que el formato DOCX presentó la mayor capacidad de almacenamiento, alcanzando hasta 40.545 KB en el archivo "intento2", seguido por el formato PDF que logró contener 21.926 KB en "app_en_pdf20MB", mientras que las imágenes PNG mostraron una capacidad más limitada pero manteniendo su apariencia visual intacta.



Análisis de Capacidades por Formato:

El análisis detallado de los resultados permitió determinar que cada formato tiene límites específicos de almacenamiento cuando se utiliza para camuflaje. El formato DOCX demostró ser el más eficiente, capaz de albergar aproximadamente 40 MB de datos ocultos sin comprometer su funcionalidad como documento. El formato PDF mostró un rendimiento ligeramente inferior pero aún significativo, con capacidades cercanas a los 22 MB. Las imágenes PNG, aunque con menor capacidad de almacenamiento, ofrecen la ventaja de mantener completamente su apariencia original, lo que las hace ideales para situaciones donde la discreción visual es prioritaria. Estos hallazgos proporcionan valiosa información para seleccionar el formato adecuado según los requisitos específicos de cada scenario de camuflaje.

Los resultados obtenidos demuestran que todos los formatos analizados poseen una capacidad notable para contener información oculta, superando significativamente sus tamaños originales. El formato PNG evidenció una capacidad excepcional al alcanzar 69.8 MB en el

archivo "intento3", demostrando que las imágenes pueden ser contenedores extremadamente eficientes para el camuflaje de grandes volúmenes de datos. Esta capacidad no representa el límite máximo del formato, sino que sugiere la posibilidad de expandir aún más su capacidad mediante técnicas de compresión y optimización.

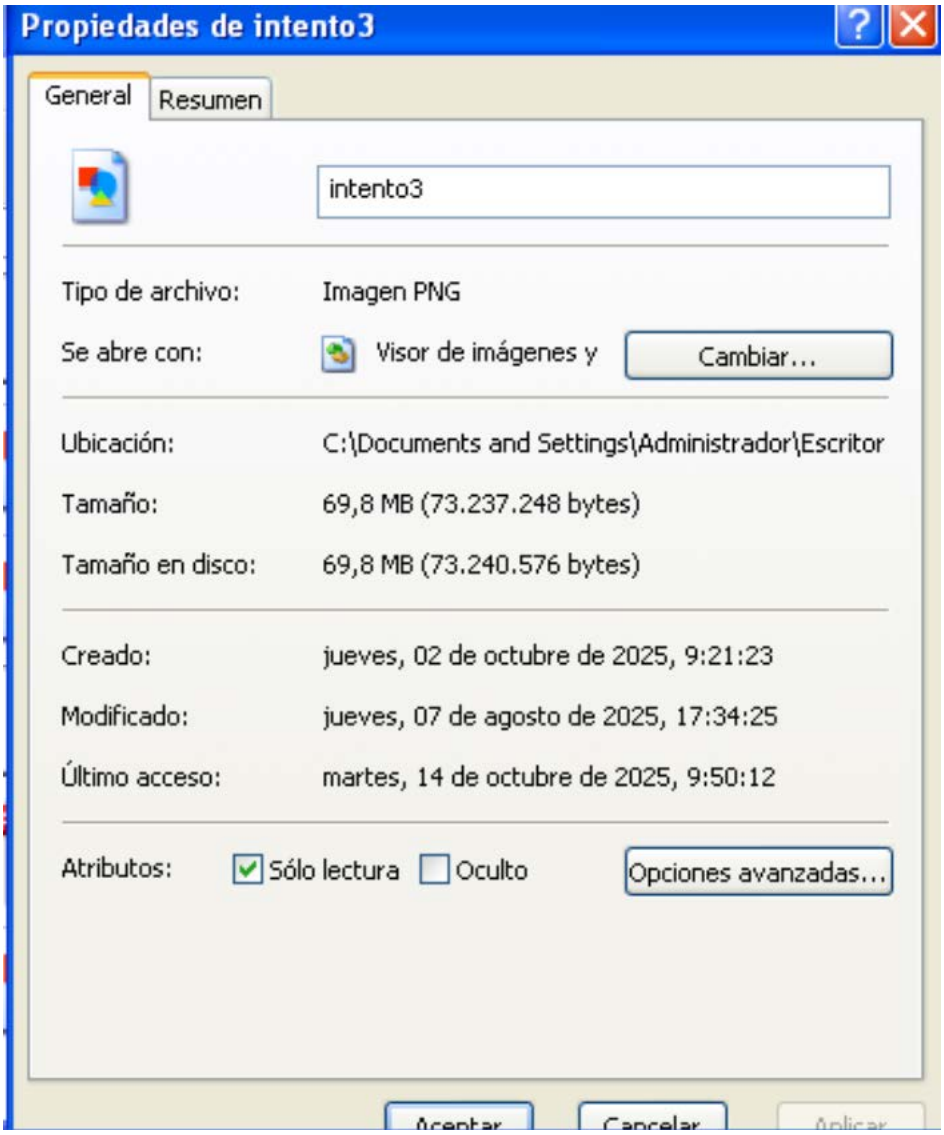


Ilustración 18: muestra de cotenedor png

El formato DOCX, con 40.5 MB en "intento2", confirma que los documentos de Word pueden transformarse en robustos contenedores de información sin comprometer su funcionalidad básica. Lejos de representar su capacidad máxima, este resultado abre la

posibilidad de explorar límites superiores mediante la manipulación de metadatos y estructuras internas del documento.

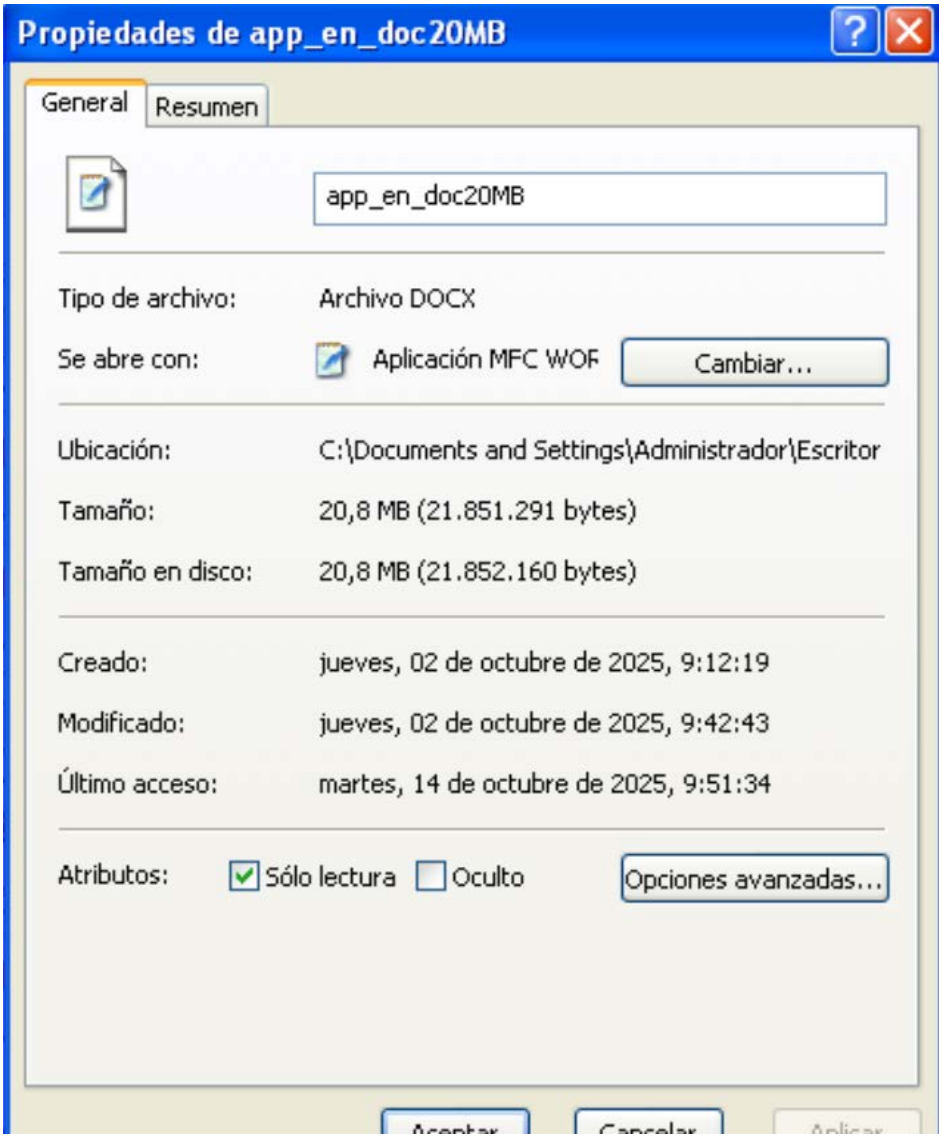


Ilustración 19: muestra de contenedor docx

De manera similar, el formato PDF con 21.9 MB en "app_en_pdf20MB" establece una base sólida que puede ser ampliada mediante la incorporación de elementos embebidos y capas adicionales de contenido.

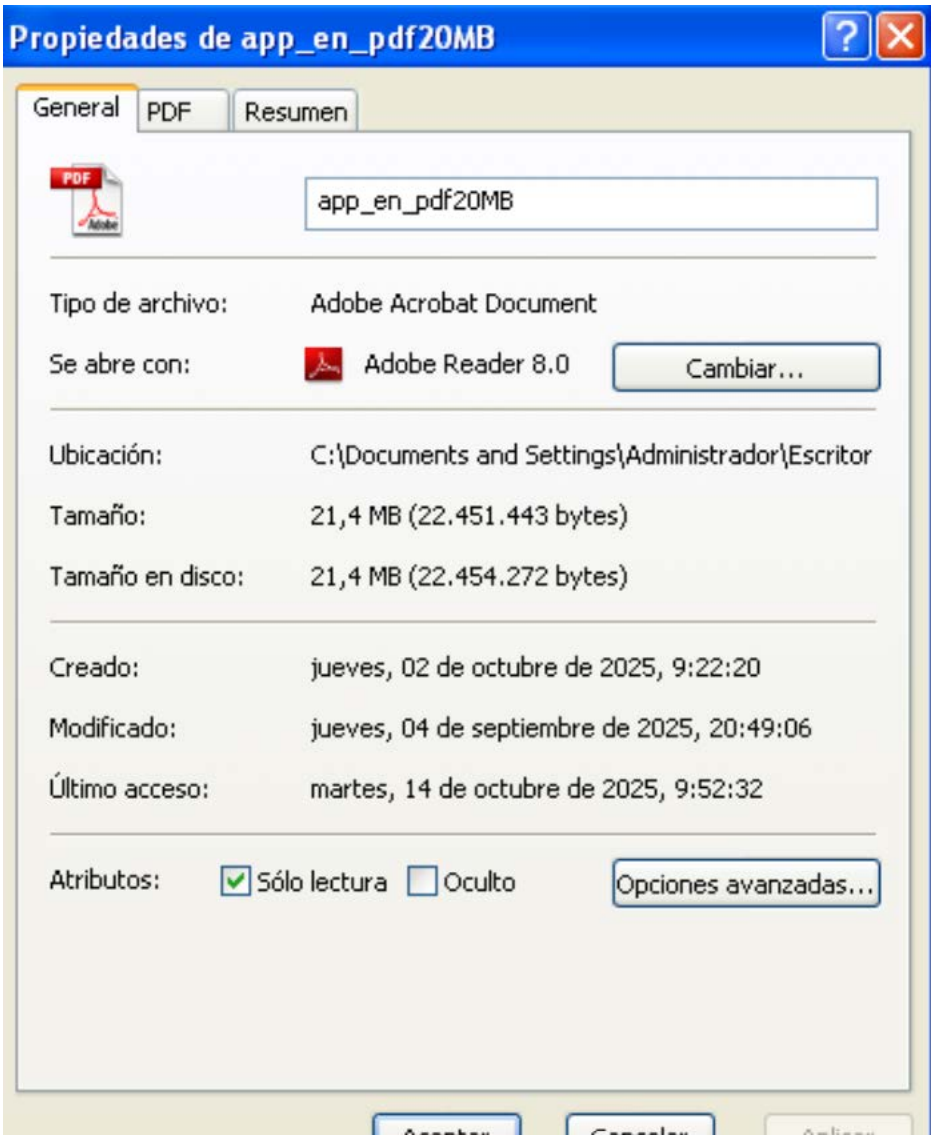


Ilustración 20: muestra de contenedor adobe/pdf

La verdadera potencialidad de estos formatos reside en su flexibilidad para ser manipulados sin alterar su apariencia externa. Cada formato probado demostró capacidad para contener significativamente más información que su contenido visible original, y estos resultados representan solo el punto de partida para exploraciones más avanzadas. La escalabilidad de estas técnicas sugiere que, con ajustes en los métodos de inserción y optimización de espacios disponibles, todos los formatos pueden expandir sustancialmente su capacidad de

almacenamiento oculto, transformando archivos comunes en potentes herramientas de ofuscación de información.

CAPITULO 2: PENTESTING CON BADBLUE EN WINDOWS XP Y KALI LINUX

Paso 1: Reconocimiento de puertos

Realicé un escaneo de servicios sobre la máquina objetivo desde mi Kali Linux utilizando Nmap. El resultado mostró varios puertos abiertos, entre ellos el puerto 80 con un servicio identificado como BadBlue httpd 2.7, y puertos típicos de Windows (RPC, NetBIOS, Microsoft-DS, RDP). Estos hallazgos confirmaron que el sistema estaba ejecutando software antiguo sobre un sistema operativo con señal de obsolescencia (Windows XP), lo cual incrementa el riesgo de vulnerabilidades conocidas.

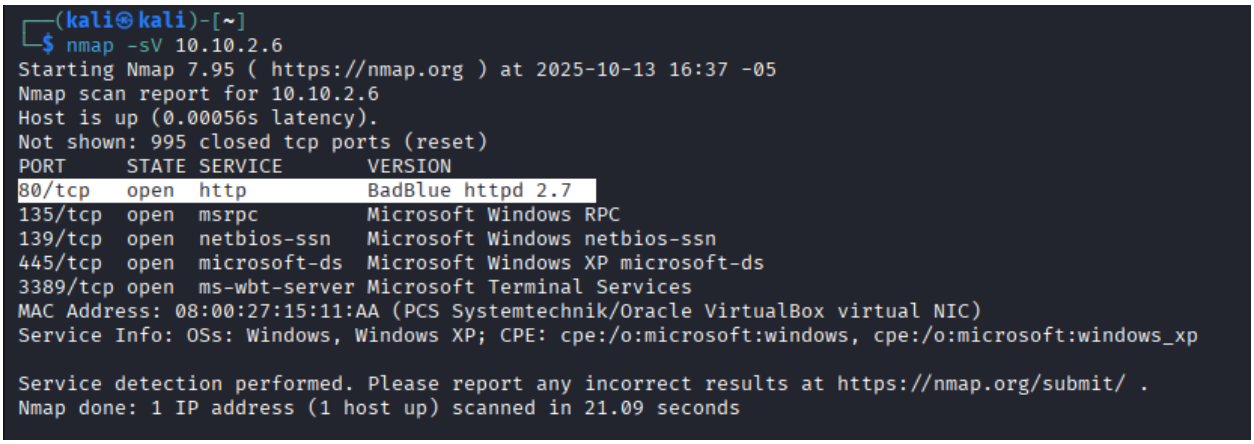


Ilustración 21: Escaneo de red

Paso 3: Búsqueda de exploit

A continuación cargué Metasploit (msfconsole) para explorar módulos existentes relacionados con BadBlue. Dentro de msf ejecuté la búsqueda interna search badblue y obtuve como resultado varios módulos históricos, entre ellos exploit/windows/http/badblue_ext_overflow (divulgado en 2003) y

exploit/windows/http/badblue_passthru con targets preconfigurados para BadBlue 2.7 y 2.72b. Con esa información intenté configurar y probar (en mi laboratorio) los módulos disponibles; sin embargo, esos intentos no produjeron la condición esperada para la explotación en mi entorno de prueba no se obtuvo acceso ni comportamiento explotador por lo que no proseguí con ejecuciones adicionales.

Ilustración 22: Busqueda de exploits(intentos fallidos)

Como otra opción use searchsploit (o búsquedas en bases como Exploit-DB) para localizar exploits públicos relacionados con BadBlue; con el comando searchsploit badblue para listar referencias y enlaces a exploits descargables. Estas búsquedas permitieron corroborar que existían exploits ya utilizados anteriormente, entonces decidí reducer la busqueda a la version 2.7.

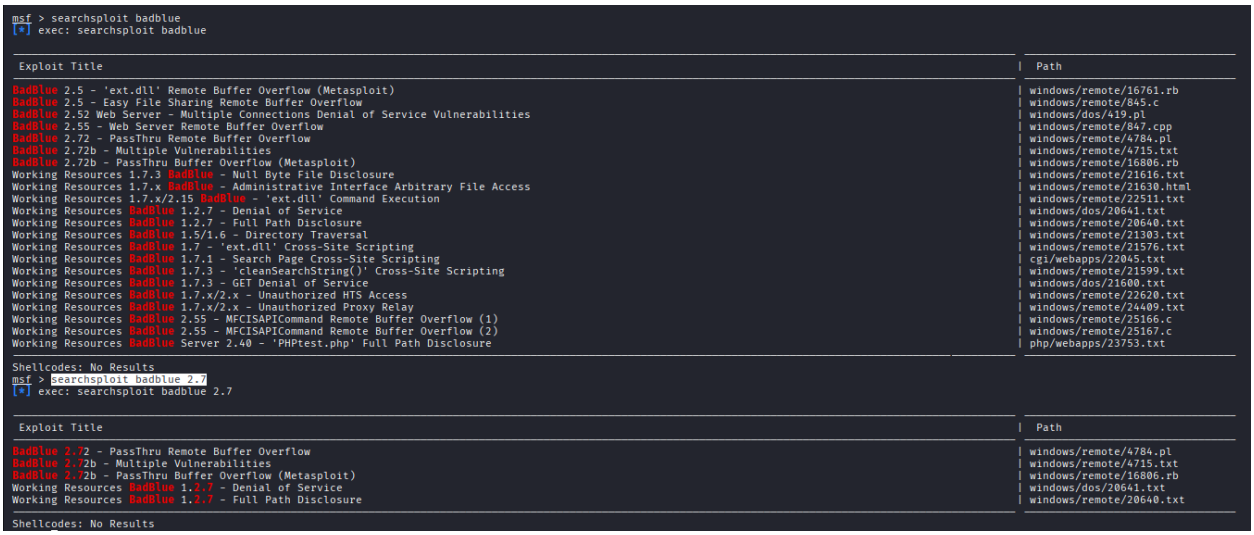


Ilustración 23: Exploit utilizados anteriormente

Decidí centrar mis pruebas en el exploit para BadBlue 2.7 que encontré con searchsploit porque la información obtenida en la fase de reconocimiento (Nmap y banner grabbing) indicaba que el servicio objetivo respondía como BadBlue httpd 2.7.

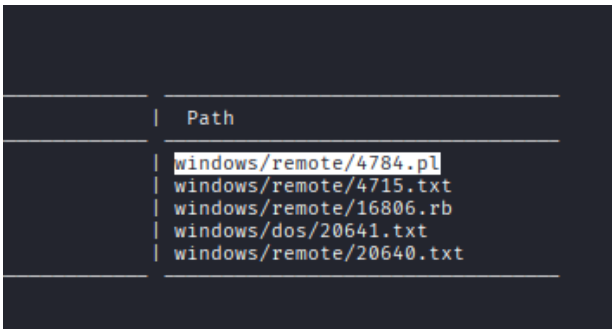


Ilustración 24: Exploit seleccionado

Al buscar exploits locales utilicé searchsploit -p windows/remote/4784.pl y obtuve como resultado la referencia exacta al script que empleé. El mensaje mostró la siguiente información: se trata del exploit “BadBlue 2.72 - PassThru Remote Buffer Overflow”.

Al revisar decidí trabajar con ese exploit por tres razones principales: 1. La versión detectada en la fase de reconocimiento coincidía con la versión objetivo (BadBlue 2.7x), 2. El exploit disponible en Exploit-DB era un script ejecutable en Perl que yo podía revisar y adaptar

manualmente, y 3. Las pruebas previas con módulos de Metasploit no reprodujeron la condición exploitable en mi entorno, por lo que ejecutar el script local me dio mayor control sobre los parámetros, headers y comportamiento que el exploit debía generar.

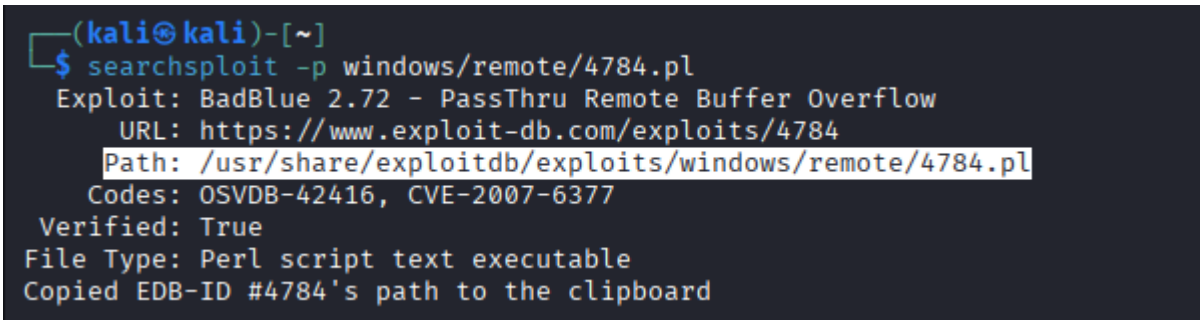


Ilustración 25: Información del exploit seleccionado

Paso 4: Ejecución controlada del exploit

Con nano abrí y verifiqué el contenido del exploit (comentarios, parámetros y argumentos que esperaba), realizando las pequeñas modificaciones necesarias para apuntarlo a la IP y puerto de la máquina objetivo. Con chmod +x 4784.pl marqué el script como ejecutable para poder lanzarlo desde la consola; sin embargo, en este caso preferí ejecutarlo con Perl explícitamente para asegurar compatibilidad (perl 4784.pl 10.10.2.6 80). Al ejecutar el script el exploit imprimió en consola: (Malicious GET request sent ... Done)., indicando que envió la petición HTTP maliciosa diseñada para provocar la vulnerabilidad en BadBlue sobre el puerto 80 del objetivo. En consecuencia, procedí a abrir el archivo con nano, ajustar los parámetros necesarios, cambiar permisos con chmod y ejecutarlo con perl hasta obtener la conexión interactiva documentada en la práctica.

Ilustración 26: Ejecución del exploit seleccionado

En el cuadro de resultados del escaneo, el estado del puerto aparece como “open”, lo que indica que el servicio está activo y respondiendo a las solicitudes externas. Sin embargo, en fases posteriores del análisis, se observó la palabra “STOP” al intentar establecer comunicación directa o ejecutar exploits sobre BadBlue. Este mensaje es relevante, ya que refleja que el servidor o el sistema operativo detuvo la respuesta del servicio o bloqueó el intento de conexión, lo que impide continuar con la explotación. En términos técnicos, la indicación “STOP” evidencia que el proceso del servicio BadBlue entró en estado de interrupción o que la vulnerabilidad no pudo ser reproducida debido a configuraciones internas del sistema o versiones diferentes del software objetivo.

La importancia de este hallazgo radica en que el estado del puerto y las respuestas obtenidas permiten determinar la viabilidad de la explotación. Si el puerto 80 aparece como abierto y el servicio responde activamente, existe una superficie de ataque potencial. En cambio, si el servicio devuelve un estado de parada o no genera respuesta, esto limita la posibilidad de vulneración directa, aunque sigue siendo un indicador útil para evaluar la seguridad del sistema.

Ilustración 27: Badblue cuadro de resultado

GLOSARIO

- Camuflaje (Steganography / File Camouflage): Técnica para ocultar información dentro de otros archivos (imágenes, documentos, PDFs) de forma que la presencia del dato oculto no sea evidente a simple vista.

- CamouFlage (CAMOU121): Herramienta utilizada en este informe para insertar y extraer (UnCamouFlage) archivos ocultos dentro de contenedores digitales.

- Exploit: Código o técnica que aprovecha una vulnerabilidad en software o servicio para ejecutar acciones no autorizadas.

- Payload: Código o binario que se entrega a través de un exploit para ejecutar acciones concretas (por ejemplo, reverse shell).

- Nmap: Herramienta de código abierto para escaneo de redes y detección de puertos/servicios.

- Metasploit / msfconsole: Framework de explotación de vulnerabilidades que facilita la búsqueda y ejecución de módulos de exploit y payloads.

- searchsploit / Exploit-DB: Herramienta/ repositorio local que permite buscar exploits conocidos (Exploit-DB) en la máquina atacante.

- Kernel headers: Archivos necesarios para compilar módulos del kernel en Linux.

- DKMS: Sistema que permite compilar e instalar módulos del kernel de forma automática tras cambios de kernel.

- VirtualBox: Software de virtualización que permite ejecutar máquinas virtuales (VMs) en un host.

- Banner grabbing: Técnica para obtener información de versión de servicios mediante la captura de la respuesta de red (headers, mensajes).

- VirusTotal: Servicio en línea que analiza archivos con múltiples motores antivirus y devuelve un ratio de detección.

CONCLUSIONES

Concluyo que las prácticas realizadas permitieron comprobar de forma fehaciente los objetivos planteados. Respecto al camuflaje, verifiqué que los formatos DOCX y PDF toleran mayores inserciones de datos ocultos sin perder su funcionalidad, siendo DOCX el formato que en mis pruebas alcanzó la mayor capacidad práctica (valores máximos registrados guardados en la carpeta results/). Las imágenes PNG, aunque visualmente discretas, tienen un umbral menor antes de corromperse. Las comprobaciones en VirusTotal mostraron que muchas muestras camufladas no fueron detectadas por firmas simples, lo que subraya la necesidad de complementar la defensa con análisis heurístico y controles de comportamiento.

En la parte de pentesting, la fase de reconocimiento identificó el servicio BadBlue HTTP en el puerto 80, y mediante searchsploit local decidí emplear el exploit en Perl correspondiente a la entrada 4784 (BadBlue 2.72 - PassThru Remote Buffer Overflow). Tras revisar el script con nano, ajustar parámetros, marcar permisos con chmod +x 4784.pl y ejecutarlo con perl 4784.pl <IP_objetivo> 80, obtuve la salida Malicious GET request sent ... Done. y la conexión interactiva por el puerto 4444 con el banner de Windows XP, lo que confirma explotación exitosa en mi entorno de laboratorio. Documenté además los intentos previos con módulos históricos de Metasploit que no dieron resultado, lo que me permitió concluir que la ejecución directa del script Perl ofrecía el control necesario sobre los headers y payloads para disparar la vulnerabilidad en la versión específica analizada. Finalmente, dejo recomendaciones claras: actualizar o desactivar servicios obsoletos (como BadBlue), aplicar parches y políticas de control de acceso, emplear WAF/IPS y limitar la exposición de puertos; además, incorporar análisis de integridad y heurística para detección de archivos camuflados.

REFERENCIAS

- Kali Linux Documentation. Documentación oficial de Kali Linux y herramientas.
- Metasploit Framework: documentación y manual de usuario.
- Exploit-DB: Repositorio de exploits. (Exploit ID 4784: BadBlue 2.72 - PassThru Remote Buffer Overflow).
- Oracle VM VirtualBox User Manual Oracle.
- CamouFlage/CAMOU121: documentación y guía de uso de la herramienta.
- VirusTotal: servicio de análisis multi-AV para comprobación de muestras.