UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO DE FIN DE CARRERA

Análisis de herramientas y técnicas de apoyo a la recuperación de información cifrada

Javier Manzano Vázquez

2012

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

INGENIERÍA DE TELECOMUNICACIÓN

Trabajo Fin de Carrera

Análisis de herramientas y técnicas de apoyo a la recuperación de información cifrada

Autor: Javier Manzano Vázquez
Director: Enrique de la Hoz de la Hoz

TRIBUNAL:

Presidente: D. Iván Marsá Maestre

Vocal 1º: D. Juan Ignacio Pérez Sanz

Vocal 2º: D. Enrique de la Hoz de la Hoz

CALIFICACIÓN.....

FECHA: .....

Un globo, dos globos, tres gloooobos...

la luna es un globo que se me escapó...

Dedicado a mi Madre

Índice

Índice de Figuras

Resumen

El proyecto aborda el problema de la optimización de los procesos de descifrado de evidencias informáticas protegidas con contraseña. En este proyecto se analizan alternativas tecnológicas para la realización de una plataforma de tratamiento masivo de información cifrada utilizando la tecnología GPGPU *(General-Purpose Computing on Graphics Processing Units)* para procesar datos. Dentro de este contexto, también se estudia la viabilidad de la utilización de esta tecnología GPU dentro de un entorno de virtualización basado en Xen y se adaptan soluciones existentes para poder utilizar la tarjeta gráfica nativa por los huéspedes virtuales. El objetivo final que se persigue es posibilitar la integración de distintas herramientas de descifrado en una misma plataforma con independencia del sistema operativo para el que fueron desarrolladas.

Palabras clave: Informática forense, criptografía, GPU, XEN, VGA passthrough.

Summary

The project addresses the problem of decrypting password-protected computer evidences. This project will analyze differents technological alternatives in order to achieve the realization of a decryption platform using the GPGPU technology (General-Purpose Computing on Graphics Processing Units) to process data. Within this context, the project examines the feasibility of using this GPU technology within a virtualization environment based on Xen and adapt existing solutions to use the native graphics card inside the virtual guests. The ultimate goal pursued is to enable the integration of various cracking tools on the same system despite of the operating system for which they were developed.

Key words: Computer forensics, Cryptography, GPU, XEN, VGA passthrough.

1

Introducción

La informática forense es una disciplina basada en la aplicación de técnicas científicas y analíticas en cualquier tipo de dispositivo tecnológico permitiendo identificar, analizar y presentar datos que sean válidos dentro de un proceso legal.

Una de las tareas de la informática forense es buscar y analizar información en sistemas de ordenadores para buscar evidencias potenciales de un delito, sin embargo, muchas veces los datos más críticos que podrían ser usados como pruebas en un juicio se encuentran cifrados o protegidos con contraseña por lo que se deben utilizar herramientas que permitan el descifrado de estos archivos en el menor tiempo posible.

Estas herramientas desde hace unos años están experimentando un fuerte cambio gracias a las tecnologías CUDA[1] y OpenCL[2] que permiten utilizar las tarjetas gráficas para el cómputo de las instrucciones de una manera concurrente, aprovechando el alto grado de paralelización que ofrecen las tarjetas modernas (Graphic Process Unit o GPU).

Estas tecnologías surgieron en el año 2007 y fueron rápidamente incorporadas a las herramientas comerciales de descifrado de archivos. Las herramientas de código libre sin embargo, han experimentado un desarrollo más lento y es ahora cuando estas aplicaciones están lo suficientemente maduras como para poder realizar un estudio detallado de las mismas.

Para poder comparar el rendimiento de las diferentes herramientas de código libre tenemos a nuestra disposición el programa comercial de descifrado de archivos *Passware Kit Forensic*[3] que servirá de base para estudiar el rendimiento de las diferentes

[1] CUDA: *Compute Unified Device Architecture,* herramienta de desarrollo creada por nVidia que permite a los programadores usar una variación del lenguaje de programación C para codificar algoritmos en GPUs de nVidia.

[2] OpenCL: *Open Computing Language*, consta de una interfaz de programación y un lenguaje de programación que permiten crear aplicaciones con paralelismo a nivel de datos y tareas que pueden ejecutarse tanto en CPU como en unidades gráficas.

[3] *Passware* es una empresa de reconocido prestigio especializada en la recuperación de contraseñas de diferentes tipos de archivos. Para la realización del proyecto se dispone de la licencia de uso de su producto estrella, el programa *Passware Kit Forensic*.

herramientas. Esto nos permitirá seleccionar aquellos programas de descifrado que sirvan como alternativa a las costosas herramientas comerciales.

La capacidad de procesar datos en tarjetas gráficas GPU es tan potente que incluso los entornos de virtualización están intentando facilitar el uso de la tarjeta gráfica nativa por los huéspedes virtuales. El problema es que existen muchos obstáculos que impiden que esta tarea pueda realizarse de manera sencilla y por ello todavía no existe una solución adecuada al alcance de los usuarios. La capacidad de utilizar la tarjeta gráfica nativa por las máquinas virtuales es de gran interés para el proyecto ya que permitiría unificar en un mismo sistema las mejores herramientas de descifrado actuales sea cuál sea el sistema operativo que utilicen.

Uno de los entornos de virtualización que más avanzado está en este aspecto es el proyecto de código libre Xen que incluye algunas soluciones para poder utilizar diferentes modelos de tarjetas gráficas en las máquinas virtuales. Una de las partes importantes de este proyecto será realizar un estudio detallado de este tipo de soluciones.

Otro de los aspectos interesantes de la tecnología GPU es su reciente incorporación a plataformas de computación en la nube como Amazon-EC2[4] que ha aprovechado esta tecnología para ofrecer a sus usuarios una gran capacidad de cómputo a bajo coste lo que puede ser de gran utilidad a la hora de agilizar el proceso de descifrado de archivos.

Todos estos avances ofrecen la oportunidad de mejorar las herramientas de recuperación de información cifrada y por ello a lo largo de este proyecto se abordarán los diferentes temas aquí expuestos tratando de lograr la consecución de los objetivos que se detallan a continuación.

[4] Amazon-EC2 o *Amazon Elastic Compute Cloud* es un servicio web que proporciona capacidad informática con tamaño modificable en la nube. Está diseñado para facilitar a los desarrolladores recursos informáticos escalables y basados en web. Existen varios tipos de instancias entre los cuales destacan las instancias basadas en GPU.

2

Objetivos del Proyecto

El objetivo principal del proyecto es allanar el camino para la realización de una plataforma de descifrado que unifique las mejores herramientas de descifrado de archivos y las últimas tecnologías existentes en la actualidad.

La realización de esta plataforma puede materializarse en forma de sistema centralizado o distribuido.

- Sistema centralizado: la plataforma de descifrado se realizará en una máquina de alto rendimiento o *cluster* utilizando la tecnología de procesado de las tarjetas gráficas modernas GPU *(Graphics Processing Unit).* En este *cluster* se podrán incluir todas las herramientas de seguridad elegidas independientemente del sistema operativo que utilicen. Para ello será necesario la utilización de tecnologías de virtualización que permitan el uso de las tarjetas gráficas GPU en las máquinas virtuales.
- Sistema distribuido: la plataforma de descifrado estará formada por un conjunto de nodos entre los que se distribuirá el ataque de descifrado que se desee realizar. Entre estos nodos se contempla la posibilidad de utilizar la capacidad de procesamiento de sistemas en la nube que utilicen tarjetas GPU.

El objetivo del proyecto no es la creación de la plataforma final sino realizar el estudio y la adaptación de las herramientas y tecnologías necesarias que permitan la creación de este tipo de sistemas. Para ello será necesaria la consecución de unos objetivos parciales que se detallan a continuación:

- Realizar un análisis de la seguridad que presentan los archivos que se pretenden descifrar en la plataforma. En concreto se deberá analizar la estructura de los archivos más comunes utilizados por los usuarios de Windows:
  - Archivos comprimidos Winrar.
  - Archivos comprimidos Zip.
  - Documentos de Office ya sean Word, Excel o PowerPoint.
  - Archivos PDF.
- Realizar una búsqueda de herramientas que analicen unidades de memoria y sean capaces de identificar los archivos cifrados o protegidos con contraseña de apertura aportando información sobre el tipo de cifrado que presentan. En el caso de no encontrar tales herramientas se deberá estudiar el uso de indexadores y desarrollar una herramienta que los identifique.

- Buscar y analizar herramientas de software libre o de licencia gratuita que sirvan de alternativa a las costosas herramientas de descifrado comerciales desarrolladas para Windows.
- Seleccionar las mejores herramientas para el descifrado de archivos y prepararlas para automatizar el lanzamiento de ataques de descifrado tanto de forma local como de forma remota. En el caso de no haber encontrado herramientas que sustituyan a los programas comerciales se deberá automatizar la herramienta comercial *Passware Kit Forensic* (La licencia de este programa es uno de los recursos disponibles para la realización del proyecto).
- Analizar la posibilidad de utilizar las tecnologías de computación en la nube ofrecidas por Amazon en el que se incluyen tarjetas gráficas GPU
- Realizar un estudio sobre el entorno de virtualización Xen y la característica *VGA-Passthrough* diseñada para utilizar las tarjetas gráficas nativas en las máquinas virtuales. Esto permitirá unificar en una misma máquina todas las herramientas de seguridad seleccionadas independientemente del sistema operativo que utilicen. El entorno de virtualización Xen da paso a la creación de un *cluster* GPU que ofrezca todas las ventajas que proporcionan las herramientas criptográficas desarrolladas para Windows, la seguridad y fiabilidad del sistema operativo Linux y la mejora de rendimiento que proporcionan las tarjetas GPU.
- Finalmente se definirán una serie de recomendaciones sobre qué herramientas utilizar para el descifrado de los diferentes tipos de archivo y el procedimiento a seguir para construir la plataforma de descifrado de la manera más completa, automatizada y unificada posible.

La memoria de este trabajo fin de carrera ha sido estructurada conforme a la lista de objetivos que acabamos de presentar. Los siguientes capítulos detallarán los estudios realizados y los resultados alcanzados para cada uno de los objetivos de la enumeración anterior.

3

Estado del Arte de la Protección de Archivos Digitales

En este apartado abordaremos los mecanismos de seguridad que implementan los diferentes tipos de archivos para proteger su contenido.

En primer lugar, cuando se quiere descifrar un archivo es importante saber identificar qué algoritmo se ha utilizado para proteger su contenido. Esto permitirá determinar la configuración apropiada de las herramientas de descifrado de cara a utilizar los ataques adecuados en función del algoritmo de cifrado.

Aunque existen varias herramientas comerciales que identifican el tipo de cifrado utilizado para proteger un archivo, se pretende buscar alternativas de código abierto o de libre distribución que permitan llevar a cabo esa misma tarea. Antes de poder estudiar estas herramientas, debemos conocer cuáles son los principales estándares y mecanismos empleados para la creación de archivos cifrados. Al analizar la estructura de los documentos, se podrá conocer a priori la seguridad que utilizan y si fuera necesario se podría desarrollar una aplicación que identificase automáticamente el cifrado de cada uno de los archivos de interés. Además realizando este análisis se tendrá una clara idea de qué tipo de archivos son los más robustos en cuanto a la protección de la información que contienen.

A continuación se detalla la forma de identificar el tipo de algoritmo utilizado en archivos de *Microsoft Office*, archivos comprimidos RAR, archivos comprimidos ZIP, y archivos PDF.

3.1 Archivos DOCX

El paquete de utilidades *Microsoft Office* es ampliamente utilizado por los usuarios de Windows y todas sus utilidades soportan la protección de documentos mediante criptografía. En este apartado se detalla la estructura interna de los documentos de *Office* 2007 en adelante haciendo hincapié en los documentos de Word, no obstante, todo lo expuesto a continuación es aplicable a cualquier tipo de documento de *Office* 2007 o posterior.

Toda la información relativa al formato de los archivos de Office incluyendo los aspectos de seguridad se encuentra en la *Open XML Specification* de Microsoft[5]. Esta especificación es muy extensa y cubre muchos aspectos relacionados con la estructura interna de los documentos de Word. Para nuestros objetivos, analizaremos cómo se puede obtener información útil relativa a documentos protegidos mediante cifrado. Ejemplos de esto son identificar el tipo de cifrado del archivo o saber dónde se encuentra el verificador de la contraseña (si es que existe uno).

3.1.1 Identificar el tipo de cifrado

Un documento de Word es simplemente un fichero comprimido ZIP que contiene diferentes archivos XML en su interior donde se definen el contenido, el estilo y las propiedades del documento.

Así pues, abriendo un documento con extensión 'docx' con una herramienta de compresión de archivos ZIP (por ejemplo utilizando la utilidad de *software* libre 7zip[6]) se pueden ver todos los archivos XML que lo forman y leer el contenido del documento. Si el documento ha sido protegido con contraseña y por lo tanto está cifrado, los archivos XML estarán en una estructura de directorios como la mostrada en la figura:

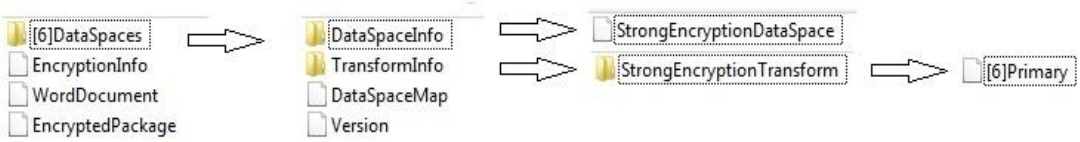


Fig. 1 – Estructura de directorios de un archivo docx cifrado

Cabe mencionar que en la especificación de Microsoft se definen varios tipos de estructuras dependiendo del algoritmo de cifrado, sin embargo, al proteger un documento de Word no se tiene opción de elegir como se cifrará sino que se utiliza automáticamente el método definido en la especificación como *Standard Encryption Approach*. Este método presenta la estructura de directorios mostrada en la figura y utiliza AES y SHA-1 como algoritmos de cifrado y *hash* respectivamente.

Los archivos XML mostrados en la figura anterior se denominan *streams.* Estos *streams* contienen información muy diversa, en concreto los *streams* ***EncryptionInfo*** y ***0x06Primary*** son los que contienen la información sobre el algoritmo de cifrado utilizado. Al abrir el *stream* ***EncryptionInfo*** con un editor de textos o con un editor hexadecimal se puede leer entre otros datos la cadena "M i c r o s o f t E n h a n c e d R S A a n d A E S

[5] Office Open XML es un estándar abierto propuesto para documentos de procesamiento de textos, presentaciones y hojas de cálculo que puede ser libremente implementado por múltiples aplicaciones en multitud de plataformas. El estándar OpenXML fue diseñado principalmente para describir los formatos binarios utilizados por Microsoft en sus aplicaciones de Office.

[6] 7zip es una herramienta de compresión de archivos que permite abrir los archivos de Word mostrando su estructura interna. Otras utilidades sin embargo detectan errores en el formato y muestran el mensaje 'formato dañado'.

C r y p t o g r a p h i c P r o v i d e r" donde se puede ver que el cifrado se realiza con un proveedor de servicios de Microsoft que implementa el algoritmo AES. Según la especificación, si la información del *stream* ***0x06Primary*** difiere de este, se debe dar prioridad a lo que ponga en ***EncryptionInfo***. En el ejemplo de la figura el *stream* ***0x06Primary*** muestra la cadena "M i c r o s o f t . C o n t a i n e r . E n c r y p t i o n T r a n s f o r m AES 128" por lo que se verifica que el algoritmo utilizado es AES en este caso con 128 bits de longitud.

3.1.2 Verificador de la contraseña

El verificador de la contraseña del documento sirve para comprobar si la contraseña introducida es correcta antes de llevar a cabo el descifrado; se evita así el tener que descifrar el documento y comprobar después si su contenido es legible (contraseña correcta) o no (contraseña incorrecta). El verificador en los documentos de Word se encuentra en el *stream* ***EncryptionInfo*** en el objeto ***EncryptionVerifier*** que se utilizará para verificar si la contraseña introducida es correcta o no. Este verificador se obtiene tras una larga serie de operaciones que lo protegen frente a ataques de fuerza bruta y frente ataques con tablas *rainbow*[7].

Para proteger al verificador de los ataques de fuerza bruta se realizan 50.000 repeticiones del algoritmo hash SHA-1 aplicado a la contraseña, de esta forma cuando un programa de seguridad quiera realizar un ataque de fuerza bruta, deberá realizar 50.000 repeticiones del algoritmo hash para cada una de las combinaciones de caracteres que quiere probar. Este hecho provoca un retardo considerable, protegiendo al documento de posibles ataques.

Para proteger al verificador de ataques con tablas *rainbow* se concatena una semilla o *salt* aleatoria a la contraseña antes de realizar el cálculo del hash, evitando así que los *hashes* precalculados de las tablas puedan coincidir con el verificador.

Todas estas operaciones se detallan en mayor profundidad en la sección 2.3.4.7 del documento MS-OFFCRYPTO que contiene la especificación pública de Microsoft. Este documento se incluye en el CD-ROM adjunto al proyecto en el directorio *MicrosoftCryptoDocs* o bien puede obtenerse directamente desde la web de Microsoft:

http://msdn.microsoft.com/en-us/library/cc313071%28v=office.12%29.aspx

[7] Las tablas arcoíris (*rainbow table*) son tablas que contienen los *hashes* precalculados de multitud de contraseñas en claro. Estas tablas permiten utilizar el espacio de almacenamiento en vez de utilizar la capacidad de procesamiento de la CPU.

3.2 Archivos DOC

En el caso de los documentos con extensión *doc*, Microsoft proporciona el documento *MS-DOC Word(.doc) Binary File Format*. Éste puede encontrarse en la carpeta *MicrosoftCryptoDocs* del CD-ROM o bien descargarlo desde la siguiente URL:

http://msdn.microsoft.com/en-us/library/cc313153%28v=office.12%29.aspx

Al igual que en el documento anterior, en éste se define la estructura de los documentos de Word ('doc') incluyendo los aspectos criptográficos que son de interés para el proyecto como por ejemplo identificar el tipo de cifrado utilizado o algunos fallos de seguridad existentes en los archivos .doc.

3.2.1 Identificar el tipo de cifrado

En primer lugar podemos ver la estructura de un archivo de Word ('doc') cifrado abriéndolo con un editor de archivos comprimidos ZIP. De esta forma se pueden ver los *streams* que lo forman:

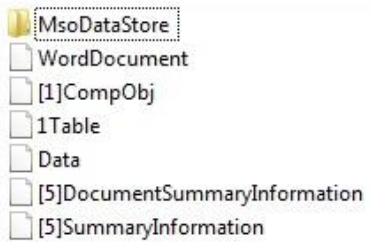


Fig. 2 - Estructura de directorios de un archivo doc cifrado

A diferencia de los documentos 'docx', los *streams* de los documentos 'doc' no contienen cadenas de texto que ayuden a identificar el algoritmo utilizado para cifrarlos. Debido a esto se debe profundizar en la estructura del documento para localizar los bytes que indican el algoritmo empleado. Según la especificación, esta información se encuentra definida por unos *flags* contenidos en el *stream* ***0x01Table***. Dentro de este *stream* hay una estructura llamada *FIB* que tiene un conjunto de punteros a estructuras con las propiedades del archivo 'doc'. Una de las partes de esta estructura *FIB* llamada *FIBbase* es la que contiene la serie de *flags* mencionados anteriormente. Entre estos *flags* se identifican dos bits llamados *FibBase.fEncryption* y *FibBase.fObfuscated* en función de cuyo valor se distinguen dos tipos de cifrado:

- *XOR Obfuscation: FibBase.fEncryption=1* y *FibBase.fObfuscated=1*
- *RC4 Encryption* o *RC4 CryptoAPI Encryption : FibBase.fEncryption=1* y *FibBase.fObfuscated=0*

En la práctica, el método *XOR Obfuscation* no se utiliza nunca, siempre se emplea cifrado RC4 de 40 bits. A diferencia de los documentos 'docx' que utilizaban una clave de 128 bits (a no ser que se indique una mayor), en los documentos 'doc' la clave siempre es de 40 bits. Al ser la longitud de la clave tan corta es posible realizar ataques de fuerza bruta directamente sobre ésta y obtener así la contraseña independientemente de los

caracteres que tuviera. Éste es uno de los fallos en la seguridad de los documentos de Word, en el siguiente apartado se explican algunos más.

3.2.2 Fallos de seguridad y debilidades

3.2.2.1 *Streams* almacenados en claro

Uno de los fallos de seguridad en los documentos 'doc' es que no se cifran todos los *streams* que lo forman y por tanto hay mucha información sensible. Un ejemplo de esto puede ser el *stream* ***0x05DocumentSummaryInformation.*** En este *stream* se almacenan en claro todos los enlaces incluidos en el documento por lo que basta con abrirlo con un editor hexadecimal para poder leer el contenido:

[5]DocumentSummaryInformation

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000180	03	00	00	00	1B	00	00	00	03	00	00	00	00	00	00	00	....
00000190	03	00	00	00	05	00	00	00	1F	00	00	00	22	00	00	00	............"...
000001A0	68	00	74	00	74	00	70	00	3A	00	2F	00	2F	00	77	00	h.t.t.p.:././.w.
000001B0	77	00	77	00	2E	00	67	00	6F	00	6C	00	75	00	62	00	w.w...g.o.l.u.b.
000001C0	65	00	76	00	2E	00	63	00	6F	00	6D	00	2F	00	72	00	e.v...c.o.m./.r.
000001D0	61	00	72	00	67	00	70	00	75	00	2E	00	68	00	74	00	a.r.g.p.u...h.t.
000001E0	6D	00	00	00	1F	00	00	00	01	00	00	00	00	00	69	09	m.............i.
000001F0	03	00	00	00	16	00	72	00	03	00	00	00	18	00	00	00	......r.........
00000200	03	00	00	00	00	00	00	00	03	00	00	00	05	00	00	00	....
00000210	1F	00	00	00	2A	00	00	00	68	00	74	00	74	00	70	00	....*...h.t.t.p.
00000220	3A	00	2F	00	2F	00	67	00	6F	00	6C	00	75	00	62	00	:././.g.o.l.u.b.
00000230	65	00	76	00	2E	00	63	00	6F	00	6D	00	2F	00	66	00	e.v...c.o.m./.f.
00000240	69	00	6C	00	65	00	73	00	2F	00	69	00	67	00	72	00	i.l.e.s./.i.g.r.
00000250	61	00	72	00	67	00	70	00	75	00	5F	00	76	00	30	00	a.r.g.p.u._.v.0.
00000260	35	00	2E	00	7A	00	69	00	70	00	00	00	1F	00	00	00	5...z.i.p.......
00000270	01	00	00	00	00	00	69	09	03	00	00	00	18	00	45	00	......i.......E.
00000280	03	00	00	00	15	00	00	00	03	00	00	00	00	00	00	00	....
00000290	03	00	00	00	05	00	00	00	1F	00	00	00	31	00	00	00	............1...
000002A0	68	00	74	00	74	00	70	00	3A	00	2F	00	2F	00	73	00	h.t.t.p.:././.s.
000002B0	6F	00	75	00	72	00	63	00	65	00	66	00	6F	00	72	00	o.u.r.c.e.f.o.r.
000002C0	67	00	65	00	2E	00	6E	00	65	00	74	00	2F	00	70	00	g.e...n.e.t./.p.
000002D0	72	00	6F	00	6A	00	65	00	63	00	74	00	73	00	2F	00	r.o.j.e.c.t.s./.
000002E0	75	00	6E	00	72	00	61	00	72	00	65	00	78	00	74	00	u.n.r.a.r.e.x.t.
000002F0	72	00	61	00	63	00	74	00	72	00	65	00	63	00	2F	00	r.a.c.t.r.e.c./.
00000300	00	00	00	00	1F	00	00	00	01	00	00	00	00	00	69	09	..............i.
00000310	03	00	00	00	1B	00	58	00	03	00	00	00	12	00	00	00	......X.........
00000320	03	00	00	00	00	00	00	00	03	00	00	00	05	00	00	00	....
00000330	1F	00	00	00	16	00	00	00	68	00	74	00	74	00	70	00	........h.t.t.p.
00000340	3A	00	2F	00	2F	00	77	00	77	00	77	00	2E	00	63	00	:././.w.w.w...c.
00000350	72	00	61	00	72	00	6B	00	2E	00	6E	00	65	00	74	00	r.a.r.k...n.e.t.
00000360	2F	00	00	00	1F	00	00	00	01	00	00	00	00	00	69	09	/.............i.
00000370	03	00	00	00	73	00	66	00	03	00	00	00	0F	00	00	00	....s.f.........
00000380	03	00	00	00	00	00	00	00	03	00	00	00	05	00	00	00	....
00000390	1F	00	00	00	43	00	00	00	68	00	74	00	74	00	70	00	....C...h.t.t.p.
000003A0	3A	00	2F	00	2F	00	77	00	77	00	77	00	2E	00	75	00	:././.w.w.w...u.
000003B0	6E	00	69	00	78	00	2D	00	61	00	67	00	2E	00	75	00	n.i.x.-.a.g...u.
000003C0	6E	00	69	00	2D	00	6B	00	6C	00	2E	00	64	00	65	00	n.i.-.k.l...d.e.
000003D0	2F	00	7E	00	63	00	6F	00	6E	00	72	00	61	00	64	00	/.~.c.o.n.r.a.d.

Fig. 3 – Contenido hexadecimal del *stream 0x05DocumentSummaryInformation*

En la imagen se pueden ver uno por uno todos los enlaces contenidos en un documento de Word cifrado. Esto puede ser de utilidad en algunos casos permitiendo conocer detalles del autor o del documento sin necesidad de utilizar ninguna herramienta adicional.

3.2.2.2 Debilidad en el procedimiento de cifrado de documentos Word (doc)

En la especificación de Microsoft, en concreto en la sección *Security Considerations* del documento *MS-OFFCRYPTO* (incluido en el CD-ROM) se enumeran una serie de

debilidades del algoritmo empleado en los documentos con extensión .doc, algunas de estas se listan a continuación:

- El algoritmo de generación de claves especificado en la sección 2.3.5.2 es débil debido a la ausencia de un mecanismo de repeticiones o iteraciones por lo que la contraseña puede ser objeto de ataques rápidos de fuerza bruta.
- El algoritmo de cifrado RC4 utiliza un generador de números pseudo-aleatorios para crear las claves de sesión. Existe una vulnerabilidad[8] conocida en relación al primer byte de la secuencia pseudo-aleatoria. Para solventar esta vulnerabilidad se recomienda no utilizar el primer byte del número pseudo-aleatorio, sin emabargo, los documentos de Office generan las claves de sesión a partir del primer byte de esta secuencia.
- No hay ningún mecanismo para comprobar corrupción en los *streams* cifrados lo que expone a los datos cifrados a posibles ataques *bit-flip.*
- La pequeña longitud de clave utilizada permite realizar ataques de fuerza bruta directamente en la clave sin tener la necesidad de conocer la contraseña.
- Algunos *streams* no se cifran.
- La clave puede ser reutilizada en los *streams* de datos del documento, incluido algunos con plantillas conocidas por lo que algunas porciones del texto del documento pueden ser extraídas directamente u obtenerse de manera trivial.
- Las propiedades del documento no se cifran lo que supone una fuga de información.
- Debido a todas las debilidades presentes, el mecanismo *RC4 CryptoAPI encryption* se considera inseguro y por lo tanto no se recomienda su uso para almacenar información sensible.

3.2.2.3 Fallo en la implementación del algoritmo RC4 en documentos de Word.

Como información adicional se incluye un documento[9] que explica algunos fallos de seguridad en la implementación realizada por Microsoft del algoritmo RC4 en sus archivos de Word. Aunque esto no es relevante para los objetivos del proyecto si sirve para reforzar la idea de la débil seguridad de este tipo de archivos.

---

[8] RC4 utiliza claves del estilo 'clave de sesión = vector de inicialización||clave principal'. Si el vector de inicialización obtiene el valor del primer byte de la secuencia pseudo-aleatoria este será con alta probabilidad idéntico al primer byte de la clave principal. Los problemas que esto supone se explican y aprovechan en el siguiente documento:

http://www.quequero.org/uicwiki/images/Attacks_on_the_RC4_stream_cipher.pdf

[9] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.2125&rep=rep1&type=pdf

En resumen el fallo de seguridad tiene que ver con el vector de inicialización (IV) del algoritmo RC4. Este vector debe ser distinto en cada nueva ejecución del algoritmo, sin embargo, en los documentos de Word el vector no se actualiza cuando se modifica un archivo ya cifrado. Debido a esto si se obtienen dos copias de un mismo documento con distinta fecha de modificación (es decir, que tengan alguna diferencia entre ellos) se podría obtener total o parcialmente la información del documento.

3.3 Archivos comprimidos ZIP

Los archivos comprimidos ZIP son creados principalmente con la herramienta de compresión de archivos Winzip, no obstante existen multitud de herramientas de este tipo que soportan dicha extensión y también pueden generar archivos comprimidos Zip. Todas estas utilidades mantienen la compatibilidad entre ellas implementando el cifrado de estos archivos comprimidos de la misma forma. Esto es posible gracias a que el formato de los documentos ZIP es público, incluyendo los mecanismos y algoritmos de seguridad utilizados para proteger la información.

El cifrado de los archivos comprimidos ZIP se realiza de acuerdo a la *Encryption Specification AE-1* y *AE-2*[10]. En concreto WinZip utiliza una librería *Open Source* que implementa el algoritmo AES para 128 y 256 bits. Esta librería fue desarrollada por el Dr. Gladman de la Universidad de Londres y puede obtenerse directamente de su página web:

- Página del proyecto *Open Source*:

  http://www.gladman.me.uk/cryptography_technology/fileencrypt/

- Enlace de descarga de la librería:

  http://gladman.plushost.co.uk/oldsite/cryptography_technology/fileencrypt/files.zip

El algoritmo de cifrado simétrico AES se incorporó a WinZip a partir de la versión 9.0 publicada en el año 2003. Antes de esto el algoritmo utilizado por Winzip para cifrar los datos comprimidos se llamaba *"Winzip Compatible 2.0".* Este algoritmo está actualmente obsoleto y puede romperse fácilmente realizando un ataque por fuerza bruta o utilizando alguna herramienta que aproveche algunas de las vulnerabilidades de este algoritmo.

[10] AE-1 y AE-2 son especificaciones de seguridad utilizadas por WinZip a partir de la versión 9.0. Ambas utilizan AES como algoritmo de cifrado, con longitudes de 128 y 256 bits. La única diferencia entre las dos especificaciones es el tratamiento del campo CRC. Más información sobre el uso del algoritmo AES en archivos ZIP puede encontrarse aquí:

http://www.winzip.com/aes_info.htm

http://www.winzip.com/aes_tips.htm

3.4 Archivos PDF

El uso de los archivos PDF está ampliamente extendido por los usuarios independientemente del sistema operativo que se utilice. Al igual que en los archivos de Office o en los archivos comprimidos ZIP, existe una especificación pública donde se define el formato de este tipo de documentos. Esta especificación varía dependiendo de la versión que se utilice. Los documentos utilizados incluidos en el CD-ROM son dos:

1. Versión 1.6:

   http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf

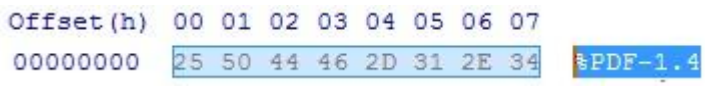
2. Versión 1.7:

   http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_1-7.pdf

Al guardar un documento PDF se ofrece la posibilidad de cifrarlo para proteger su contenido mediante una contraseña. Según la especificación cada una de las opciones que se ofrece utiliza un algoritmo diferente:

- *Compatible with Acrobat 3.0 and later*: Si se selecciona esta opción todos los contenidos se cifrarán utilizando el algoritmo RC4 con 40 bits de longitud.
- *Compatible with Acrobat 5.0 and later*: Si se selecciona esta opción todos los contenidos se cifrarán utilizando el algoritmo RC4 con 128 bits de longitud.
- *Compatible with Acrobat 6.0 and later*: Si se selecciona esta opción todos los contenidos se cifrarán utilizando el algoritmo RC4 con 128 bits de longitud. Esta versión permite dejar sin cifrar los metadatos[11] del archivo.
- *Compatible with Acrobat 7.0 and later*: A partir de aquí en adelante se utiliza el algoritmo AES con 128 bits.

Por lo tanto, para detectar que algoritmo se ha utilizado basta con saber con cuál de las opciones anteriores se cifró el archivo. Cuando un archivo se guarda con la opción de compatibilidad con una versión anterior se cambia automáticamente la versión real del documento, esta información puede encontrarse en los 8 primeros bytes de la cabecera:



Por tanto viendo que la versión es la 4.0 (1.4) podemos decir que el algoritmo utilizado es RC4 de 40 bits. Comprobémoslo:

[11] Los metadatos muestran información sobre los propios datos contenidos en el documento como por ejemplo nombre del archivo, fecha de modificación, fecha de creación, autor etc.

En primer lugar, debemos verificar si existe la entrada */Encrypt* que indica si el documento está cifrado o no, debe aparecer dentro del objeto "*trailer*":

```
00000280  30 30 30 20 6E 0D 0A 30 30 30 30 30 30 33 37 34 31 20 30 30 30 30 30 20 6E 0D 0A 74 72 61 69 6C  000 n..0000003741 00000 n..trail
000002A0  65 72 0D 3C 3C 0D 2F 53 69 7A 65 20 32 37 32 30 0D 2F 49 6E 66 6F 20 32 36 39 34 20 30 20 52 20  er.<<./Size 2720./Info 2694 0 R
000002C0  0D 2F 45 6E 63 72 79 70 74 20 32 36 39 38 20 30 20 52 20 0D 2F 52 6F 6F 74 20 32 36 39 37 20 30  ./Encrypt 2698 0 R ./Root 2697 0
000002E0  20 52 20 0D 2F 50 72 65 76 20 33 30 38 32 30 37 30 20 0D 2F 49 44 5B 3C 31 64 38 36 32 30 37 36   R ./Prev 3082070 ./ID[<1d862076
```

Ahora se debe ver qué valor tiene la entrada "V" dentro del objeto *filter* que viene a continuación:

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
000003C0   30 20 6F 62 6A 0D 3C 3C 20 0D 2F 46 69 6C 74 65  0 obj.<< ./Filte
000003D0   72 20 2F 53 74 61 6E 64 61 72 64 20 0D 2F 52 20  r /Standard ./R
000003E0   32 20 0D 2F 4F 20 28 C1 50 13 21 4A 89 09 98 5A  2 ./O (ÁP.!J‰.˜Z
000003F0   C1 4D AD FB 85 48 92 1C EB EE 30 70 75 A2 7C AD  ÁM.û…H'.ëî0pu¢|.
00000400   CB 23 AC 35 A6 45 A2 29 0D 2F 55 20 28 E9 E3 68  Ë#¬5¦E¢)./U (éãh
00000410   37 E9 3F 0B 27 5F 31 02 8D 34 5E FB 7E 1D 98 B8  7é?.'_1..4^û~.˜¸
00000420   14 C5 60 B7 E0 24 AC 9A B3 E3 A6 34 A4 29 0D 2F  .Å`·à$¬š³ã¦4¤)./
00000430   50 20 2D 36 30 20 0D 2F 56 20 31 20 0D 2F 4C 65  P -60 ./V 1 ./Le
00000440   6E 67 74 68 20 34 30 20 0D 3E 3E 20 0D 65 6E 64  ngth 40 .>> .end
00000450   6F 62 6A 0D 32 37 31 38 20 30 20 6F 62 6A 0D 3C  obj.2718 0 obj.<
```

Como se puede ver en la imagen en las últimas 3 líneas tenemos el valor de la entrada "V" y el valor de la entrada "Length", el valor V=1 indica que el algoritmo es RC4 y el valor de la entrada *Length* nos dice que la longitud son 40 bits.

No obstante con sólo verificar la versión podemos estar seguros del algoritmo utilizado.

3.5 Archivos RAR

La especificación del formato de los archivos comprimidos RAR no es pública. Desde la versión WinRAR 3.0, el algoritmo utilizado para proteger la información de sus documentos es AES con una longitud de clave de 128 bits. Además la función de derivación de clave[12] utilizada es especialmente lenta lo que ralentiza al máximo los intentos de descubrir la contraseña mediante ataques de fuerza bruta. Esto lo hace especialmente seguro, incluso mucho más que otros compresores que usan claves de 256 bits. Por ejemplo, un ataque a un fichero ZIP cifrado con AES-256 es 10 veces más rápido en las mismas condiciones que el mismo ataque a un fichero RAR cifrado con AES-128.

Para poder apreciar mejor la robustez de los archivos cifrados RAR se ha realizado una tabla comparativa entre archivos cifrados con extensión docx, rar y zip. Para ello se ha utilizado el software comercial de descifrado de archivos *Passware Kit Forensic* indicando

[12] La función de derivación de clave es el conjunto de operaciones que se realizan hasta obtener la clave de cifrado a partir de la contraseña de usuario. Entre estas operaciones se incluyen normalmente múltiples iteraciones de una función hash concreta así como el uso de un valor aleatorio llamado semilla (salt) que se concatena a la contraseña antes de proceder con las iteraciones.

el tiempo necesario para realizar un ataque de fuerza bruta a partir de letras en minúscula y cuatro caracteres de longitud máxima. Lo importante de esta comparativa son las diferencias de tiempo entre los diferentes archivos y no los tiempos de forma individual ya que estos cambiarán dependiendo del programa de descifrado utilizado y de la capacidad de procesamiento de la máquina donde se ejecute.

	Archivo .docx (AES 128 bits)	**Archivo .zip (AES 256 bits)**	**Archivo .rar (AES 128 bits)**
Tiempo total para realizar un ataque de fuerza bruta de 4 letras	8 minutos	20 segundos	15 minutos

Como se puede observar, a pesar de utilizar los tres el mismo algoritmo de cifrado, cada uno de ellos presenta un tiempo diferente, esto se debe a que cada programa implementa su propia función de derivación de claves realizando un número diferente de iteraciones para generar la clave con la que se cifrará el documento. Teniendo en cuenta estos valores orientativos se puede ver como los archivos RAR son mucho más robustos que los archivos comprimidos ZIP o que los documentos de Word 2007 en adelante.

3.6 Conclusión

Como se ha podido ver, todos los archivos analizados vienen utilizando desde hace unos años el algoritmo de cifrado AES que se considera seguro. Además para generar la clave de cifrado se realizan multitud de iteraciones de una función hash determinada lo que ralentiza en gran medida la velocidad de los posibles ataques que se realicen contra este tipo de archivos. En otras palabras, si se pretendiese atacar la seguridad de estos archivos mediante un ataque de fuerza bruta serían necesarios años de procesamiento para finalmente dar con la contraseña de apertura del documento. Esto puede entenderse más claramente analizando el siguiente caso:

Se tiene un documento de Word creado con la versión 2007 de Office. Al protegerlo con contraseña este se cifra automáticamente utilizando el algoritmo AES. Para su descifrado se dispone de la herramienta de seguridad *Passware Kit Forensic*. Se pretende realizar un ataque de fuerza bruta para probar todas las contraseñas posibles con una longitud de hasta 7 caracteres.

Según el programa *Passware*, hay más de 8 mil millones de contraseñas posibles de 7 caracteres de longitud si solo se utilizan letras en minúscula. Si se tiene en cuenta que además se suelen añadir números, letras mayúsculas o símbolos la cifra final ascendería a un total de 52 billones de contraseñas a probar.

Por otro lado, la velocidad máxima de ataque que alcanza *Passware* con un procesador moderno es del orden de 1000 *passwords* por segundo, supongamos siendo generosos que se obtiene una velocidad de 2000 psw/s. Si se utilizan tarjetas gráficas en el ataque se consigue un factor multiplicativo del rendimiento. Supongamos que se tiene una de las

mejores tarjetas del mercado y que se obtiene un factor que multiplica el rendimiento por 10:

Velocidad = 2000 psw/s x 10 = 20.000 psw/s.

Supongamos además que se dispone de un *cluster* que posee 10 de estas tarjetas gráficas y que no se tienen pérdidas de rendimiento al utilizar varias tarjetas GPU:

Velocidad final = 20.000 x 10 = 200.000 psw/s.

El tiempo total necesario para llevar a cabo el ataque de fuerza bruta sería de:

52 billones de contraseñas / 200.000 contraseñas por segundo = 8.24 años

Es decir, que aún siendo generosos en los cálculos el tiempo total es tan grande que no merece la pena realizar el ataque. Cuando esto ocurre se dice que el ataque es computacionalmente inviable.

Queda comprobado que un ataque por fuerza bruta no es lo más eficaz a la hora de atacar este tipo de archivos ya que podría pasar mucho tiempo antes de encontrar la contraseña correcta que podría ser algo tan simple como "*Welcome*" o "*Goodbye*".

Cuando esto ocurre se suele recurrir a la alternativa de utilizar ataques por diccionario que consiste en comprobar directamente multitud de contraseñas o palabras almacenadas en un archivo de texto. Este tipo de ataques no comprueba todas las posibilidades como en el caso de fuerza bruta por lo que al acabar el ataque puede que no se haya encontrada la contraseña. No obstante, un ataque por diccionario depende mucho de lo bueno que sea el diccionario en cada caso. Un diccionario es bueno o malo según lo acertado de su contenido con respecto al propietario del archivo, es decir, un diccionario con contraseñas en alemán no sería muy acertado para un usuario español a menos que se sepa que dicho usuario sabe alemán y lo utiliza en su día a día. Así pues, para solucionar este tipo de situaciones se generarán diccionarios personalizados cuyo contenido dependa del propietario del archivo o archivos que se pretenden descifrar.

La idea es crear una herramienta que analice y procese los datos de diferentes fuentes de información y genere automáticamente un diccionario personalizado para cada caso. Algunas posibles fuentes de información son:

- Datos personales del usuario y de su entorno.
- Archivos temporales de internet e información de los sitios que visita.
- Cuentas (Usuario/Contraseña) en páginas de internet.
- Comentarios o contenido en las redes sociales que el usuario utiliza.
- Documentos sin cifrar presentes en el mismo ordenador.
- Contraseñas de administrador o de usuario del sistema operativo.
- Memoria RAM del ordenador utilizado.
- Datos contenidos en la memoria de intercambio o SWAP del sistema operativo.
- Datos presentes en unidades de almacenamiento externas o dispositivos móviles.

Las palabras/contraseñas obtenidas tras procesar toda la información irán acompañadas de permutaciones, sustituciones, alteraciones de mayúsculas/minúsculas, inclusión de dígitos, combinación de palabras etc. Estas operaciones adicionales pueden realizarse a priori e incluir estas claves en el diccionario o pueden realizarse a posteriori con las

diferentes funciones que proporcionan las herramientas de descifrado analizadas en el presente proyecto.

La herramienta deberá incluir además la opción de dividir el diccionario en tantas partes como se desee permitiendo de esta forma realizar el ataque de manera distribuida (un diccionario por nodo) o de forma centralizada (todo el diccionario en un nodo de alto rendimiento).

El desarrollo de la herramienta de generación de diccionarios personalizados es una parte importante de la plataforma de descifrado de archivos y está siendo realizada en paralelo en otro proyecto de investigación dentro del marco de trabajo del grupo IUICP (Instituto Universitario de Investigación en Ciencias Policiales).

De ahora en adelante se supondrá que la plataforma funciona con diccionarios personalizados y se tendrá en cuenta en la realización de los apartados posteriores.

4

Software para la Identificación de Archivos Cifrados.

Uno de lo objetivos principales de este trabajo es la automatización del proceso de búsqueda e identificación de archivos cifrados en un determinado soporte digital. En esta capítulo vamos a realizar un análisis de las principales herramientas de libre distribución que pueden ser utilizados para este fin y vamos a comparar su rendimiento con el que ofrece la herramienta *Passware Kit Forensic*.

La mayoría de las herramientas comerciales de descifrado de archivos incluyen la función de buscar archivos protegidos tanto en el propio ordenador como en unidades de almacenamiento externas. Para la realización del proyecto se dispone de la licencia de uso de uno de estos programas comerciales llamado *Passware Kit Forensic.*

También existen herramientas gratuitas o de *software* libre que pueden explorar el disco duro y encontrar archivos cifrados aunque esta no sea su función principal. Por ejemplo el antivirus de *software* libre *ClamAV* y su correspondiente versión para Windows llamada *ClamWin* permiten analizar el disco en busca de amenazas para la seguridad del equipo incluyendo la posibilidad de marcar como amenazas los archivos cifrados.

Para poder comparar el estado en que se encuentra la función de búsqueda tanto del programa *Passware* como de la herramienta *ClamAV* se ha creado una base de datos con diferentes tipos de archivos cifrados. Esta base de datos contiene un total de 22 archivos de 14 tipos diferentes. Algunos tipos de archivo tienen duplicados para poder realizar pruebas con contraseñas de diferente longitud.

Tipos de archivos incluidos:

1) Archivos comprimidos RAR cifrados con AES sin cifrar los metadatos.
2) Archivos comprimidos RAR cifrados con AES y con la cabecera cifrada.
3) Archivos comprimidos y cifrados ZIP con una versión anterior a la 9.0, que utilizan el algoritmo de cifrado “Winzip Compatible 2.0”.
4) Archivos comprimidos ZIP cifrados con AES-256.
5) Archivos antiguos de Word, Excel y PowerPoint (ver. 1997-2003). Cifrados con RC4.

6) Archivos de Word, Excel y PowerPoint (ver. 2007 en adelante). Cifrados con AES-128.

7) Archivos de Word, Excel y PowerPoint creados con OpenOffice (.odt, .ods, .odp). Cifrados con AES.

8) Archivos PDF versión 5.0 cifrados con RC4[13].

A continuación se realizará una comparativa entre las herramientas de búsqueda de archivos cifrados que se ha indicado.

4.1 Comparativa *Passware* vs *ClamAV/ClamWin*

En este apartado realizaremos una comparativa entre las funciones de búsqueda de archivos cifrados que proporcionan las herramientas *Passware* y ClamAV/ClamWin.

4.1.1 Búsqueda de archivos cifrados con ClamAV/ClamWin

Tras realizar un escáner con esta herramienta en el directorio donde se encuentran los archivos cifrados se han obtenido los siguientes resultados:

```
Scan Started Tue Apr 17 15:55:42 2012
------------------------------------------------------------------------------

C:\Users\Javier\Desktop\TFC_BaseDatos\Archivos\CIfrado.zip: Heuristics.Encrypted.Zip FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\Archivos\CIfrado_no_contenido.rar: Heuristics.Encrypted.RAR FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\Archivos\CIfrado_y_contenido.rar: Heuristics.Encrypted.RAR FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\Archivos\Documento_cifrado.pdf: Heuristics.Encrypted.PDF FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\PdfCifrado\Fraxinus.pdf: Heuristics.Encrypted.PDF FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\RARcifrado\RARcifrado.rar: Heuristics.Encrypted.RAR FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\ZipCifrado\AES256.zip: Heuristics.Encrypted.Zip FOUND
C:\Users\Javier\Desktop\TFC_BaseDatos\ZipCifrado\ZipCrypto.zip: Heuristics.Encrypted.Zip FOUND
----------- SCAN SUMMARY -----------
Known viruses: 1196190
Engine version: 0.97.4
Scanned directories: 13
Scanned files: 22
Infected files: 8

Data scanned: 26.79 MB
Data read: 13.13 MB (ratio 2.04:1)
Time: 7.468 sec (0 m 7 s)

---------------------------------------
Completed
---------------------------------------
```

Fig. 4 – Resultado de la búsqueda de archivos cifrados realizada con ClamWin

Como se puede ver en la imagen de los 22 archivos tan solo se han encontrado 8 de los archivos cifrados. Esto se debe a que en la actual versión 0.97.4 el motor de búsqueda no incluye la definición de los archivos de *MicrosoftOffice* ni tampoco de los documentos de

[13] No se dispone de archivos PDF cifrados con AES pues para cifrar archivos PDF con este algoritmo se necesita el programa comercial *Adobe PDF Writter*.

OpenOffice. Todos los demás archivos ya sean documentos PDF, ZIP o RAR han sido detectados como cifrados.

La búsqueda se ha realizado tanto en Linux utilizando *ClamAV* (por consola) como en Windows con *ClamWin* (GUI) obteniéndose los mismos resultados con ambas versiones. Para lanzar la búsqueda es necesario activar la detección de archivos cifrados.

En Linux esto se hace así:

```
clamscan --block-encrypted=yes -r -l scan.txt
./Escritorio/TFC_BaseDatos/
```

En Windows se debe añadir la opción `--block-encrypted=yes` en la pestaña de opciones avanzadas de *ClamWin*.

4.1.2 Búsqueda de archivos cifrados con Passware

Tras realizar un escáner con la herramienta *Passware* en el directorio donde se encuentran los archivos cifrados se han obtenido los siguientes resultados:

File Name	Recovery Options	File Type	Document Type
CIfrado.zip	Brute-force - Fast, Instant Plaintext atta...	Archivo WinRAR ZIP	Zip 1.0
CIfrado_no_contenido.rar	Brute-force - Slow, Hardware accelerati...	Archivo WinRAR	RAR 3.0
CIfrado_y_contenido.rar	Brute-force - Slow, Hardware accelerati...	Archivo WinRAR	RAR 3.0
Documento_cifrado.pdf	Brute-force - Medium, File patching re...	Archivo PDF	Acrobat 5.0
MExcel2007.xlsx	Brute-force - Slow, Hardware accelerati...	Archivo XLSX	MS Office 2007
MExcel97-2003.xls	Brute-force - Fast, Instant Online Decry...	Archivo XLS	MS Excel 97-2003
MPowerPoint2007.pptx	Brute-force - Slow, Hardware accelerati...	Archivo PPTX	MS Office 2007
MPowerPoint97-2003.ppt	Brute-force - Medium	Archivo PPT	MS Powerpoint 2003
MWord2007.docx	Brute-force - Slow, Hardware accelerati...	Documento XML abierto de Offi...	MS Word 2007
MWord97-2003.doc	Brute-force - Fast, Instant Online Decry...	Archivo DOC	MS Word 97-2003
docx6l.docx	Brute-force - Slow, Hardware accelerati...	Documento XML abierto de Offi...	MS Word 2007
docxAES.docx	Brute-force - Slow, Hardware accelerati...	Documento XML abierto de Offi...	MS Word 2007
Fraxinus.pdf	Brute-force - Medium, File patching re...	Archivo PDF	Acrobat 5.0
RARcifrado.rar	Brute-force - Slow, Hardware accelerati...	Archivo WinRAR	RAR 3.0
docAES.doc	Brute-force - Fast, Instant Online Decry...	Archivo DOC	MS Word 97-2003
docRC4.doc	Brute-force - Fast, Instant Online Decry...	Archivo DOC	MS Word 97-2003
libroXLS.xls	Brute-force - Fast, Instant Online Decry...	Archivo XLS	MS Excel 97-2003
AES256.zip	Brute-force - Slow, Hardware accelerati...	Archivo WinRAR ZIP	Zip 2.0
ZipCrypto.zip	Brute-force - Fast, Instant Plaintext atta...	Archivo WinRAR ZIP	Zip 2.0

Fig. 5 – Resultado de la búsqueda de archivos cifrados realizada con *Passware Kit Forensic*

Como se puede ver de los 22 archivos cifrados *Passware* ha sido capaz de encontrar 19 de ellos además de aportar información útil para el posible descifrado de estos archivos. Entre los archivos detectados solo faltan aquellos creados con *OpenOffice* mientras que los documentos creados con *Microsoft Office*, los archivos PDF y los archivos comprimidos ZIP y RAR han sido todos detectados como cifrados. Además la información que aporta el programa sobre los archivos y su seguridad es de gran utilidad. Al guardar el informe de resultados se obtiene un documento de texto con toda la información relativa a los archivos.

Para ver qué tipo de información se proporciona se pone como ejemplo el archivo de Excel creado con la versión antigua de *Microsoft Office*:

```
Nombre del archive: MExcel97-2003.xls
Directorio: C:\Users\Javier\Desktop\TFC_BaseDatos\Archivos\
Opciones de recuperación: Brute-force - Fast, Instant Online
Decryption possible
Tipo de archivo: Archivo XLS
Tipo de documento: MS Excel 97-2003
Flags de protección: Open Password
Fecha de modificación: 20/09/2011 11:30
Tamaño: 18 KB
Hash MD5: 8BDA5CC847A36D46F35B16CC59A822F0
```

4.1.3 Conclusión

Tras comparar los resultados obtenidos con *Passware* frente a los obtenidos con ClamAV/ClamWin se puede ver como la herramienta comercial *Passware* obtiene mejores resultados al detectar una mayor cantidad de archivos y aportar información sobre su seguridad. No obstante la lista de desarrolladores de ClamAV ya ha recibido peticiones[14] para que en su siguiente versión se incluya la detección de archivos tanto de Open Office como de Microsoft.

[14] http://www.gossamer-threads.com/lists/clamav/users/53192?search_string=Office%20encrypted;#53192

5

Análisis de Herramientas de Libre Distribución para el Descifrado de Archivos

Dentro del proceso de estudio de evidencias digitales en una investigación, una vez que se han determinado los archivos y unidades potencialmente protegidos junto con la protección empleada, el siguiente paso consiste en intentar revertir esa protección. En la mayor parte de los casos, desconoceremos la clave o contraseña empleada por lo que será necesario la utilización de herramientas específicas que permitan evadir o romper los mecanismos de seguridad empleados.

En la actualidad existe una gran cantidad de programas comerciales que sirven para descifrar archivos cuando desconocemos la clave empleada para dicho cifrado, sin embargo, estos programas tienen dos inconvenientes asociados. El primero de ellos es el alto coste que suponen las licencias de uso de este tipo de herramientas que suele ser del orden de varios miles de euros. El segundo inconveniente que presentan se debe a la dificultad de automatizar los ataques de descifrado de una manera sencilla como por ejemplo utilizando la línea de órdenes. Al ser aplicaciones comerciales desarrolladas para plataformas Windows suelen implementar su uso a través de una interfaz gráfica de usuario y no suelen incluir la opción de configurar y/o de ejecutar los ataques desde la consola. Se pretende por lo tanto conseguir dos objetivos, el primero de ellos es encontrar herramientas alternativas de descifrado de archivos que no sean comerciales, y el segundo es tratar de buscar en la medida de lo posible aquellas herramientas que puedan ser ejecutadas desde la línea de órdenes.

Para que las herramientas encontradas constituyan una alternativa viable deberán mostrar una eficiencia similar al programa comercial del que disponemos en el proyecto, es decir, la herramienta de descifrado *Passware Kit Forensic*. El siguiente apartado presenta un análisis del rendimiento del programa comercial *Passware* que nos permitirá compararlo posteriormente con las distintas herramientas de software libre que se encuentren.

5.1 Rendimiento de *Passware* frente a diferentes documentos

Para mostrar el rendimiento del programa *Passware* se han realizado una serie de ataques a diferentes tipos de archivo utilizando un PC con procesador *Intel Quad Core CPU Q6600* a 2.4 GHz y una tarjeta gráfica *nVIDIA Geforce GTX 460* con 1GB de memoria y 336 núcleos CUDA.

Los resultados se muestran en la siguiente tabla indicando además el tipo de cifrado de cada uno de los archivos atacados:

Tipo de Ataque	**Archivo .docx (AES-128)**	**Arhivo .doc (Cifrado RC4)**	**Archivo .zip (Cifrado ZIP)**	**Archivo .zip (AES-256)**	**Archivo .rar (AES-128)**
Ataque por fuerza bruta	6.115 psw/s	1.192.700 psw/s	66.471.200 psw/s	18.461 psw/s	2.400 psw/s
Ataque por diccionario	6.103 psw/s	1.102.076 psw/s	8.800.300 psw/s	18.275 psw/s	1.980 psw/s
¿Se activa la utilización de GPU?	Sí	No	No	Sí	Sí

Teniendo en cuenta los resultados se pueden obtener las siguientes conclusiones:

- Se puede ver como la tarjeta gráfica GPU no se activa al intentar descifrar archivos protegidos con algoritmos obsoletos como son RC4 o el cifrado que implementaba ZIP en sus primeras versiones. Esto se debe a que estos algoritmos son tan débiles que no se ha implementado la aceleración GPU para su descifrado por no ser necesaria.
- En los casos en los que se activa la GPU se puede observar que la velocidad del ataque no varía en función del tipo de ataque realizado. Esto es importante, pues al realizar ataques por diccionario se deben realizar accesos al disco que pueden limitar la velocidad del ataque. *Passware*, no obstante, ha sido capaz de implementar este tipo de ataques utilizando la GPU sin pérdidas significativas de rendimiento. Para más detalles sobre la utilización de GPU en ataques por diccionario consultar el apartado “5.3 Nota sobre la utilización de la GPU en ataques por diccionario”.
- La velocidad en claves/segundo es diferente para cada uno de los archivos cifrados con AES. Esto es debido a que cada uno de ellos implementa el cifrado de una manera diferente y utilizan una función de derivación de clave con distinto número de iteraciones. Esto se explica en más detalle en el apartado 3.
- Se puede ver que el archivo más robusto de todos es el documento cifrado RAR pues gracias a la lentitud de su función de derivación se obtienen velocidades de ataque muy bajas.

5.2 Herramientas no comerciales

Una vez visto el rendimiento de un programa comercial de descifrado se pueden buscar alternativas no comerciales que realicen las mismas tareas. Existen multitud de proyectos de *software* libre dedicados al descifrado de archivos que pueden encontrarse a través de internet, sin embargo, todos ellos son abandonados tarde o temprano debido a que las aplicaciones comerciales llevan mucha ventaja en este terreno y se necesita mucho tiempo para desarrollar un software eficiente capaz de realizar los mismos ataques que una aplicación de pago. También se debe tener en cuenta que pueden existir vulnerabilidades en la implementación de los algoritmos de cifrado y son los equipos de trabajo de las herramientas comerciales los primeros en explotar dichas vulnerabilidades para sacar beneficio de ello.

A continuación se explican las herramientas de descifrado encontradas para cada uno de los distintos tipos de archivos de interés comparando el rendimiento de estas herramientas con el del programa *Passware* e indicando si sirven o no como alternativas gratuitas.

5.2.1 Archivos .doc y .xls

5.2.1.1 Herramienta *Free Word Excel Password*

Para documentos de Word y Excel anteriores a la versión 2007 no se ha encontrado ninguna herramienta de descifrado desarrollada para el sistema operativo Linux. Para plataformas Windows existe un programa gratuito llamado "*Free Word Excel Password*[15]" que sirve para descifrar archivos con extensión .doc y .xls.

Esta herramienta permite realizar ataques por fuerza bruta y ataques por diccionario utilizando únicamente la CPU. Al igual que en el caso de *Passware* no se utiliza la GPU para acelerar el proceso.

Al ser una herramienta desarrollada para Windows la configuración de los ataques se realiza a través de su interfaz gráfica sin incluir la opción de utilizar la línea de órdenes para su uso.

[15] Más información sobre la herramienta y acceso a su descarga en la web:

http://www.freewordexcelpassword.com/

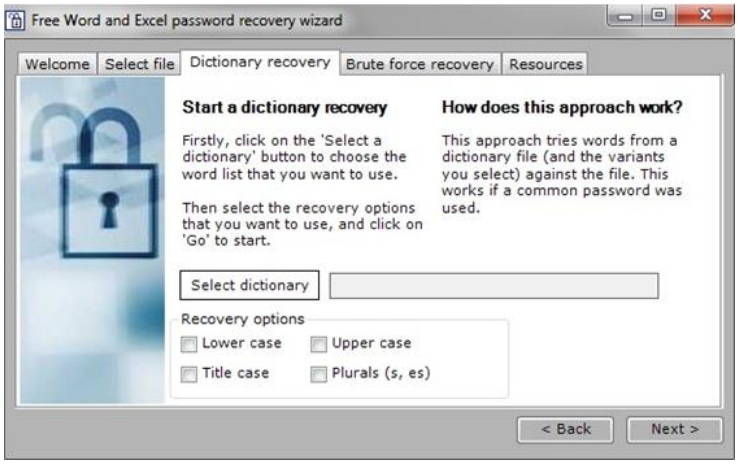


Fig. 6 – Interfaz gráfica del programa de descifrado Free Word Excel Password

A la hora de analizar su eficiencia nos encontramos con el problema de que no muestra la velocidad con la que se realiza el ataque, sino que se muestra el estado del ataque mediante una barra de progreso. Por lo tanto, para poder comparar esta herramienta con *Passware* se ha realizado un ataque de fuerza bruta de 5 caracteres y anotado el tiempo total necesario para llevarlo a cabo. De esta manera se puede tener una idea orientativa de la eficiencia del programa.

Las pruebas siguientes han sido realizadas en un PC Intel Core i7 a 2.2 GHz modelo 2670QM.

Ataque por fuerza bruta:

Programa/tiempo	Archivo .Doc	Archivo .Xls
Passware	26 segundos	26 segundos
Free Word Excel Password	1 min 47 segundos	1 min 51 segundos

Como se puede ver la eficiencia del programa gratuito deja mucho que desear frente al programa comercial *Passware* que es capaz de conseguir un tiempo hasta cuatro veces mejor.

No obstante es necesario comprobar también el rendimiento que se obtiene cuando se realizan ataques por diccionario ya que la eficiencia del código puede variar entre los diferentes tipos de ataque.

Como la herramienta gratuita "*Free Word Excel Password*" no muestra la velocidad del ataque para comparar la eficiencia entre los dos programas se ha anotado el tiempo total necesario para completar un ataque con un diccionario de 101Mb de tamaño:

Ataque por diccionario (101Mb):

Programa/tiempo	Archivo .Doc	Archivo .Xls
Passware	12 segundos	12 segundos
Free Word Excel Password	52 segundos	54 segundos

Como cabía esperar, la herramienta comercial *Passware* finaliza la ejecución del ataque en menor cantidad de tiempo.

5.2.1.2 Servicios de descifrado online

Es importante mencionar que para este tipo de archivos cifrados existen empresas que ofrecen servicios online de descifrado a un precio asequible y de manera prácticamente instantánea. Por ejemplo en la página de la empresa *Decryptum* se ofrece la posibilidad de descifrar gratuitamente una parte del archivo y obtener una vista preliminar de su contenido. Si se desea descifrar completamente el archivo se debe pagar el servicio cuyo precio ronda los 25€ pudiendo obtener descuentos si se desea realizar más de un descifrado.

Veamos un ejemplo de su utilización para comprobar su eficiencia:

1) En primer lugar se ha cifrado el contenido de un archivo de Word con la contraseña "docRC4" que contiene tanto letras minúsculas como mayúsculas y números.
2) Accediendo a la página web http://www.decryptum.com/ se selecciona *"Start Decryption"*. A continuación se deben aceptar los términos de uso y cargar el archivo a descifrar.
3) Tras pulsar en "siguiente" da comienzo el descifrado que tras **15 segundos** aproximadamente finaliza mostrando parte del contenido del documento:

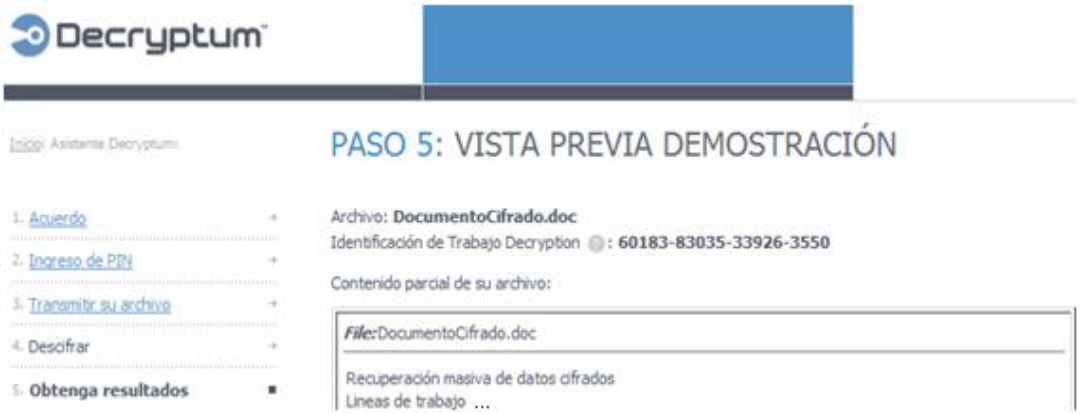


Fig. 7- Descifrado de un documento de Word en Decryptum.com

Viendo el ejemplo se puede ver como el servicio ofrecido por *Decryptum* es capaz de descifrar el documento en tan sólo 15 segundos sin tener en cuenta la longitud de la contraseña utilizada pues se realiza el ataque de fuerza bruta directamente en la clave de 40 bits del algoritmo RC4.

5.2.1.3 Conclusión archivos .doc/.xls

Sólo se ha encontrado una herramienta gratuita para romper la seguridad de los archivos de Word y Excel con versiones anteriores al año 2007. Además su eficiencia es menor que la del programa comercial *Passware* y no es posible utilizar la línea de órdenes para automatizar los ataques. Sin embargo, hay que tener en cuenta que este tipo de archivos se encuentra cada vez con menos frecuencia en los ordenadores de los usuarios y como se ha visto existe la opción de descifrarlos online de manera prácticamente instantánea.

Por lo tanto, no se debería condicionar el uso del programa comercial por el hecho de no tener una herramienta alternativa para este tipo de archivos ya que si se encontrasen programas para descifrar los demás tipos de archivos se podría prescindir de la herramienta comercial y en los casos necesarios utilizar los servicios de recuperación online.

5.2.2 Archivos ZIP

En el caso de los archivos cifrados ZIP existen varias herramientas tanto para Linux como para Windows que permiten obtener la contraseña de los archivos cifrados ZIP. Sin embargo, todas estas solo sirven para descifrar las antiguas versiones que utilizaban el algoritmo "*WinZip Compatible 2.0*". No se ha encontrado ninguna herramienta que sea capaz de romper la actual seguridad utilizada por ZIP (basada en el algoritmo AES).

A continuación se detallan unas cuantas de las herramientas encontradas para descifrar archivos ZIP ordenadas por la fecha de su última actualización:

1. **Picozip**[16]**:** es una herramienta gratuita para Windows que permite realizar ataques por fuerza bruta y por diccionario además de incluir la posibilidad de lanzar los ataques desde la línea de órdenes. El problema de *Picozip* es que es un proyecto abandonado hace tiempo y su última actualización fue en el año 2006.
2. **Fcrackzip**[17]**:** es una herramienta de *software* libre desarrollada para Linux que permite descifrar archivos comprimidos Zip con versiones anteriores a WinZip 9.0. Esta herramienta implementa diferentes tipos de ataque y ha sido optimizada para varias arquitecturas. No obstante, parece ser que está bastante anticuada ya que la última actualización es del año 2008.
3. **ZipCrack**[18]**:** es una herramienta desarrollada para Linux que permite lanzar por consola ataques de fuerza y ataques por diccionario contra archivos cifrados ZIP. Si el ataque tiene éxito la contraseña se mostrará por pantalla y se descomprimirá el archivo automáticamente. Este programa ha sido actualizado más recientemente que los anteriores, la última actualización es de Febrero de 2011.

5.2.2.1 Análisis de *hashkill*

Al igual que estas tres herramientas hay muchas otras que realizan las mismas funcionas sin aportar nada que las diferencie de las demás. No obstante, de entre todas las

[16] Página oficial del proyecto picozip: http://picozip.com/

[17] El programa fcrackzip puede descargarse desde la página web del autor: http://oldhome.schmorp.de/marc/fcrackzip.html

[18] Zipcrack puede obtenerse desde el sitio web de sourceforge: http://sourceforge.net/projects/zipcrack/

herramientas encontradas para descifrar archivos ZIP existe una que aporta un valor añadido.

Hashkill es una herramienta de línea de órdenes desarrollada para Linux cuya principal utilidad es realizar ataques de fuerza bruta o por diccionario para la ruptura de funciones hash utilizando la aceleración GPU. Sin embargo, esta herramienta incluye también la posibilidad de descifrar archivos ZIP.

Si se echa un vistazo al análisis de rendimiento del programa *Passware* (Apartado "5.1 Rendimiento de *Passware* frente a diferentes documentos") se puede ver como al descifrar archivos ZIP protegidos con el antiguo algoritmo de cifrado no se activa la utilización de la GPU, es decir, que *Passware* no utiliza la GPU para este tipo de archivos mientras que Hashkill en principio sí, por lo que cabe esperar que su rendimiento sea mejor.

Para comprobarlo se debe descargar la herramienta Hashkill desde su página oficial[19] y lanzar un ataque de fuerza bruta contra un archivo cifrado ZIP (soporta tanto las versiones antiguas como las nuevas).

Orden para utilizar la GPU: `./hashkill-gpu -pzip -b -f prueba.zip`

Orden para utilizar solo la CPU: `./hashkill-cpu -pzip -b -f prueba.zip`

La siguiente tabla muestra los resultados obtenidos tras realizar las pruebas en un PC *Intel Quad Core CPU Q6600* a 2.4 GHz y la tarjeta gráfica *nVIDIA Geforce GTX 460*:

Programa	Hashkill-cpu	Hashkill-gpu	Passware
Velocidad del ataque	3.400.000 psw/s	3.400.000 psw/s	66.000.000 psw/s

En la tabla de resultados se puede observar que aunque se utilice el comando *hashkill-gpu* el rendimiento sigue siendo el mismo que cuando se utiliza el comando *hashkill-cpu* y además mucho menor que el rendimiento de *Passware*. Esto parece indicar que Hashkill no utiliza la GPU en el caso de los archivos ZIP. Para descartar que esta situación no se debe a una mala instalación de los drivers de nVIDIA (Linux) se han realizado ataques a diferentes funciones hash para comprobar si existe una mejora de rendimiento. Por ejemplo en el caso de un hash SHA-1 se obtienen los siguientes resultados:

Comando con Hash SHA1:

```
./hashkill-gpu  -psha1 -b 711383a59fda05336fd2ccf70c8059d1523eb41a
```

[19] Página oficial de *Hashkill*: http://www.gat3way.eu/hashkill/index.php

Resultados:

	Hashkill-cpu	Hashkill-gpu
Velocidad del ataque	27.100.000 hashes/s	291.000.000 hashes/s

Como se puede ver la GPU sí que se utiliza al atacar funciones *hash* con *Hashkill* por lo que queda claro que solo en el caso de los archivos ZIP la GPU permanece inactiva.

Este problema queda aclarado tras consultar el foro de Hashkill en el que se indica que la versión actual (0.2.4) estaba a mitad de camino en la implementación de ataques con la GPU. La buena noticia es que para verano del 2012 Hashkill presentará su versión 0.3.0 que incluirá entre otras cosas "GPU password recovery for ZIP, RAR and 7Z files"[20] convirtiéndose así en el primer proyecto *opensource* que utiliza la GPU para atacar estos archivos.

5.2.2.2 Conclusión archivos ZIP

Como ya se comentó anteriormente no se ha encontrado ninguna herramienta gratuita o de *software* libre que permita atacar la seguridad de los archivos Zip protegidos con AES. Esto hace que siga siendo necesaria la utilización del programa comercial *Passware* para el descifrado de archivos protegidos Zip. Sin embargo, *Hashkill* es una herramienta prometedora que cada vez tiene más apoyos y que incluirá en su próxima versión tanto ataques contra archivos Zip como contra archivos RAR utilizando la GPU para dicha tarea.

5.2.3 Archivos RAR

En el caso de los archivos RAR, se utiliza un mecanismo de cifrado muy fuerte (algoritmo AES y una función de derivación de clave especialmente lenta) por lo que es necesario buscar herramientas que sean capaces de utilizar la aceleración GPU para el descifrado de archivos.

Algunas herramientas de utilidad para la recuperación de contraseñas en archivos RAR se detallan a continuación:

1. **UnrarExtract[21]:** es un proyecto gratuito y *open-source* recientemente actualizado que permite descomprimir archivos RAR desde la línea de órdenes. Aunque este es un programa para la descompresión de archivos, puede servir para realizar ataques de diccionario. Esto es posible debido a que el programa es capaz de utilizar una lista de palabras para tratar de abrir el archivo RAR en el caso de que

[20] Información sobre la próxima versión de *Hashkill*:

http://www.gat3way.eu/forum/viewtopic.php?f=4&t=75

[21] Página del proyecto UnrarExtract: http://www.nvglabs.com/

esté protegido con contraseña. La herramienta en sí solo utiliza la CPU para realizar esta tarea por lo que sí el diccionario de palabras a probar es de gran tamaño el ataque conllevaría demasiado tiempo. Por todo esto, el programa *UnrarExtract* no sería una buena alternativa al programa comercial *Passware*.

2. **Igrargpu[22]:** es un proyecto que comenzó siendo gratuito y gracias al alto rendimiento que obtenía utilizando la GPU pasó a formar parte del *software* comercial *Accent Rar Password Recovery[23]*. No obstante, aunque el programa pasó a ser comercial en Diciembre del 2010, la versión desarrollada hasta ese momento todavía está disponible para su descarga y utilización de forma gratuita.

5.2.3.1 Estudio y Evaluación de la herramienta cRARk

A diferencia del programa *Igrargpu,* existe otra herramienta también de alto rendimiento que tiene una versión comercial pero cuya versión gratuita llamada "cRARk" sigue siendo actualizada frecuentemente. Las características de esta herramienta hacen que merezca la pena analizarla en detalle y comparar su rendimiento con el programa comercial *Passware*.

La herramienta *cRARk* ha sido diseñada única y exclusivamente para recuperar la contraseña de archivos cifrados Winrar mediante ataques de fuerza bruta o ataques por diccionario. Este programa está disponible tanto para plataformas Linux, como para Windows y Mac OS aunque el rendimiento de la versión para Linux es ligeramente superior. Para configurar los ataques la herramienta utiliza la librería PCL[24] que permite definir las reglas de generación de claves que se utilizarán al atacar un archivo.

- El programa comercial *Parallel Password Recovery[25]* posee la licencia de utilización del motor cRARk para sus propios productos y además utiliza la última versión de la librería PCL. Algunas de las características que no ofrece *cRARk* y sí ofrece *Parallel* se listan a continuación:
- Interfaz gráfica de usuario (GUI)
- *Password Definition Master*
- Soporte para *multicores/multiprocessors*
- Pausa/resumen del proceso de recuperación

[22] La herramienta Igrargpu puede obtenerse desde la página Web del autor:

http://www.golubev.com/rargpu.htm

[23] *Accent Rar Password Recovery* es una herramienta professional de recuperación de contraseñas de archivos RAR. Web de la empresa: http://passwordrecoverytools.com

[24] PCL o *Password Cracking Library* es una potente herramienta que permite definir una serie de reglas para la generación de claves. La librería PCL 2.0 utilizada por cRARk se distribuye con licencia *freeware* o gratuita. La licencia es de pago a partir de la versión 3.0.

[25] Página web de la herramienta *Parallel Password Recovery:* http://www.parallelrecovery.com/

- Nuevas funciones de la librería PCL (versión 3.0 frente a la 2.0 utilizada por cRARk).

Como se puede ver en la lista, la herramienta cRARk no soporta varios núcleos, sin embargo, es capaz de utilizar eficientemente los núcleos GPU de las tarjetas gráficas tanto de AMD como de nVIDIA. Lo bueno de la herramienta cRARk es que sigue siendo desarrollada y actualizada al mismo tiempo que su versión comercial.

Para realizar un análisis del rendimiento de este programa es necesario conocer la sintaxis tanto de la librería PCL como de los argumentos que se le pueden pasar por consola. Una descripción de esta sintaxis puede encontrarse en el manual del programa[26].

Para realizar los ataques se debe configurar el archivo *password.def* indicando las reglas de generación de claves.

Ejemplo de configuración de ataques por fuerza bruta:

Claves formadas solo por letras minúsculas con cualquier longitud:

```
# using only lower-case Latin letters.
$a *
```

Claves formadas por letras minúsculas, mayúsculas y números con cualquier longitud:

```
# The same as the above, but using both cases Latin letter and digits
[$a $A $1] *
```

Ejemplo de configuración de ataques por diccionario:

```
#Dictionary definition - only if $u and $w will be used
$w = "C:\\Users\\Javier\\Desktop\\TFC\\CRARK\\main.dic"
##
# Here password definitions begins.
$w
```

Es importante remarcar la diferencia existente entre la sintaxis de los dos tipos de ataque. En los ataques por fuerza bruta se utiliza el carácter '*' que indica la repetición del patrón anterior, es decir, `$a` indica una letra minúscula del alfabeto y `$a*` generará palabras cada vez de mayor longitud combinando todas las letras minúsculas. Según se indica en el manual del programa la utilización de la GPU está vinculada a la existencia del carácter '*' en el archivo de configuración. Teniendo esto en cuenta, podemos saber a priori que cRARk no utiliza la GPU al realizar ataques por diccionario y si la utiliza para realizar ataques por fuerza bruta.

A continuación se muestra una comparativa de rendimiento entre cRARk y *Passware* utilizando ambos tipos de ataque. Las pruebas han sido realizadas en un PC con

[26] Manual del programa cRARk: http://www.crark.net/cRARk.html

procesador *Intel Quad Core CPU Q6600* a 2.4 GHz y una tarjeta gráfica *nVIDIA Geforce GTX 460* con 1GB de memoria y 336 núcleos CUDA.

Comparativa de ataque por diccionario (Tamaño del diccionario 414Kb):

	cRARk	**Passware**
Velocidad ataque por diccionario	75 psw/s	900 - 1000 psw/s
¿Se activa la GPU?	No	Sí
Tiempo total	13 minutos	2:27

Se puede ver como la velocidad del programa *Passware* es claramente mayor ya que es capaz de utilizar la GPU al realizar ataques por diccionario.

Comparativa ataque de fuerza bruta (Todos los caracteres ASCII sin límite de longitud):

	cRARk	**Passware**
Velocidad ataque de fuerza bruta	3800 – 4200 psw/s	1900 – 2400 psw/s
¿Se activa la GPU?	Sí	Sí

Como se puede ver en la tabla la velocidad del ataque realizado por cRARk es mayor llegando incluso a doblar el rendimiento de *Passware* y eso teniendo en cuenta que cRARk utiliza para el ataque solo un procesador y la GPU mientras que *Passware* utiliza además de la GPU los cuatro procesadores disponibles.

Viendo que el código de cRARk es más eficiente que el de *Passware* conviene comprobar si su versión comercial (*Parallel Password Recovery*) es capaz de utilizar la GPU en ataques por diccionario. Tras contactar por correo electrónico con el servicio de soporte de *Parallel Password Recovery* se sabe que actualmente su *software* no utiliza la GPU en los ataques por diccionario.

5.2.3.2 Conclusión archivos RAR

Teniendo en cuenta los resultados obtenidos puede decirse que en el caso de querer realizar ataques por fuerza bruta cRARk sirve como alternativa de *Passware* mientras que en el caso de querer realizar ataques por diccionario sigue siendo necesaria la utilización de esta herramienta comercial. No obstante, hay que tener en cuenta que el motor de descifrado de cRARk está incluido en el programa comercial *Parallel Password Recovery* y si éste quiere estar a la altura de sus competidores deberá implementar el uso de la GPU en ataques por diccionario. Por ello cabe esperar que en futuras versiones cRARk incluya esta característica lo que permitirá utilizar esta herramienta como sustituto del programa *Passware* en cuanto a los archivos RAR se refiere.

Además se debe tener en cuenta que el proyecto de código libre *Hashkill* incluirá en su siguiente versión soporte para este tipo de archivos incluyendo la utilización de la GPU.

5.2.4 Archivos PDF

En el caso de archivos PDF hay que distinguir dos tipos de protección:

En primer lugar está la contraseña denominada "*owner password"* que evita copiar, imprimir y seleccionar texto del documento PDF pero no cifra el contenido por lo que puede ser eliminada con tan solo editar la cabecera del pdf. Para eliminar esta protección existen multitud de utilidades, sin embargo, no son de interés para el proyecto ya que el contenido en sí sigue siendo legible.

El otro tipo de protección se denomina "o*pen password"* o "u*ser password"*. Este tipo de protección cifra el contenido del documento evitando su lectura. Desafortunadamente no se ha encontrado ninguna herramienta gratuita que permita descifrar documentos PDF.

Por lo tanto, para este tipo de archivos es necesario utilizar una herramienta comercial.

5.2.5 Archivos Office 2007 y posterior

Para los documentos de Office 2007 en adelante no existe ninguna herramienta gratuita o de *software* libre que permita descifrar el contenido de los documentos. De hecho, las herramientas comerciales y las empresas de seguridad como *Decryptum* no aseguran el descifrado de este tipo de archivos si no se indica alguna información adicional sobre la contraseña.

Para los archivos de Office 2007 en adelante es estrictamente necesario utilizar una aplicación comercial.

5.3 Nota sobre la utilización de la GPU en ataques por diccionario

La utilización de la GPU al realizar ataques por diccionario es uno de los aspectos clave para este proyecto ya que permite abordar más eficientemente la estrategia de diccionarios personalizados explicada en el apartado 3.6**.** Por ello es importante explicar una de las limitaciones que presenta la aceleración GPU en este tipo de ataques.

En el apartado 5.1 se vio como el programa *Passware* no utilizaba la GPU al realizar todos los ataques por diccionario sino que en algunos de ellos se utilizaba únicamente la CPU (como en el caso de documentos de Office con versión anterior al año 2007 o archivos comprimidos Zip con versión anterior a la 9.0). Además de *Passware*, otras herramientas conocidas como *Accent Password Recovery* o las herramientas de recuperación de archivos de *Elcomsoft* tampoco utilizan la GPU en estos casos lo que da a entender que para este tipo de archivos no es posible utilizar aceleración GPU. Para ratificar esto se ha contactado con los servicios técnicos de las diferentes herramientas obteniéndose respuesta por parte de *Accent Password Recovery*:

```
AccentOPR using GPU acceleration for all attacks for documents with
strong Office 2007/2010 protection. Attacks for documents saved as
"97-2000-XP-2003" (both brute-force and dictionary based ones)
cannot be accelerated with GPU. We're recommend you to download and
try demo version first (which is freely available at our website)
to be sure that everything works as expected before purchasing.
```

El porqué en estos casos no se puede utilizar la GPU no se explica en ninguna de las páginas oficiales de las herramientas de seguridad mencionadas. Sin embargo, se ha encontrado información[27] proporcionada por la herramienta de competición de ruptura de *hashes OCLhashcat* que permite entender los motivos.

El autor de *OCLHashcat* explica que en el caso de algoritmos *hash* rápidos como son MD4, MD5 o NTLM utilizar ataques por diccionario con la GPU no es eficiente ya que se tarda más tiempo en copiar el diccionario en la memoria de la GPU que atacarlos directamente con la CPU. Por otro lado en el caso de algoritmos *hash* lentos como *md5crypt* que utiliza 1000 iteraciones o WPA/WPA2 que realiza unas 16.000 iteraciones el uso de la GPU acelera considerablemente el ataque ya que mientras se realizan las iteraciones se puede copiar la siguiente lista de palabras del diccionario.

Extrapolando la información ofrecida por *OCLHashcat* al caso de archivos cifrados se puede entender por qué los programas comerciales no utilizan la GPU al atacar la seguridad de archivos como DOC, XLS o ZIP (cifrado 2.0). En estos archivos los algoritmos de seguridad son tan débiles que conlleva más tiempo transferir los datos del diccionario a la memoria global de la tarjeta gráfica que atacarlos directamente con la CPU.

5.4 Conclusión

Considerando el análisis realizado sobre las herramientas de libre distribución podemos decir que el uso de una herramienta comercial de descifrado de archivos sigue siendo indispensable sobre todo en el caso de archivos PDF o archivos de Office 2007 en adelante. Debido a esto será necesario automatizar el uso del programa comercial *Passware* para que pueda utilizarse a través de la línea de órdenes.

Antes de detallar el proceso de automatización de *Passware* analizaremos una serie de herramientas para la ruptura de funciones *hash*. Aunque estas herramientas no se encuentran entre los principales objetivos del proyecto, es interesante incluir información sobre ellas ya que serán de utilidad para el desarrollo de la "herramienta de creación de diccionarios personalizados". Esta herramienta como ya se comentó en el apartado 3.6, está siendo realizada en paralelo en colaboración con el presente proyecto. Ambos proyectos permitirán la construcción de la plataforma de procesado masivo de información cifrada detallada en los objetivos del proyecto.

[27] Información sobre el uso de la GPU: http://hashcat.net/wiki/oclhashcat_plus

6

Análisis de Herramientas para la Ruptura de Funciones *Hash*

En el siguiente apartado se presentan algunas herramientas especializadas en la ruptura de funciones hash.

De todas las herramientas de ruptura de hashes que se pueden encontrar, hay dos que destacan especialmente por encima de las demás. Estas herramientas son *Hashkill*[28] y *OCLHashcat*[29]. El motivo de su éxito viene dado por la utilización de las tarjetas gráficas como medio para acelerar el tiempo de ejecución de sus ataques.

No se pretende realizar una comparativa en profundidad de estas herramientas sino simplemente presentarlas y mencionar algunas de las características más importantes que poseen para que puedan ser utilizadas en la creación de los diccionarios personalizados.

6.1 Hashkill

Hashkill es una herramienta de *software* libre para la recuperación de contraseñas bajo el sistema operativo Linux. En su versión actual sólo permite realizar ataques a funciones *hash* aunque a largo plazo la herramienta incluirá la opción de descifrar cualquier tipo de archivo de los listados en el apartado 3 del proyecto. De hecho, su siguiente versión prevista para verano de 2012 incluirá ya la opción de atacar archivos cifrados ZIP, RAR y 7zip utilizando la GPU. Algunas de las características actuales que presenta Hashkill se enumeran a continuación:

[28] La herramienta hashkill puede obtenerse desde la página web del autor:

http://www.gat3way.eu/hashkill/index.php

[29] OclHashcat es una herramienta de libre distribución diseñada para eventos de competición de ruptura de funciones hash. Las diferentes versiones disponibles pueden descargarse desde su web:

http://hashcat.net/oclhashcat/

- Multi-hilo, aumenta el rendimiento en sistemas multi-core/multi-CPU.
- Algoritmos SSE2-acelerados para alcanzar velocidades superiores en CPUs x86 modernas.
- Cuatro tipos de ataque: diccionario/fuerza-bruta/hibrido/markov
- 35 *plugins* para diferentes tipos de contraseña.
- Soporte para guardar sesión/reanudar. Las sesiones se auto-guardan cada 3 segundos. El ataque puede reanudarse en los siguientes casos *"stopped/killed/system crashes/power down/etc*".
- Soporte para ataques Multi-hash (la lista puede tener una longitud de de hasta un millón de hashes).
- Soporte para tarjetas GPU rápidas tanto nVIDIA como ATI (4xxx, 5xxx. 6xxx).
- Soporte Multi-GPU.

Como se puede ver *Hashkill* es una herramienta bastante completa que sigue desarrollándose activamente.

A continuación se muestran algunos ejemplos de utilización y los resultados obtenidos:

```
./hashkill-gpu  -G2 -pmd5  -b:8:8:ascii -f hashMD5.txt
./hashkill-gpu  -G2 -psha1 -b:8:8:ascii -f hashSHA1.txt
./hashkill-gpu  -G2 -pntlm -b:8:8:ascii -f hashNTLM.txt
```

	MD5	SHA1	NTLM
Hashkill	1050 MH/s	223 MH/s	20 MH/s

6.2 OCLHashcat-plus y OCLHashcat-lite

El grupo de herramientas gratuitas *Hashcat*[30] ha sido diseñado especialmente para eventos de competición de ruptura de hashes y cada una de sus versiones presume de ser la mejor y más rápida en su campo. Actualmente sus mejores versiones son *OCLHashcat-plus* y *OCLHashcat-lite*[31] ya que ambas son herramientas basadas en procesado GPU. La herramienta "Lite" esta optimizada para la ruptura de hashes individuales y hashes rápidos

[30] Hashcat es la herramienta original basada solo en ataques con CPU. Se creó una versión paralela que sí utilizaba la GPU llamada OCLHashcat. Esta a su vez fue sustituida por las versiones "Plus" y "Lite" de la herramienta optimizando el código en función del tipo de hash a atacar.

[31] Estas dos versiones de la herramienta OCLHashcat pueden obtenerse desde su página web:

Versión plus: http://hashcat.net/oclhashcat-plus/

Versión lite: http://hashcat.net/oclhashcat-lite/

como pueden ser NTLM, MD5, SHA1… mientras que la herramienta "Plus" esta optimizada para realizar la ruptura de múltiples hashes en paralelo o hashes modernos con un alto número de iteraciones como pueden ser md5crypt, phpass, WPA2 etc.

Es importante mencionar que estas herramientas no planean incluir soporte para el descifrado de archivos ya que al ser herramientas utilizadas en competiciones de ruptura de hashes no tienen ningún interés[32] en implementar ataques contra archivos comprimidos, archivos PDF o documentos de Office.

6.2.1 Características de *OCLHashcat*

Con la información que se ofrece en su página web se ha rellenado la siguiente tabla que muestra las características comunes a las dos herramientas y las diferencias principales entre ellas:

Common features	***Exclusive features Plus version***	***Exclusive features Lite version***
Multi-GPU (up to 16 gpus)	*World's fastest md5crypt, phpass, mscash2 and WPA / WPA2 cracker*	*World's fastest NTLM, MD5, SHA1, SHA256, descrypt and SL3 cracker*
Multi-OS (Linux & Windows native binaries)	*World's first and only GPGPU based rule engine*	*Supports mask attack*
Multi-Platform (OpenCL & CUDA support)	*Multi-Hash (up to 24 million hashes)*	*Supports distributed cracking*
Multi-Algoritmo	*Focuses highly iterated, modern hashes*	*Focuses one-shot, lightweight hashes*
Low resource utilization, you can still watch movies or play games while cracking	*Focuses single dictionary based attacks*	*Supports sessions/restore*
Supports pause / resume while cracking	--	--

[32] Más información sobre las características que se implementarán y cuáles no puede encontrarse aquí: http://hashcat.net/wiki/feature_requests

Integrated thermal watchdog	--	--
20+ Algorithms implemented with performance in mind	--	--

En la tabla puede verse como la herramienta "Plus" es la indicada a utilizar en el caso de *hashes* con alto número de iteraciones mientras que la herramienta "Lite" se centra en hashes "rápidos". Todas las herramientas *OCLHashcat* son bastante completas y siguen siendo desarrolladas activamente.

Ejemplos de utilización y resultados obtenidos:

```
NTLM: ./cudaHashcat-plus32.bin -o --show --attack-mode 3 --force --
custom-charset1          ?l?d?s?u          --hash-type          1000
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa ?1?1?1?1?1?1?1?1
```

```
SHA1: ./cudaHashcat-plus32.bin -o --show --attack-mode 3 --force --
custom-charset1       ?l?d?s?u       --hash-type              1400
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
?1?1?1?1?1?1?1?1
```

```
MD5: ./cudaHashcat-plus32.bin -o --show --attack-mode 3 --force --
custom-charset1          ?l?d?s?u          --hash-type           0
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa ?1?1?1?1?1?1?1?1
```

En la siguiente tabla se muestran los resultados obtenidos:

	MD5	**SHA1**	**NTLM**
OCLHashcat-plus	550 MH/s	145 MH/s	900 MH/s
OCLHashcat-lite	800 MH/s	105 MH/s	1800 MH/s

Como puede observarse la versión "Lite" obtiene mejores resultados que la versión "Plus" en los casos de *hashes* rápidos. No obstante, *Hashkill* parece obtener mejores resultados que OCLHashcat mostrando una velocidad de 1050 M/s en MD5 y 223 M/s en SHA1. Esto es debido al modelo de tarjeta gráfica utilizada ya que tras analizar las comparativas que se pueden encontrar tanto en la página web de Hashcat como en la de Hashkill se puede ver que el rendimiento de ambos programas varía drásticamente dependiendo tanto del modelo como de la marca de la tarjeta. Como ejemplo de esto se muestra la siguiente tabla creada a partir de las comparativas oficiales.

GPU y Hash	**Hashkill**	**OCLHashcat-plus**	**OCLHashcat-lite**
ATI hd6990 **MD5**	**10965** MH/s	7385.2 MH/s	10908.6 MH/s
nVIDIA gtx570 **MD5**	1659 MH/s	1508.2 MH/s	**2170.5** MH/s
ATI hd6990 **SHA1**	3526 MH/s	3411.1 MH/s	**3764.5** MH/s
nVIDIA gtx570 **SHA1**	505 MH/s	**731.8** MH/s	665.7 MH/s

6.2.2 Conclusión

Para sacar el máximo partido a estas herramientas se recomienda comprobar la velocidad de cada una de ellas antes de realizar el ataque al algoritmo hash que se desee romper. Por lo general (como se puede ver en la tabla anterior) la herramienta OCLHashcat obtendrá resultados similares o mejores que Hashkill.

Como curiosidad comentar que se ha encontrado un documento[33] en el que se construyen dos *clusters* GPU uno con 8 tarjetas nVIDIA GTX580 y otro con 4 tarjetas dobles ATI HD6990 y se muestra una comparativa del rendimiento obtenido utilizando la herramienta OCLHashcat-lite. En el documento se llega a alcanzar una velocidad de 45.7 GH/s (*GigaHashes* MD5 por segundo) en el *cluster* ATI y una velocidad de 18.3 GH/s en el *cluster* nVIDIA.

Tras haber analizado las diferentes herramientas tanto de descifrado de archivos como de ruptura de funciones *hash* el siguiente paso es la automatización del programa comercial *Passware* para poder utilizarlo de a través de la línea de órdenes tanto de forma local como remota.

[33] *Building clusters*: http://www.tmto.org/docs/HardwareComparisonGTXvsHD.pdf

7

Automatización de Programas de Interfaz Gráfica

Tal y como hemos apuntado anteriormente, uno de los principales problemas que presentan los programas comerciales de descifrado de archivos es lo poco automatizables que son y la pérdida de tiempo que conlleva configurarlos a través de una interfaz gráfica (GUI). Esto es debido sobre todo a que la mayoría de estos programas son desarrollados para plataformas Windows y por lo tanto están pensados para ser utilizados por usuarios acostumbrados a la navegación por ventanas por lo que no suelen implementar su utilización por línea de órdenes. En este tipo de entornos, no es posible configurar un tipo de ataque concreto con una única orden sino que se debe navegar por todas las ventanas disponibles para llevar a cabo la configuración apropiada para un dterminado análisis. Otro problema asociado a la obligación de utilizar la interfaz gráfica es que resulta prácticamente imposible lanzar el programa de manera remota sin ayuda de ninguna otra herramienta adicional.

La necesidad de una interfaz por línea de órdenes se hace más acuciante en el caso de querer realizar un *cluster* de descifrado de archivos ya que en la mayoría de los casos no se dispondrá ni tan siquiera de un servidor gráfico o puede que el *cluster* se encuentre en un servidor y se desee acceder a él desde varios sitios diferentes a la vez lo que sería inviable si se accediese de manera gráfica.

En este apartado se presenta la herramienta de programación de eventos AutoIT[34] que se utilizará para automatizar el programa de descifrado de archivos *Passware Kit Forensic* facilitando así su posible uso en un *cluster* de tarjetas GPU.

[34] AutoIT es un software gratuito de scripting basado en BASIC de gran utilidad para la programación de eventos. Web oficial:

http://www.autoitscript.com/site/autoit/

7.1 Introducción AutoIT

AutoIT es un software gratuito de *scripting* basado en BASIC y diseñado para automatizar interfaces gráficas de Windows y en general para realizar cualquier tipo de scripts. Este lenguaje de *scripting* utiliza una combinación de simulaciones como movimientos del ratón, pulsación de teclas, control y manipulación de ventanas etc. para automatizar tareas de una forma poco posible para otros lenguajes (*i.e VBScript y SendKeys).* AutoIT además es bastante ligero, autocontenido y puede funcionar en cualquiera de las versiones de Windows sin ningún tipo de problemas.

7.1.1 Utilización de AutoIT

El programa AutoIT dispone de tres herramientas básicas para su utilización.

La principal herramienta es el editor de código SciTE donde se escribirán los *scripts* de eventos. El aspecto de este editor puede verse en la siguiente imagen:

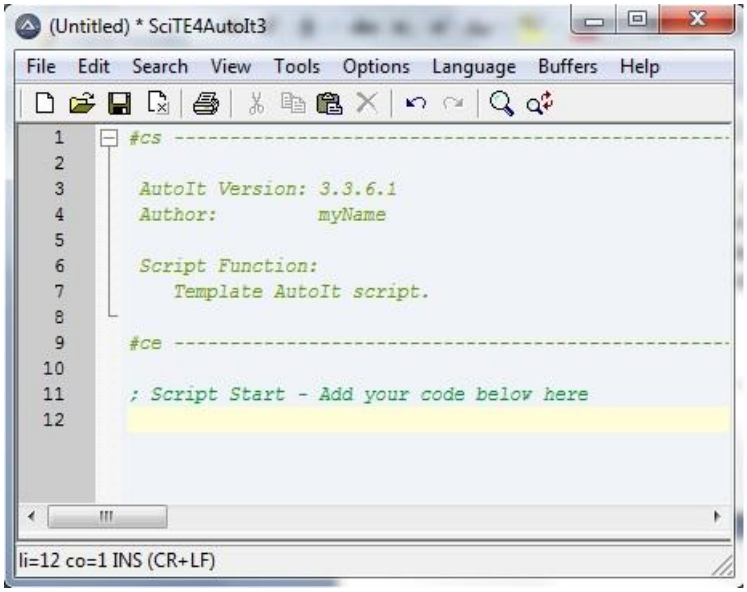


Fig. 8 - Editor SciTE de AutoIt v3

La segunda herramienta facilita mucho la labor de programación al permitir identificar cualquier tipo de control con el que se pueda interactuar, esta herramienta se denomina "AutoIT Window Info":

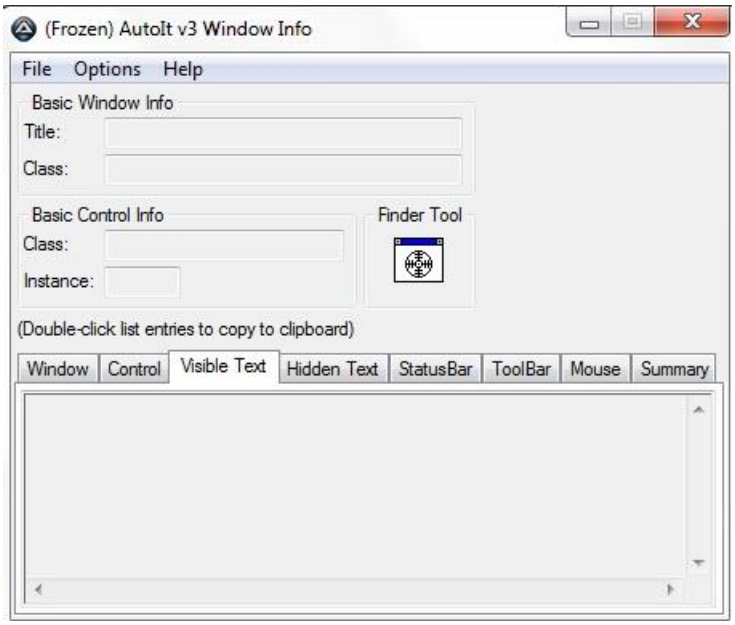


Fig. 9 - AutoIt v3 Window Info

Para utilizar esta herramienta basta con hacer click en el icono "Finder Tool" y arrastrarlo hasta el control deseado para obtener toda la información posible de ese objeto lo que permitirá al programador interactuar con el controlador mediante el lenguaje AutoIT.

Para entender mejor el funcionamiento de esta herramienta se muestra un ejemplo de su utilización programando la activación de la ventana de un programa, esto es, traerla al frente para interactuar con ella.

La ventana del programa *Passware* tiene el siguiente aspecto:

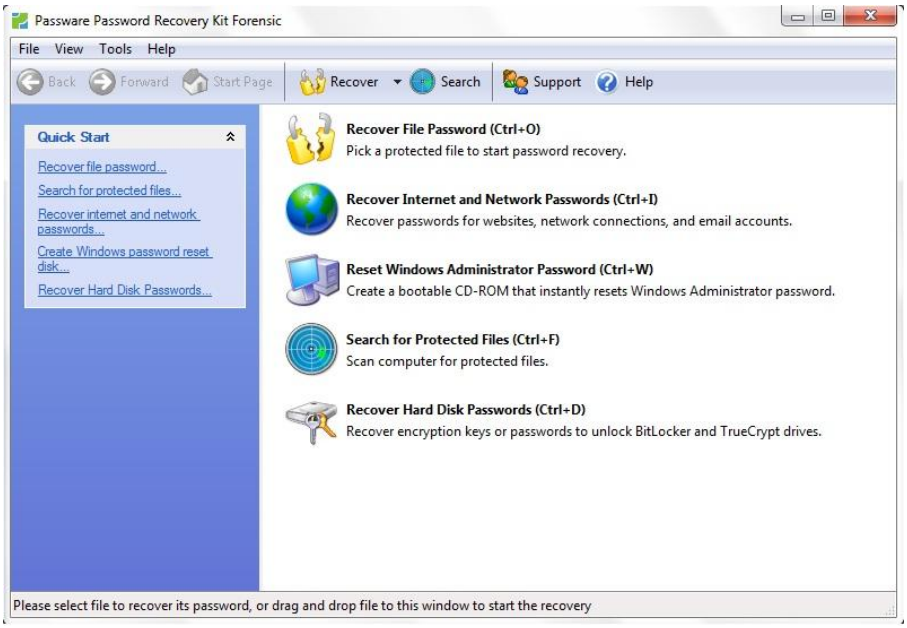


Fig. 10 - Programa Passware

Arrastrando el “Finder Tool” hasta el título de la ventana gráfica anterior se obtiene:

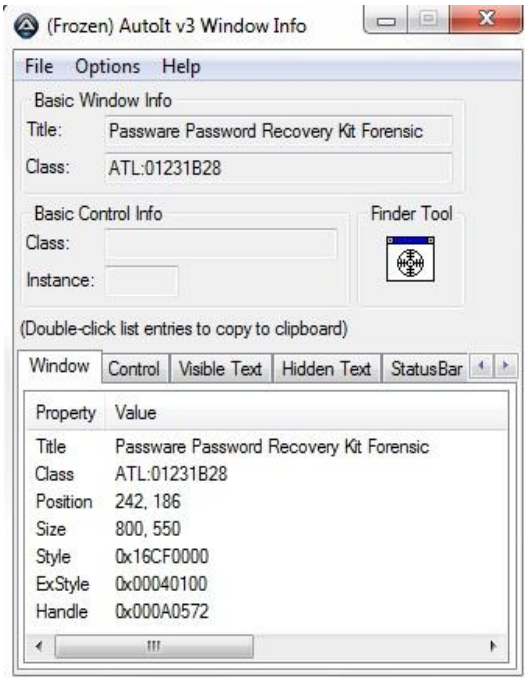


Fig. 11 – Información de la ventana de Passware

Como se puede ver el “Finder Tool” ha extraído toda la información posible del control “ventana” del programa *Passware*.

Para programar el código necesario para hacer click en la ventana se utilizará la tercera herramienta de AutoIT, “AutoIT Help File”. Esta herramienta contiene toda la información sobre los métodos existentes para interaccionar con una gran cantidad de controles gráficos, por ejemplo para activar la ventana en cuestión basta con buscar “Window activate” y seleccionar la función “WinActivate” para ver su sintaxis y ejemplos de utilización:

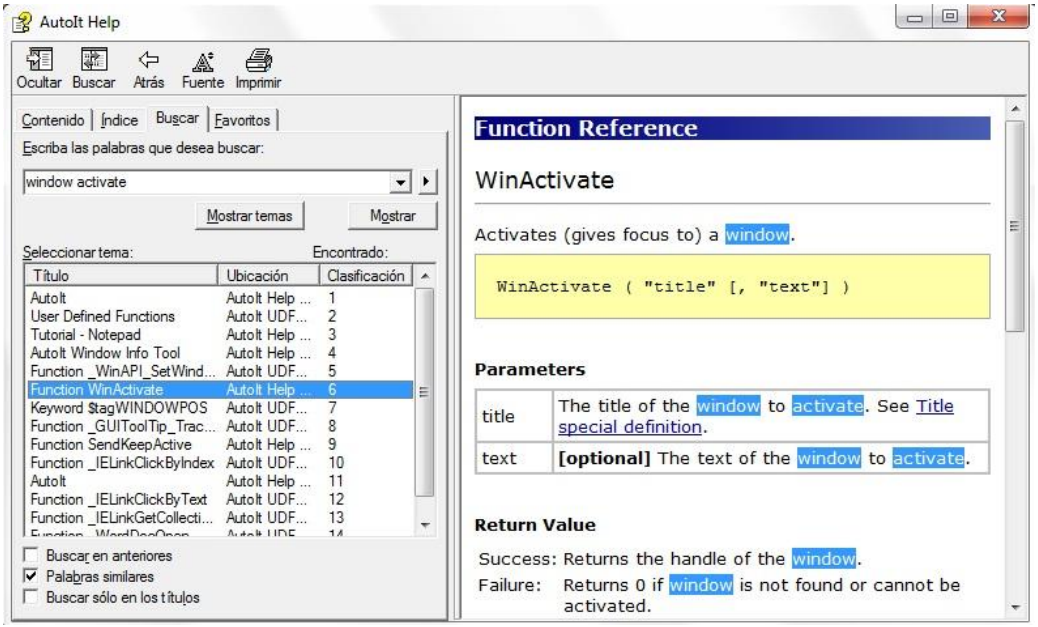


Fig. 12 – AutoIt Help

Observando la sintaxis del método, el script se escribiría así:

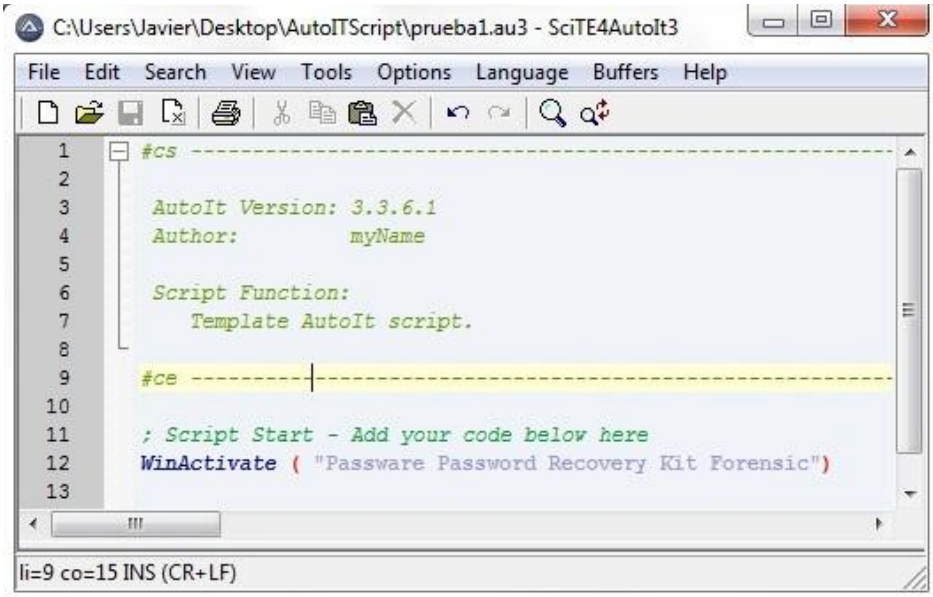


Fig. 13 – Script de ejemplo

Tras ejecutar este *script,* automáticamente la ventana del programa *Passware* se pondrá al frente de todas las demás.

7.2 Automatización de Passware Kit Forensic

La automatización por línea de órdenes de un programa basado en una interfaz gráfica de usuario no es una tarea sencilla. Una interfaz gráfica puede contener diferentes tipos de controles gráficos para realizar la configuración del programa dificultando así la tarea del programador. Incluso puede resultar que AutoIT no pueda interactuar con algunos controles especiales y por tanto se necesiten utilizar funciones específicas para solventar estas dificultades.

Otro de los problemas presentes en la automatización de los programas gráficos es el tiempo necesario para llevarla a cabo de manera completa sobre todo debido a la multitud de opciones y herramientas que un programa con interfaz gráfica puede ofrecer.

El programa *Passware Kit Forensic* es una herramienta profesional que ha sido desarrollada durante años incluyendo multitud de opciones y características que se pueden configurar a través de su interfaz gráfica.

Por estos motivos, aunque la automatización de *Passware* se ha realizado de manera parcial dado que no hemos automatizado todas las posibles operaciones, podemos considerar que es bastante completa de cara a los objetivos que perseguimos ya que se ha conseguido obtener una interfaz de línea de órdenes para varias de sus funciones principales como, por ejemplo, buscar archivos cifrados en el disco y configurar ataques de descifrado.

7.2.1 Lista de *scripts* realizados y su funcionalidad

A continuación se explican cada una de las funcionalidades de *Passware* que han sido automatizadas utilizando *scripts* en AutoIT. También se incluyen ejemplos de ejecución de los *scripts* para tener una referencia rápida sobre su utilización. Una descripción más completa de la sintaxis se encuentra en los apartados posteriores:

- ***Script* FindProtected.au3**:

  Script que sirve para lanzar por línea de órdenes la herramienta de búsqueda de archivos de *Passware* además de permitir la configuración de las opciones de búsqueda disponibles.

  Ejemplos de uso:

  ```
  AutoITScript>FindProtected.exe fast local
  AutoITScript>FindProtected.exe full drive NombreUnidadUSB
  ```

- ***Script* CheckSearchEnd.au3**:

  Script que comprueba el estado actual de la búsqueda y guarda los resultados en un archivo de texto en caso de haber finalizado. Los resultados se guardarán en la carpeta *SearchResults* dentro del directorio donde se encuentren los scripts, por ejemplo en "C:\AutoITScript\SearchResults". La carpeta *SearchResults* debe existir para el correcto funcionamiento del script.

  Ejemplo de uso:

  ```
  AutoITScript>CheckSearchEnd.exe
  ```

- ***Script* Decrypt.au3**:

  Este *script* permite lanzar el descifrado de archivos ya sea mediante ataques por fuerza bruta, ataques por diccionario, ataques predefinidos o ataques híbridos combinando fuerza bruta y diccionarios. Además permite configurar estos ataques automatizando multitud de opciones que ofrece *Passware*.

  Para que este *script* funcione correctamente necesita de la existencia de tres archivos que permitirán configurarlo de manera rápida sin que los parámetros pasados por consola se multipliquen. Los archivos se detallan a continuación:

  1) Documento de texto *CustomBruteForce.txt*:

     Puede estar vacío. Los caracteres que se incluyan en este archivo serán los caracteres adicionales que se utilizarán en los ataques por fuerza bruta.

  2) Documento de texto *Pattern.txt*:

     Al igual que el anterior puede estar vacío, este archivo sirve para especificar el patrón de generación de claves que se utilizará en los ataques por diccionario. La explicación de cómo utilizar esta característica puede encontrarse en el documento de ayuda del programa *Passware*.

3) Archivo de configuración *MyModel.xml*:

Passware utiliza el lenguaje XML para guardar la configuración de los ataques. Este archivo en concreto guarda la configuración de un ataque vacío y por lo tanto permite resetear el estado del programa a su origen, así cada vez que se lance el script Decrypt.au3 lo primero que se hará será cargar este modelo en *Passware* automáticamente para borrar la información de configuración previa. A continuación se muestra el contenido de este archivo aunque puede encontrarse en el CD-ROM adjunto al proyecto en la carpeta *AutoITScripts*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<XmlValues version="1.0">
  <MetaData>
    <ModelDescription>Passware Kit Attacks
List</ModelDescription>
    <ModelVersion>1.0</ModelVersion>
  </MetaData>
  <Node>
    <Name>search-project</Name>
    <Node>
      <Name>model</Name>
    </Node>
  </Node>
</XmlValues>
```

Ejemplos de uso:

1. Ataque por diccionario en español con palabras de longitud mínima 4 y máxima 8:

```
AutoITScript>Decrypt.exe C:\Cifrado.rar  dic  spanish  4  8
```

2. Ataque de fuerza bruta en ruso de longitud 1 a 9 incluyendo letras minúsculas mayúsculas y números:

```
AutoITScript>Decrypt.exe C:\Cifrado.rar  bf  russian  1  9  5
```

3. Ataque híbrido combinando fuerza bruta (números y símbolos, longitud de 1 a 4) y diccionario utilizando un diccionario personalizado (palabras de entre 3 y 6 letras):

```
AutoITScript>Decrypt.exe C:\Cifrado.rar  bf-dic  spanish  1 4 12
C:\Diccionario.txt 3 6
```

- ***Script* Restore.au3**:

Script que permite continuar con el proceso de descifrado que se estaba ejecutando la última vez que se lanzó *Passware*. Si por cualquier motivo el programa *Passware* se cierra, ya sea de forma intencionada o inesperada, se podrá recuperar el estado del ataque anterior utilizando este *script*. El *script* Restore.au3 debe ser ejecutado sólo en el caso de querer recuperar el proceso anterior. Si tras el cierre inesperado o intencionado de *Passware* se ejecutase cualquiera de los demás scripts antes que este *script* de recuperación, no se podría continuar con el ataque y se perdería esa información.

Ejemplo de uso:

```
AutoITScript>Restore.exe
```

- ***Script* CheckEnd.au3**:

  Este *script* comprueba si el ataque de descifrado ha finalizado y en ese caso guarda los resultados. Este *script* realiza para los resultados de un ataque lo que el script CheckSearchEnd.au3 realiza para los resultados de la búsqueda. En el caso de que el ataque haya finalizado se guardarán los resultados en archivos HTML en el directorio *Results*. Este directorio debe existir de lo contrario se producirá un error.

  Ejemplo de uso:

  ```
  AutoITScript>CheckEnd.exe
  ```

- ***Script* Close.au3**:

  Este *script* permite cerrar el programa *Passware*. Si en el momento del cierre *Passware* estaba realizando un ataque de descifrado, utilizando el *script* Restore.au3 podríamos reanudar el proceso.

  Ejemplo de uso:

  ```
  AutoITScript>Close.exe
  ```

- ***Script* numPassware.au3**:

  Este *script* tan sólo contiene un par de líneas de código y devuelve el número de instancias de *Passware* que se están ejecutando en un instante dado.

  Al ser un *script* tan simple su código se lista en este mismo apartado:

  ```
  ; Script Start - Add your code below here
  $array = ProcessList ( "PasswareKitForensic.exe" )
  ConsoleWrite("Procesos Passware en ejecucion: "&$array[0][0]&
  @CRLF)
  ; Registrar e informar por consola (Ver apartado 7.2.2
  Mecanismos implementados en todos los scripts)
  ```

  Ejemplo de uso:

  ```
  AutoITScript>numPassware.exe
  ```

7.2.2 Mecanismos implementados en todos los scripts

Antes de explicar la sintaxis, el procedimiento y las dificultades encontradas en la realización de cada *script*, se deben explicar los mecanismos comunes implementados en todos ellos que permiten obtener información sobre el estado del programa. Esta forma de obtener realimentación por parte de los *scripts* es muy útil sobretodo en el caso de ejecutarlos en un equipo remoto o en un sistema que no disponga de una interfaz para poder visualizar el estado del programa *Passware*.

Los mecanismos de comunicación implementados permiten por un lado mostrar información por consola y por otro lado registrar los distintos eventos en un archivo de *log* a fin de poder consultarlo y ver el historial o el resultado de las operaciones.

1) Mostrar información por consola:

AutoIT proporciona una serie de funciones para interactuar con la consola, en este caso solo se necesitará utilizar una de estas funciones para escribir en pantalla.

```
ConsoleWrite ( "datos" )
```

Esta función escribe los datos pasados como argumento en el flujo *STDOUT* o de salida estándar. Para que la función escriba los datos en la consola habrá que especificar la opción "console" al compilar el script:

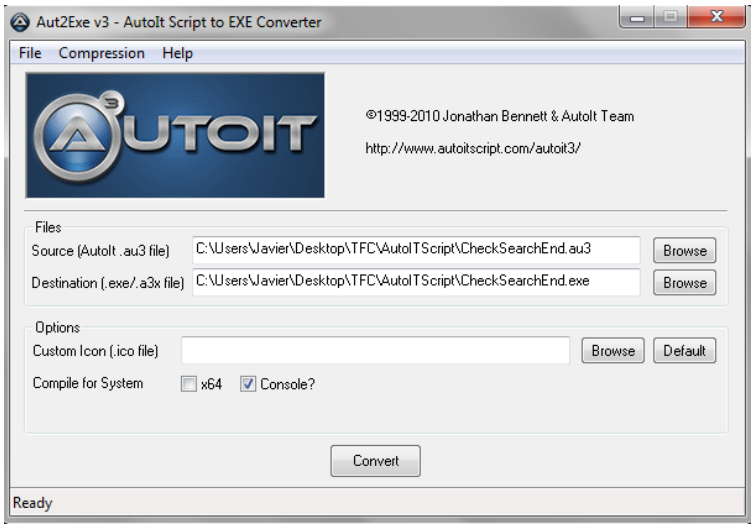


Fig. 14 – Herramienta Aut2Exe v3

Así pues, en todos los *scripts* cuando sea necesario devolver cierta información sobre el resultado o el estado de las operaciones, se añadirá una llamada a la función *ConsoleWrite*. Para clarificar su uso se muestra el ejemplo del script CheckEnd.au3:

```
$name = "PassResults"&"-"&@MDAY&"-"&@MON&"-"&@YEAR&"_"&@HOUR&"-
"&@MIN&"-"&@SEC
$path = @ScriptDir&"\Results\"
$path = $path&$name
WinWaitActive("Guardar como")
CntrolSetText("Guardar como","",1148,$path,1)
Send("!g")
ConsoleWrite("Ataque finalizado. Resultado guardado en el archivo
"&$path & @CRLF)
```

2) Registro de eventos en un archivo de log:

Es interesante tener todos los eventos que producen los scripts registrados en un archivo de *log*. De esta forma aparte de poder leer la información por consola se podrá tener la información disponible para más adelante. El archivo *log.txt* donde se registrarán todos los eventos se creará en la misma carpeta en la que se ejecuten los scripts.

Para realizar esta tarea se han utilizado las funciones de tratamiento de ficheros que proporciona AutoIT, en el ejemplo anterior la entrada de *log* se crearía así:

```
$fileLog = FileOpen("log.txt", 1)
FileWriteLine($fileLog,@MDAY&"/"&@MON&"/"&@YEAR&"-
"&@HOUR&":"&@MIN&" Ataque finalizado. Resultado guardado en el
archivo "&$path & @CRLF)
```

El *flag* numérico de valor 1 en la llamada a la función *FileOpen* implica abrir el archivo añadiendo contenido al final de este y si el archivo no existe crearlo.

3) Evitar la duplicidad de *scripts*

Otro de los mecanismos implementados de forma común en todos los *scripts* trata de evitar que un *script* no funcione correctamente debido una ejecución anterior errónea. Más específicamente se pretende evitar que al ejecutar un *script* de AutoIT exista ya otro en ejecución o en estado de espera. Para ello se utilizan al comienzo de cada *script* unas líneas de código similares a las mostradas a continuación:

```
AutoItWinSetTitle("Principal1") ;Cambia en función del script
$a = WinGetProcess ("Principal1")
$i=1
$array = ProcessList ("Decrypt.exe") ;Cambia en función del script
While $i <= ($array[0][0])
    If $a == ($array[$i][1]) then
        $i=$i+1
    Else
        ProcessClose($array[$i][1])
        ConsoleWrite("Cerrado un script AutoIT, proceso:
  "&$array[$i][1])
        $fileLog = FileOpen("log.txt", 1)
        FileWriteLine($fileLog,@MDAY&"/"&@MON&"/"&@YEAR&"-
"&@HOUR&":"&@MIN&" Cerrado un script AutoIT, proceso:
"&$array[$i][1]&@CRLF)
        FileClose($fileLog)
        $i=$i+1
    EndIf
WEnd
```

El código anterior nombra al último script lanzado con un identificador único, en este caso el identificador es “Principal1”, una vez nombrado se buscan todos los procesos “NombreScript.exe” y se terminan a no ser que coincida el nombre con el identificador. De esta forma se asegura que solo hay un *script* de AutoIT ejecutándose a la vez.

7.2.3 Ejecución remota de los scripts

Para la ejecución remota de los *scripts* es necesario utilizar alguna herramienta adicional. Si se utiliza, por ejemplo, una conexión ssh se podrá ejecutar el script de forma remota pero cuando el *script* intente ejecutar el programa *Passware* este no se lanzará gráficamente en el destino aunque sí se creará un proceso Passware.exe. Para solucionar este problema y lograr que se ejecute la GUI de *Passware* se propone utilizar la herramienta de red *Netcat*.

En primer lugar es necesario descargar la herramienta *netcat* y descomprimirla por ejemplo en la ruta “C:\” (Esta puede obtenerse desde la url http://www.securityfocus.com/tools/139). Una vez descargada y descomprimida basta con acceder al directorio para poder ejecutar la herramienta ‘nc’.

En el **nodo servidor** donde se encuentra instalado *Passware* junto con los scripts de AutoIt se debe ejecutar la sentencia de escucha (Puerto=6000):

```
C:\nc11nt>nc -L -d -e cmd.exe -p 6000
```

-L: Deja un puerto abierto a la espera de una conexión permitiendo cerrar y volver a abrir conexiones.

-d: Esta opción desvincula el programa de la consola y es el que permite a AutoIT ejecutar el GUI de *Passware* para trabajar con las ventanas. Activa el modo background.

-e cmd.exe: cuando recibe una conexión devuelve una consola del servidor, desde aquí se deberán ejecutar los scripts para realizar tareas de descifrado o búsqueda.

En el **nodo cliente** que accederá remotamente al servidor para descifrar archivos se debe ejecutar la sentencia de conexión (IP del servidor=192.168.31.128 y Puerto=6000):

```
nc -vv 192.168.31.128 6000
```

Nota: Existen más herramientas que pueden ser utilizadas para el mismo propósito, por ejemplo, el software "remote execution[35]" o "rexec" también permite ejecutar la GUI de programas de forma remota.

7.3 Automatización de la búsqueda de archivos cifrados

La búsqueda de archivos cifrados no tiene por qué automatizarse para su uso por consola ya que esta tarea no presenta un incremento de velocidad al utilizar las tarjetas GPU y por tanto no tiene por qué realizarse la búsqueda dentro de un *cluster* automatizado. No obstante, sería conveniente tener todo el sistema de búsqueda/descifrado unificado en la misma máquina y puede resultar más cómodo y rápido ejecutar la búsqueda mediante línea de órdenes.

El primer paso de cara a automatizar el trabajo con cualquier software mediante este procedimiento de programación de eventos es determinar el proceso que un usuario debería llevar a cabo para lanzar la tarea que se pretende automatizar. En el caso de la búsqueda de archivos en *Passware*, se tiene:

[35] Remote execution: http://www.codeproject.com/Articles/3518/Executing-programs-remotely-in-Windows-NT-2000-XP

1) Pantalla de bienvenida:

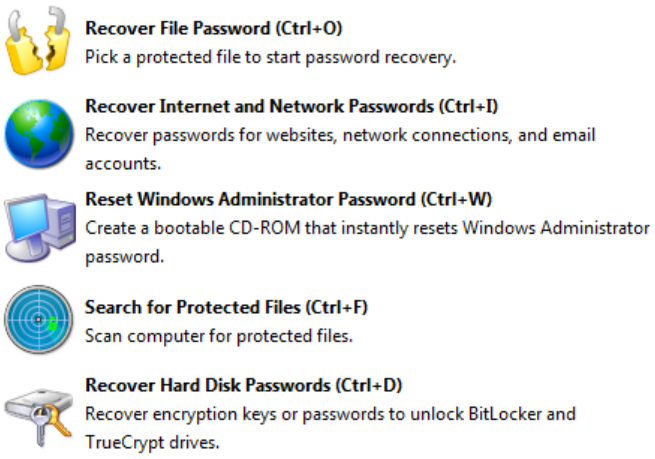


Fig. 15 – Pantalla de bienvenida de Passware

Se puede observar como la búsqueda de archivos puede lanzarse pulsando la combinación de teclas Ctrl + F.

Podríamos automatizar este proceso mediante AutoIT, esto sería tan sencillo como:

```
Send("^f")     ;ctrl + f --> search for protected files
```

2) Una vez seleccionada la opción de búsqueda, aparecen dos opciones de configuración "Type of Scan" y "Where to Scan". Utilizando la herramienta "AutoIt Window Info" se pueden obtener los identificadores de cada uno de los controles (*radio buttons*) permitiendo la configuración de la búsqueda:

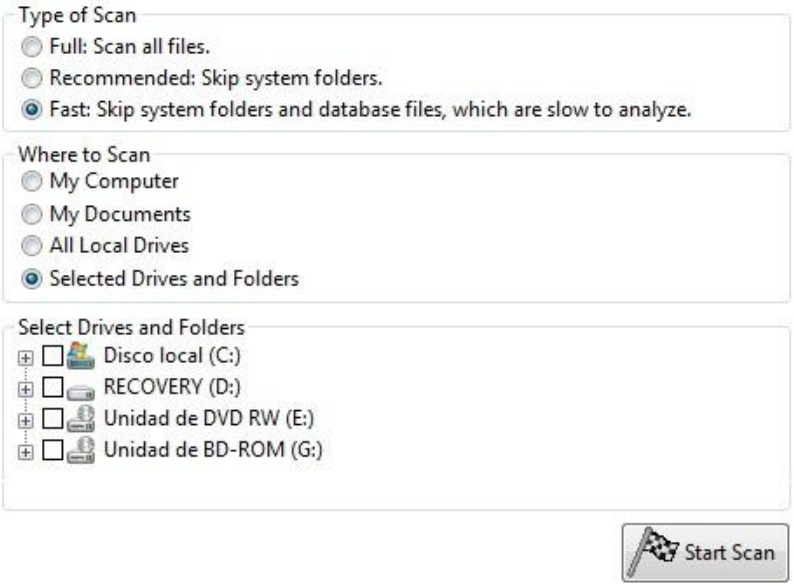


Fig. 16 – Opciones de configuración de la búsqueda de archivos en Passware

En la siguiente imagen se muestra un ejemplo de cómo se obtiene el identificador del botón “Recommended: Skip system folders”:

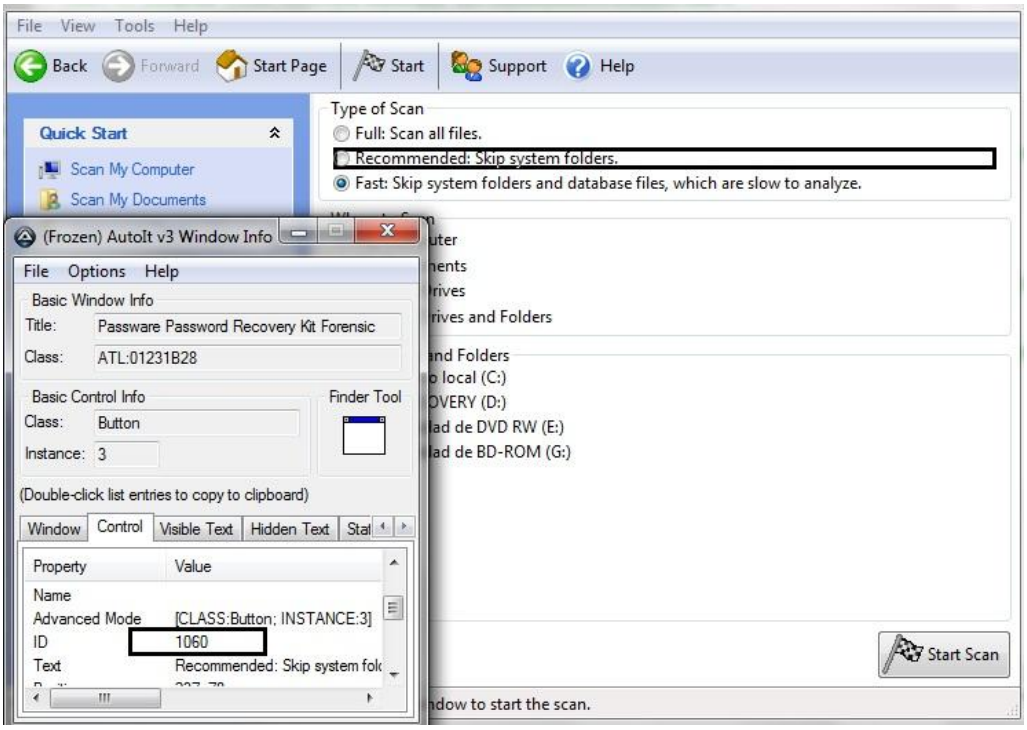


Fig. 17 – Obtención del identificador de un control mediante el Finder Tool de AutoIT

El identificador recuadrado en negro es *ID=1060* y sirve para identificar de manera unívoca a ese control en concreto. Este valor no variará entre ejecuciones ni tampoco variará de un ordenador a otro. Utilizando AutoIT se puede activar la selección de este control de la siguiente forma:

```
WinWaitActive("Passware Password Recovery Kit Forensic","Type of Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "", 1060)
```

Obteniendo los identificadores de todos los controles *radio button* se han programado los eventos necesarios para configurar la búsqueda dependiendo de los argumentos pasados por consola. El *script* FindProtected.au3 necesita dos argumentos para realizar la configuración de la búsqueda:

- El primer argumento indica el tipo de escaneo a realizar, su valor puede ser ‘full’, ‘nosys’ o ‘fast’:
  1) Full: Se seleccionará la opción “Full: Scan all files.”
  2) Nosys: Se seleccionará la opción “Skip system folders”
  3) Fast: Se seleccionará la opción “Skip system folders and database files”
- El segundo argumento indica el lugar donde se realizará la búsqueda, su valor puede ser ‘pc’, ‘docs’ o ‘local’:
  1) PC: Se seleccionará la opción “My Computer”
  2) Docs: Se seleccionará la opción “My Documents”

3) Local: Se seleccionará la opción “All local drives”

Para ver como se han implementado estas operaciones, se muestra a continuación una porción del *script* de búsqueda (*Script* completo adjunto en el CD-ROM). La selección de la primera opción en función del argumento pasado por consola puede hacerse así:

```
If $CmdLine[1]=="full" Then
  WinWaitActive("Passware Password Recovery Kit Forensic","Type of
Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "",
1059)
ElseIf $CmdLine[1]=="nosys" Then
  WinWaitActive("Passware Password Recovery Kit Forensic","Type of
Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "",
1060)
ElseIf  $CmdLine[1]=="fast" Then
  WinWaitActive("Passware Password Recovery Kit Forensic","Type of
Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "",
1061)
EndIf
```

7.3.1 Opción “Selected Drives and Folders”:

La búsqueda puede especificarse aún más si se indican en que directorios se desea buscar los archivos. Sin embargo, automatizar esta tarea presenta algunas dificultades ya que el control utilizado por *Passware* para listar los directorios no puede ser accedido directamente por AutoIT lo que obstaculiza su automatización. En la siguiente imagen se puede ver que la herramienta de información de AutoIT no es capaz de navegar a través de los distintos elementos del árbol de directorios, sino que obtiene un solo identificador para todo el conjunto:

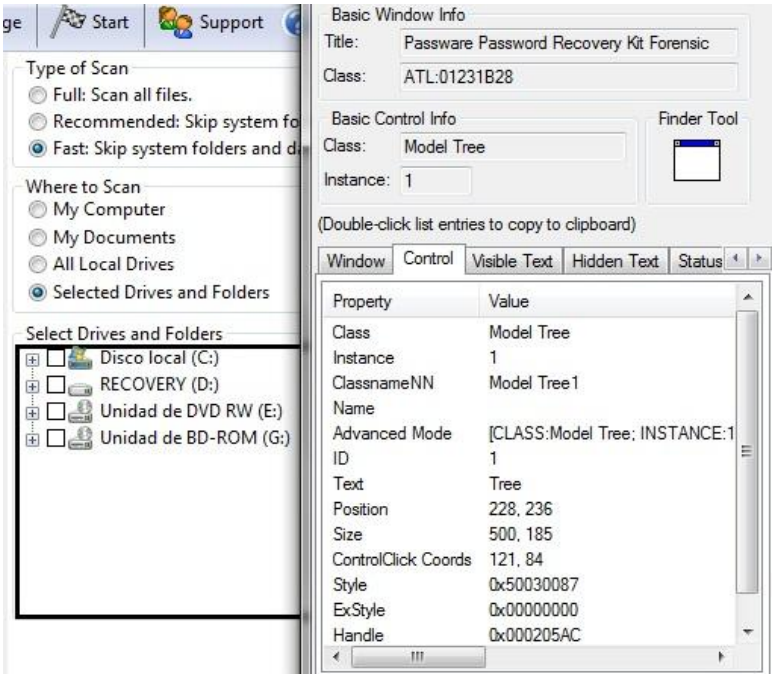


Fig. 18 – Finder Tool muestra un identificador único para todo el conjunto

El disponer únicamente de un identificador del control “Model Tree” junto con el hecho de que su contenido varíe en función del ordenador donde se ejecuta hace muy difícil la programación de eventos sobre él.

Para solucionar este problema, hay que tener en cuenta que la búsqueda de archivos se realizará siempre en un disco duro objetivo y no en subcarpetas de este ya que en un caso de informática forense de lo que se dispondrá será de una copia del disco. En el sistema de búsqueda/descifrado en el que se utilice *Passware*, habrá que montar la imagen de la evidencia. A partir de aquí, la evidencia aparecerá montada con un nombre unívoco que diferirá de los comunes “Disco Local” o “Unidad de…”. En caso contrario, bastaría con configurar el programa de montaje para cambiar el nombre con el que se muestra la evidencia.

La forma de automatizar los eventos sobre este control se basa en simular la pulsación de las letras del nombre de la evidencia para que ésta se seleccione y a continuación simular la pulsación de la barra espaciadora para marcar su casilla y poder escanear la imagen del disco. En la imagen siguiente se observa una unidad (o imagen) llamada “WORK“ seleccionada mediante la pulsación de teclas:

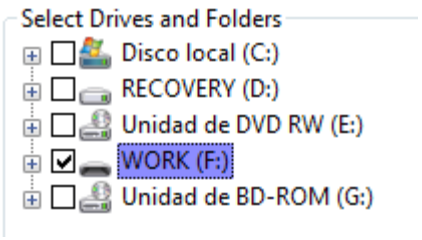


Fig. 19 – Ejemplo de selección mediante teclado

La forma de implementar esto por consola es introduciendo el parámetro “drive” y a continuación el nombre de la unidad/imagen como tercer parámetro.

El código necesario sería parecido a esto:

```
ElseIf  $CmdLine[2]=="drive" Then
  WinWaitActive("Passware Password Recovery Kit Forensic","Where
to Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "",
1065)
  ControlClick("Passware Password Recovery Kit Forensic","",1)
;click para posicionar el foco en el control
  Send($CmdLine[3]) ;Teclear el nombre de la imagen
  Send(" ")         ;Pulsar barra espaciadora
```

El *script* se lanzaría por consola de la siguiente forma:

```
C:\RutaScripts>FindProtected.exe fast drive WORK
```

El *script* FindProtected.au3, al igual que todos los demás, debe ser compilado utilizando la herramienta de compilación incluida con AutoIT o a través del editor SciTE comentado anteriormente. Para más detalles se recomienda consultar el apartado “7.9 Utilización de los *scripts* en un nuevo sistema”.

7.3.2 Evitar la pausa del *script* por ventanas emergentes

El programa *Passware* guarda el estado del programa cuando este se cierra, por lo que al abrir el programa puede que pregunte si se desea continuar con el proceso anterior mediante una ventana emergente que espera una acción por parte del usuario lo que provocará que el *script* se quede pausado indefinidamente. Para solucionar esto habrá que programar los eventos necesarios para cerrar esas ventanas en caso de que aparezcan.

Continuando con la opción de búsqueda de archivos, puede surgir una ventana emergente si se lanza la búsqueda sin indicar ningún directorio:

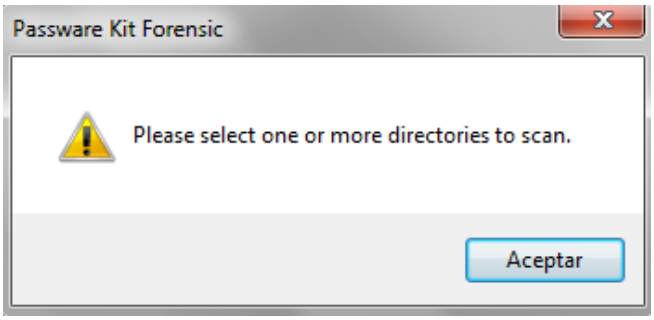


Fig. 20 – Ventana emergente en el caso de búsqueda vacía

El programa *Passware* recuerda la configuración utilizada la vez anterior y por ello si se ejecutara dos veces el *script* de búsqueda indicando la unidad (*drive*) se produciría el error mostrado en la imagen. En la primera ejecución se marcaría la unidad llamada WORK en el ejemplo anterior y en la segunda ejecución esta unidad aparecería ya marcada pues *Passware* guarda la configuración anterior. Esto haría que el *script*, al simular la pulsación de la barra espaciadora, desactivara la casilla de la unidad y lanzara la búsqueda sin especificar ningún directorio. Una forma de solventar esto es verificar si una vez lanzada la búsqueda la ventana emergente aparece. Si es así, bastaría con cerrarla y volver a seleccionar la unidad:

```
ElseIf  $CmdLine[2]=="drive" Then
  WinWaitActive("Passware Password Recovery Kit Forensic","Where
to Scan")
  ControlClick("Passware Password Recovery Kit Forensic", "",
1065)
  ControlClick("Passware Password Recovery Kit Forensic","",1)
  Send($CmdLine[3])
  Send(" ")
  ControlClick("Passware Password Recovery Kit Forensic", "", 2)
  $error = WinExists ("Passware Kit Forensic", "Please select one
or more directories to scan." )
  If $error==1 Then
        WinActivate("Passware Kit Forensic")
        ControlClick("", "Aceptar", 2) ; Aceptar y volver a
seleccionar la unidad
        WinWaitActive("Passware Password Recovery Kit
Forensic","Where to Scan")
        ControlClick("Passware Password Recovery Kit Forensic", "",
1065)
        ControlClick("Passware Password Recovery Kit
Forensic","",1)
        Send($CmdLine[3])
        Send(" ")
```

```
   EndIf
   ControlClick("Passware Password Recovery Kit Forensic", "", 2)
         ;En caso de error se lanza la busqueda por segunda vez
EndIf
```

El script completo puede encontrarse en la carpeta *AutoITScripts* del CD-ROM adjunto.

7.4 Comprobar estado de la búsqueda

Una vez lanzado el *script* de búsqueda, se necesita otro *script* que compruebe si la búsqueda terminó satisfactoriamente y que guarde los resultados en un archivo de texto dado el caso.

Cuando finaliza la búsqueda, *Passware* muestra una ventana emergente con el mensaje "Scan complete", por lo que se comprobará si existe esta ventana y se guardarán los resultados pulsando el botón "Save Files List" tal y como se ve en la siguiente imagen:

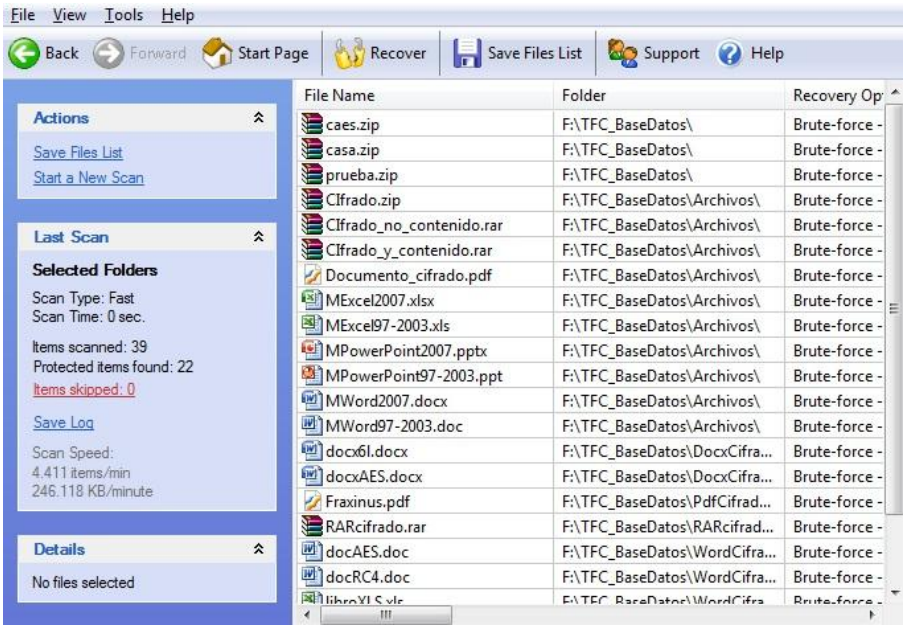


Fig. 21 – Ejemplo de resultados de búsqueda

Al igual que ocurría con el controlador "Model Tree", en este caso la barra de botones es un controlador de tipo *ToolbarWindow32* que tiene un identificador único para toda la barra y los botones no pueden ser accedidos directamente por AutoIT por lo que para pulsar el botón de guardar los archivos habrá que posicionar el ratón en las coordenadas del botón y hacer click. Para asegurar que el clic se hace en el mismo lugar independientemente de la posición de la ventana, se utilizarán las coordenadas relativas al control *ToolbarWindow32*.

El código de estos pasos se muestra abajo:

```
$a = WinActivate("Passware Kit Forensic", "Scan Complete.")
If $a Then
   WinWaitActive("Passware Kit Forensic")
   ControlClick("Passware Kit Forensic", "Aceptar", 2) ;
```

```
  ControlClick("Passware Password Recovery Kit Forensic", "",
"[CLASS:ToolbarWindow32; INSTANCE:1]", "Primary", 1, 400, 20 )
;Guardar resultados
```

Una vez pulsado el botón 'guardar' se muestra el formulario típico "Guardar como" del explorador de Windows:

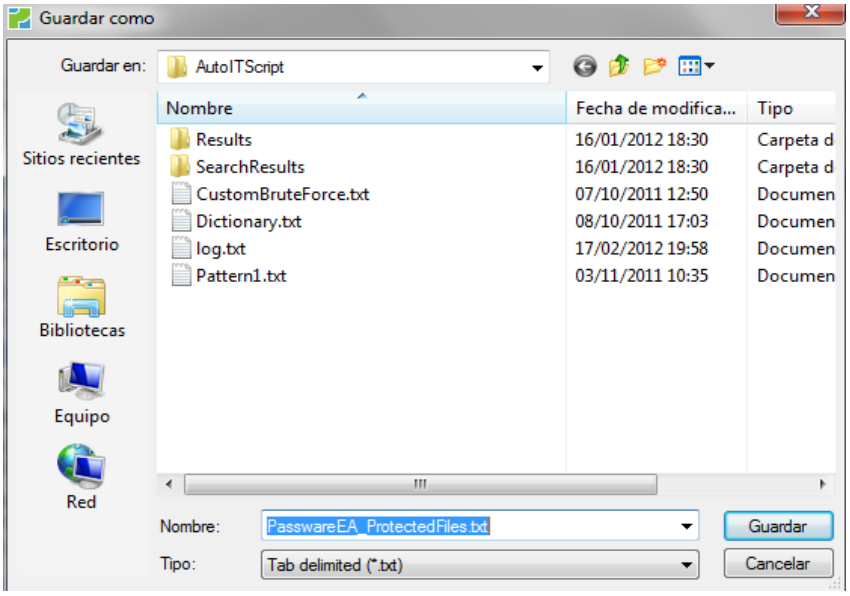


Fig. 22 – Formulario 'Guardar Como' del explorador de windows

El archivo se guardará con el nombre unívoco "ListProtected-Dia-Mes-Año-hora-minuto-segundo" dentro del directorio *SearchResults*:

```
$path = @ScriptDir&"\SearchResults\"
$path = $path&$name
WinWaitActive("Guardar como")
ControlSetText("Guardar como","",1148,$path,1)
Send("!g")
```

Si se diera el caso de que la búsqueda tardara más tiempo y no terminara, se mostraría un mensaje indicándolo por consola y el *script* terminará.

7.5 Descifrado de archivos

El descifrado de archivos es la parte más compleja de automatizar ya que *Passware* tiene multitud de opciones de ataque disponibles y automatizarlas todas conllevaría mucho tiempo y esfuerzo. Por ello, se ha hecho hincapié en automatizar las opciones más interesantes como son:

- Ataques predefinidos por el programa *Passware* dependiente del tipo de archivo a descifrar.
- Ataques por fuerza bruta incluyendo letras minúsculas, mayúsculas, números, símbolos, espacios, utilizar una lista de caracteres personalizada o utilizar letras de otros idiomas entre los cuales se encuentran árabe, holandés, inglés, francés, alemán, italiano, portugués, ruso y español.

- Ataques por diccionario utilizando los diccionarios predefinidos de *Passware* en todos los idiomas enumerados en el párrafo anterior o la opción más interesante de todas que permite utilizar un diccionario ajeno al programa que se encuentre en el sistema.
- Ataques híbridos de fuerza bruta y diccionarios que permiten generar cadenas más complejas como 'casa%$12' o al contrario '%$12casa' incluyendo todas las opciones de configuración descritas anteriormente.
- Ataques híbridos probando tanto las combinaciones FuerzaBruta-Diccionario como Diccionario-FuerzaBruta al mismo tiempo.

Para configurar los ataques en *Passware* hay que crear una lista a la que se añadirán los ataques que se quieran realizar. En la siguiente imagen pueden observarse en la parte derecha los diferentes tipos de ataque y en la parte izquierda la lista con los ataques que han sido añadidos.

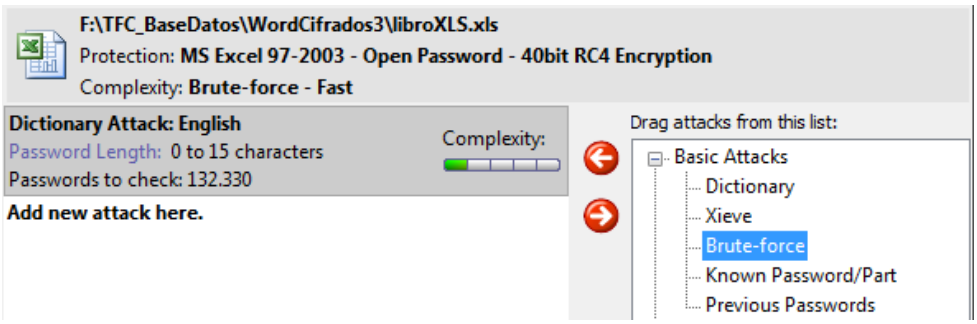


Fig. 23 – Forma gráfica de selección de ataques

El *script* añadirá a la lista el ataque indicado y lo configurará según los parámetros pasados por consola. Se requiere que la lista este vacía nada más iniciar el programa. Dado que *Passware* guarda la información de la configuración anterior, será necesario cargar un modelo de ataques vacío para así reiniciar la lista. Además será necesario que la lista de la derecha aparezca siempre expandida ya que de lo contrario podrían ocurrir errores al ejecutar el *script*. La realización de estas operaciones se muestra en las siguientes líneas de código:

```
WinWaitActive("Passware Password Recovery Kit Forensic")
Send("^e")
; Cargar MyModel.xml con ataques vacíos
WinWaitActive("Passware Password Recovery Kit Forensic","Load
Attacks")
ControlClick("Passware Password Recovery Kit Forensic", "", 2182)
WinWaitActive("Abrir")
ControlSetText("","",1148,@ScriptDir & "\MyModel.xml",1)
Send("!a")
; Expandir la lista de los tipos de ataque
WinWaitActive("Passware Password Recovery Kit Forensic","Drag
attacks from this list:")
ControlTreeView ("Passware Password Recovery Kit Forensic", "Drag
attacks from this list:", 2001, "Expand", "#0")
```

Una vez en este punto se procederá en función de los parámetros pasados por consola:

- Si se indica el parámetro 'predefined' se utilizarán los ataques predefinidos por *Passware*.
- Si se indica 'dic' se realizará un ataque por diccionario.
- Si se indica 'bf' se realizará un ataque por fuerza bruta.
- Si se indica 'dic-bf' o 'bf-dic' se realizará un ataque híbrido respetando el orden al generar las claves, es decir, 'dic-bf' generará claves del tipo palabra-combinación y 'bf-dic' generará claves del tipo combinación-palabra.
- Por último se puede utilizar el parámetro r-dic-bf para realizar los dos tipos de ataque híbrido al mismo tiempo.

Para añadir los ataques híbridos, es necesario utilizar un ataque llamado 'Join Attacks' y adjuntarle los ataques a realizar, por ejemplo en el caso 'bf-dic' habrá que añadir el ataque 'join attacks' y justo debajo el ataque de fuerza bruta seguido del ataque por diccionario, ambos configurados previamente.

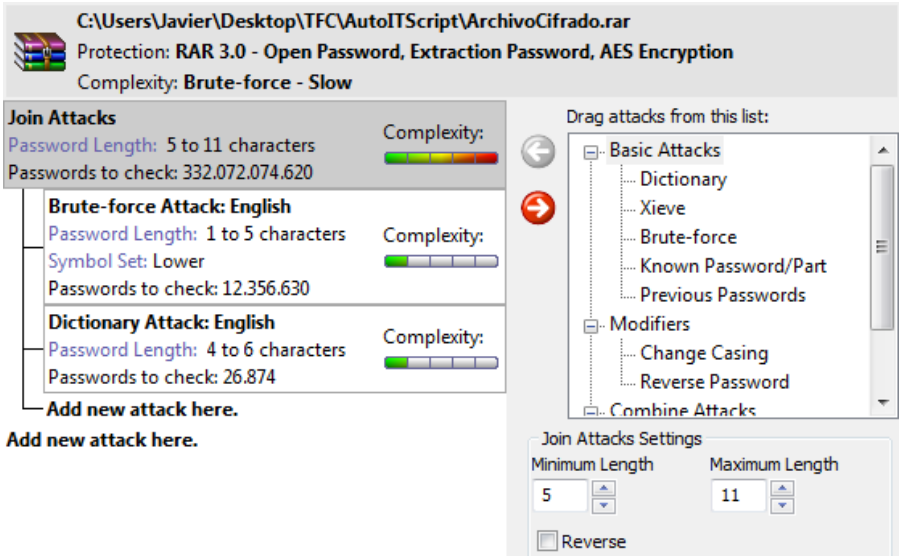


Fig. 24 – Ejemplo de configuración de ataque híbrido

Para facilitar la creación del script, se han creado dos funciones llamadas *Dictionary* y *BruteForce* que realizan la selección y la configuración de estos ataques utilizando los parámetros pasados por línea de órdenes. A continuación puede verse la estructura que se utiliza en el *script*:

```
If $CmdLine[2]=="dic" Then
   ;Llamada función Dictionary($language,$min,$max)
   Dictionary($CmdLine[3],$CmdLine[4],$CmdLine[5])
ElseIf  $CmdLine[2]=="bf" Then
   ;Llamada función BruteForce($language,$min,$max,$optionsMask)
   BruteForce($CmdLine[3],$CmdLine[4],$CmdLine[5],$CmdLine[6])
ElseIf $CmdLine[2]=="dic-bf" Then
   ;Join Attacks
   ControlTreeView ("Passware Password Recovery Kit Forensic",
"Drag attacks from this list:", 2001, "Select", "#2|Join Attacks")
   ControlClick("Passware Password Recovery Kit Forensic", "",
2002) ; pasar a la izquierda el seleccionado
   WinWaitActive("Passware Password Recovery Kit Forensic")
```

```
   $minimo = $CmdLine[4] + $CmdLine[7] ;minDic + minBF
   $maximo = $CmdLine[5] + $CmdLine[8] ;maxDic + maxBF
   ControlSetText("","",2032,$minimo,1) ; Min length
   ControlSetText("","",2036,$maximo,1) ; Max length
   Dictionary($CmdLine[3],$CmdLine[4],$CmdLine[5])
   BruteForce($CmdLine[6],$CmdLine[7],$CmdLine[8],$CmdLine[9])
 ;El código de los casos bf-dic y r-dic-bf difieren en algunos
aspectos de este último.
```

7.5.1 Función *BruteForce*

Esta función configura por completo el ataque de fuerza bruta y lo añade a la lista de ataques a realizar. En las imágenes siguientes se puede ver como se realiza esta configuración a través de la interfaz gráfica:

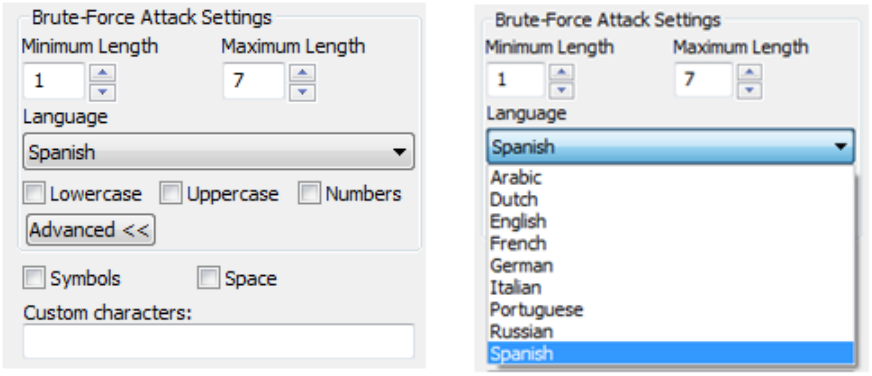


Fig. 25 – Opciones de configuración de un ataque de fuerza bruta

Como se puede ver en las imágenes es necesario indicar la longitud mínima, la longitud máxima, el idioma, y los tipos de caracteres a utilizar para configurar el ataque. Para que los parámetros pasados por consola no sean muy numerosos se ha implementado la selección de los tipos de caracteres mediante una máscara de bits. Teniendo en cuenta que hay 5 casillas además de la opción 'Custom Characters', se utilizará un número de 6 bits indicando que casillas deben activarse y cuáles no. Si el bit que activa el uso de 'Custom characters' está a uno, se leerá el contenido del archivo *CustomBruteForce.txt* para indicar los caracteres personalizados. Por tanto los parámetros que se le deben pasar a la función *BruteForce* como **argumentos** son la **longitud mínima, la longitud máxima, el idioma y la máscara de bits** que será un número decimal comprendido entre 1 y 63. Los bits activan las casillas según el orden en el que se encuentran las opciones en la imagen: 'Lowercasse', 'Uppercase', 'Numbers', 'Symbols', 'Space', 'Custom characters'.

A continuación se muestran algunas partes del código de esta función:

- Código que selecciona las opciones indicadas en la máscara:

```
For $i = 1 to 6
   If BitAND ( $aux, 1)==1 Then
         ControlCommand ( "", "Lowercase", 2061, "Check", "")
         $aux = $aux -1
   ElseIf BitAND ( $aux, 2)==2 Then
         ControlCommand ( "", "Uppercase", 2062, "Check", "")
         $aux = $aux -2
   ElseIf BitAND ( $aux, 4)==4 Then
         ControlCommand ( "", "Numbers", 2063, "Check", "")
         $aux = $aux -4
```

```
   ElseIf BitAND ( $aux, 8)==8 Then
         ControlClick("Passware Password Recovery Kit Forensic",
   "Advanced >>", 2067)
         ControlCommand ( "", "Lowercase", 2064, "Check", "")
         $aux = $aux -8
   ElseIf BitAND ( $aux, 16)==16 Then
         ControlClick("Passware Password Recovery Kit Forensic",
   "Advanced >>", 2067)
         ControlCommand ( "", "Lowercase", 2065, "Check", "")
         $aux = $aux -16
   ElseIf      BitAND( $aux, 32)==32 Then
    ; Opción Custom -- Abrir el archivo CustomBruteForce.txt
   comprobando errores de apertura y lectura.
Next
```

- Selección del idioma, se sitúa el foco en el control 'Language' y se pulsa la letra inicial del idioma a seleccionar:

```
Func BruteForce($language,$min,$max,$options)
ControlTreeView ( "Passware Password Recovery Kit Forensic", "Drag
attacks from this list:", 2001, "Select", "#0|Brute-force")
ControlClick("Passware Password Recovery Kit Forensic", "", 2002) ;
pasar a la izquierda el seleccionado
WinWaitActive("Passware Password Recovery Kit Forensic")
ControlClick("Passware Password Recovery Kit Forensic", "", 2059)
ControlClick("Passware Password Recovery Kit Forensic", "", 2059)
   Switch $language
         Case "arabic"
               Send("a")
         Case "dutch"
               Send("d")
         Case "english"
               Send("e")
         Case "french"
               Send("f")
         Case "german"
               Send("g")
         Case "italian"
               Send("i")
         Case "portuguese"
               Send("p")
         Case "russian"
               Send("r")
         Case "spanish"
               Send("s")
   EndSwitch
```

7.5.2 Función *Dictionary*

La función *Dictionary* al igual que en el caso de fuerza bruta necesita configurar tanto el idioma como la longitud mínima y máxima de las claves a probar, en este caso además existe la posibilidad de fijar reglas para que las claves generadas en el ataque por diccionario coincidan con los patrones impuestos por 'Pattern'.

En las imágenes siguientes puede verse cómo se realiza la configuración a través de la interfaz gráfica:

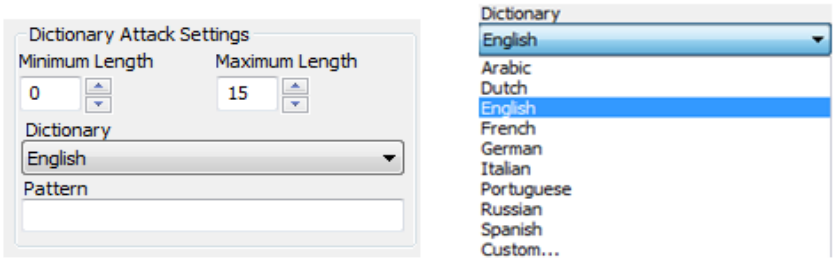


Fig. 26 – Opciones de configuración de un ataque por diccionario

Si se elige utilizar un diccionario personalizado, tras seleccionar 'Custom…' se deberá interactuar con la ventana del compilador de diccionarios mostrada a continuación:

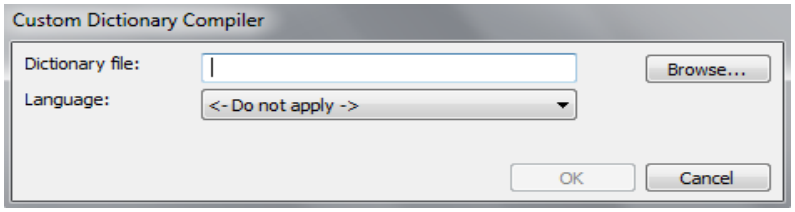


Fig. 27 – Compilador de diccionarios de Passware

Si al lanzar el *script* con un ataque de diccionario se indica la ruta de un diccionario en vez del idioma, este se compilará en *Passware* y se utilizará para realizar el ataque. En las siguientes líneas de código puede verse como se realizan estas acciones:

```
Case "spanish"
   Send("s")
Case Else
;Si el idioma no coincide se trata de la ruta de un diccionario
personalizado, la ruta se encuentra en el argumento $language:
   Send("c") ; Custom
   WinWaitActive("Custom Dictionary Compiler")
   ControlSetText("Custom Dictionary Compiler","",2135,$language,1)
   sleep(200)
   ControlClick("Custom Dictionary Compiler","OK", 1)
   sleep(200)
   WinWaitActive("Passware Password Recovery Kit Forensic")
EndSwitch
```

En el caso de habilitar la opción 'pattern' se leerá el contenido del archivo de texto Pattern.txt que deberá estar ubicado en el mismo directorio que los *scripts* de AutoIT. Código resumido:

```
$file2 = FileOpen("Pattern.txt", 0)
; Check if file opened for reading OK
If $file2 = -1 Then
; Registar y mostrar errores
EndIf
$line2 = FileReadLine($file2)
If @error = -1 Then
; Registar y mostrar errores
EndIf
ControlSetText("Passware Password Recovery Kit Forensic","Pattern",
2056,$line2,1)
FileClose($file2)
```

Los argumentos que utiliza esta función y que se le pasarán mediante la línea de órdenes son la longitud mínima, la longitud máxima y el idioma o ruta del diccionario.

7.6 Comprobar el estado del ataque

Una vez lanzado el ataque de descifrado se podrá comprobar el estado de este mediante el script CheckEnd.au3. Este *script* devolverá el tiempo estimado para completar el ataque en su totalidad o, si el ataque ya terminó, guardará los resultados en la carpeta *Results* e indicará por consola el nombre del archivo que contiene toda la información del resultado.

Es necesario por tanto distinguir dos situaciones; en la primera el descifrado está en proceso y en la segunda el proceso terminó y se muestran los resultados por pantalla:

Fig. 28 – Situaciones posibles tras lanzar el descifrado. A) En proceso. B) Terminado

Cuando el descifrado está en curso, existen tres botones para interactuar con el programa, 'Skip Attack', 'Pause' y 'Stop', mientras que cuando el ataque ha finalizado estos controles desaparecen. En el siguiente código se comprueba la existencia del botón 'Stop' y se actúa de una manera u otra en función del resultado:

```
WinActivate ( "Passware Password Recovery Kit Forensic" )
If ControlCommand("Passware Password Recovery Kit Forensic",
"Stop", 2013, "IsEnabled", "") Then ;El botón stop existe
;Mostrar información por consola y registrarla en el log.
   Exit 1
Else
   ControlClick("Passware Password Recovery Kit Forensic", "",
   "[CLASS:ToolbarWindow32; INSTANCE:1]", "Primary", 1, 300, 20 )
   ;Guardar resultados
   ;Send("^c");
   $name = "PassResults"&"-"&@MDAY&"-"&@MON&"-"&@YEAR&"_"&@HOUR&"-
   "&@MIN
   $path = @ScriptDir&"\Results\"
   $path = $path&$name
   WinWaitActive("Guardar como")
   ControlSetText("Guardar como","",1148,$path,1)
   Send("!g")
;Mostrar información por consola y registrarla en el log.
   Exit 0
EndIf
```

7.7 Reanudación del proceso de descifrado

Passware muestra una ventana emergente que permite continuar con el proceso de recuperación anterior si en el momento de cerrarlo se estaba ejecutando la herramienta de descifrado o si no se encontró la contraseña del archivo en cuestión.

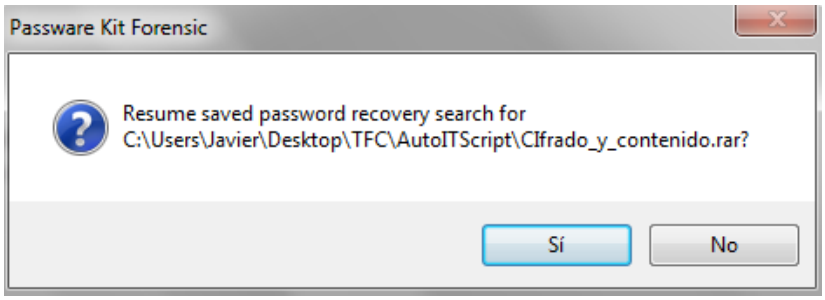


Fig. 29 – Ventana emergente para recuperar el proceso anterior

El *script* Restore.au3 interactúa con esta ventana emergente respondiendo de forma afirmativa en caso de que exista. Cabe mencionar que los *scripts* Decrypt.au3 y FindProtected.au3 también interactúan con esta ventana cuando aparece pero respondiendo negativamente.

A continuación se comenta ligeramente la estructura de este *script*, incluyendo algunas líneas de código:

```
;Lanzar Passware y esperar que arranque:
Run("C:\Program Files (x86)\Passware\Passware
Kit\PasswareKitForensic.exe")
WinWaitActive("Passware Password Recovery Kit Forensic")
;La ventana emergente puede tardar algunos milisegundos en
aparecer, se introduce un retardo:
Sleep(100)
;Función WinExists para comprobar si aparece la ventana emergente:
$a = WinExists ("Passware Kit Forensic", "Resume saved password
recovery search for")

If $a==1 Then
; Aceptar y mostrar información por consola.
Else
; Comunicar que no existe un proceso reanudable.
EndIf
```

7.8 Cierre del programa Passware

Como ya se comentó, *Passware* es un programa bastante maduro que lleva muchos años siendo desarrollado y mejorado. Esto es una desventaja a la hora de automatizar su uso mediante la programación de eventos ya que cualquier situación que no esté contemplada en los *scripts* provocará la aparición de alguna ventana emergente o de algún control con el que sea necesario interactuar para proseguir. Debido a esto es necesario que los *scripts* aquí desarrollados se ejecuten en un orden lógico, por ejemplo después de lanzar el *script* de descifrado habrá que lanzar el *script CheckEnd* y no

confundirse con el *script CheckSearchEnd* puesto que de ser así se quedará el proceso *Passware* en ejecución y el script pausado indefinidamente. Esta y otras situaciones pueden ir dejando numerosos procesos *Passware* ejecutándose al mismo tiempo, por ello es necesario crear un script que cierre las instancias del programa *Passware*. En este caso el *script* es bastante ligero por lo que se muestra listado por completo:

```
$array = ProcessList ("PasswareKitForensic.exe")
If $array[0][0]==0 Then
    ConsoleWrite("There is no Passware process running"& @CRLF)
  $fileLog = FileOpen("log.txt", 1)
  FileWriteLine($fileLog,@MDAY&"/"&@MON&"/"&@YEAR&"-
"&@HOUR&":"&@MIN&" There is no Passware process running"& @CRLF)
  FileClose($fileLog)
  Exit 1
 EndIf

WinActivate ("Passware Password Recovery Kit Forensic")
WinClose("Passware Password Recovery Kit Forensic")
WinWaitActive("Passware Kit Forensic", "Do you really want to
exit?")
Sleep(100)
Send("!s")
$array = ProcessList ("PasswareKitForensic.exe")
ConsoleWrite("Passware closed. Total Passware processes found:
"&$array[0][0]& @CRLF)
$fileLog = FileOpen("log.txt", 1)
FileWriteLine($fileLog,@MDAY&"/"&@MON&"/"&@YEAR&"-
"&@HOUR&":"&@MIN&" Passware closed. Total Passware processes found:
"&$array[0][0]& @CRLF)
FileClose($fileLog)
Exit 0
```

El *script* Close.au3 sólo cierra un proceso *Passware* cada vez por lo que habrá que ejecutarlo tantas veces como procesos *Passware* haya en ejecución. Para facilitar esta tarea el *script* devuelve por consola el número de procesos passware.exe encontrados antes de cerrar cualquiera de ellos.

7.9 Utilización de los *scripts* en un nuevo sistema

Para la utilización de estos *scripts* en un nuevo ordenador es necesario llevar a cabo una serie de operaciones para que funcionen correctamente.

En primer lugar se debe instalar el software AutoIT v3 que puede obtenerse de su página oficial http://www.autoitscript.com/site/autoit/ .

Una vez instalado el software AutoIT será necesario modificar los *scripts* que lanzan la ejecución del programa de descifrado para que tengan la ruta correcta de *Passware*. Estos *scripts* son Decrypt.au3, Restore.au3 y FindProtected.au3; todos ellos lanzan la ejecución del programa *Passware* con la siguiente línea de código:

```
; Ruta dependiente del PC, cambiar en caso necesario:
Run("C:\Program Files (x86)\Passware\Passware
Kit\PasswareKitForensic.exe")
```

La ruta deberá coincidir con el lugar de instalación del programa *Passware* en los tres *scripts* antes mencionados.

Una vez modificada esta ruta es necesario compilar los *scripts* como aplicaciones de consola para ello se deben utilizar las herramientas de compilación proporcionadas por AutoIT:

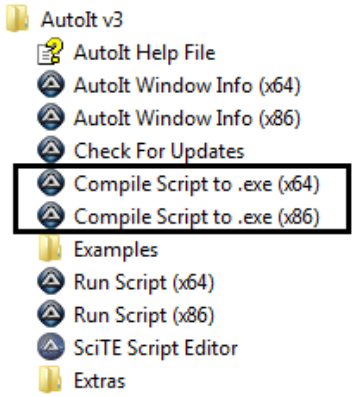


Fig. 30 – Herramientas de compilación de AutoIT

Una vez que se tienen los *scripts* compilados, el directorio donde se encuentran los scripts debería tener un aspecto similar al de la imagen:

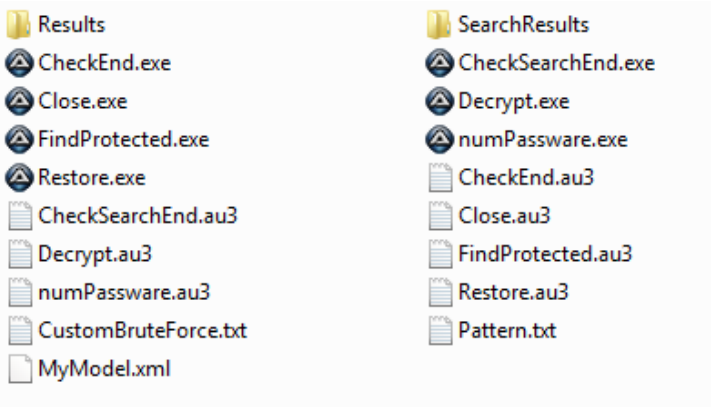


Fig. 31 – Estructura de directorios y archivos necesaria para los scripts

Si alguno de los archivos de la imagen no existe, pueden ocurrir errores inesperados en la ejecución de los *scripts*.

Una vez realizadas estas operaciones se está en condiciones de utilizar los *scripts* en el nuevo sistema de descifrado ya sea un *cluster* de tarjetas GPU, un equipo individual o varios nodos distribuidos.

8

Plataformas de Virtualización: *VGA-Passthrough*

Hasta hace poco la mayoría de sistemas de virtualización que permitían características gráficas se basaban en la emulación por software de una tarjeta de video integrada, por lo que en ningún momento se le permitía al sistema huésped el acceso a la tarjeta gráfica real. Esto suponía una limitación para los sistemas virtualizados ya que no podían utilizar las altas prestaciones que ofrecía el hardware de gráficos y por lo tanto el software que este era capaz de soportar estaba limitado por las características de sus dispositivos emulados.

La característica *VGA-Passthrough* rompe con esta tendencia y permite la utilización de la tarjeta gráfica nativa por las máquinas huésped (*guest*) para su utilización tanto en programas de altas prestaciones gráficas como en programas de seguridad que requieran la utilización de los núcleos GPU para incrementar su eficiencia y rendimiento.

El interés que se tiene en este proyecto sobre este tipo de virtualización con hardware nativo reside sobre todo en su utilidad para la creación de un *cluster* con varias tarjetas GPU que funcione bajo un sistema operativo ligero y de fácil adaptación para *clusters* como puede ser Linux y que sin embargo utilice un programa de seguridad para descifrar documentos basado en Windows (que suelen incluir más características al ser programas comerciales más completos).

8.1 Comparativa entre entornos de virtualización. CPU vs GPU.

Se ha realizado una tabla comparativa en la que se puede observar como varía el rendimiento de los programas utilizando diferentes entornos de virtualización, así como la notable mejora que se puede obtener utilizando la GPU.

Los entornos de virtualización utilizados son tres de los más conocidos: VMware, VirtualBox y KVM. Los programas de seguridad utilizados son por un lado el software comercial *Passware Kit Forensic* que es capaz de trabajar con una gran variedad de archivos cifrados, y por otro lado el programa de uso gratuito *cRARk* que sólo trabaja con archivos comprimidos RAR utilizando la GPU.

El rendimiento se muestra en forma de número de claves por segundo que son capaces de generar y probar cada uno de los programas elegidos al intentar descifrar un archivo RAR cifrado con AES-256 que es el archivo más robusto de los ya comentados al inicio. Para más detalles ver el apartado "3 Estado del Arte de la Protección de Archivos Digitales"

Todas las pruebas han sido realizadas en un PC Intel Quad Core CPU Q6600 a 2.4 GHz.

8.1.1 Resultados utilizando solo CPU:

Utilizando únicamente procesado CPU se obtienen los siguientes resultados:

	Crark	**Passware Kit Forensic**
SO Linux	106 psw/s	----
SO Windows	106 psw/s	(Windows 7) 220 psw/s
Huésped en VMware	101 psw/s	206-225 psw/s*
Huésped en VirtualBox	90 psw/s	200-248 psw/s*
Huésped en KVM	100 psw/s	103 psw/s**

* = Al utilizar 4 procesadores el valor de passwords/segundo es variable y nunca llega a estabilizarse.

** = En el caso de KVM el sistema huésped solo ha identificado 2 procesadores de los 4 que se le asignaron, por lo tanto su velocidad de proceso es la mitad que en los otros casos.

Hay que tener en cuenta que el programa Crark es gratuito y sólo utiliza un procesador para descifrar los archivos mientras que el programa "Passware Kit Forensic" es comercial y permite utilizar tantos procesadores como estén disponibles , en este caso utiliza los 4 procesadores.

8.1.2 Observaciones:

Atendiendo a los resultados obtenidos, se pueden sacar varias conclusiones:

- El programa Crark prueba solo unas 100 contraseñas por segundo frente a las 200 por segundo que prueba *Passware*. No obstante hay que tener en cuenta que Crark tan sólo utiliza un núcleo CPU mientras que *Passware* utiliza los 4 disponibles. Esto parece indicar que el código del programa Crark es más eficiente que el del programa comercial tal y como se vió en el apartado 5.2.3.1.

- Se puede observar que la diferencia de rendimiento entre los distintos entornos de virtualización es despreciable.

- La diferencia de rendimiento entre las máquinas virtuales y la máquina real es insignificante, incluso parece mejorar cuando se utiliza un entorno virtualizado. Esto se

debe a que todos los entornos de virtualización probados son capaces de utilizar las instrucciones extendidas de virtualización[36] de los nuevos procesadores del mercado.

8.1.3 Resultados utilizando GPU:

La nueva tecnología GPGPU[37] permite utilizar tarjetas gráficas como unidades de proceso. En nuestro caso podemos utilizar esta tecnología para ejecutar de una manera mucho más rápida los algoritmos de ataque que utilizan los programas *Passware* y Crark.

Las pruebas se han realizado utilizando como sistemas operativos Windows 7 y Linux Debian. La tarjeta gráfica utilizada en las pruebas es una nVIDIA Geforce GTX 460 con 1GB de memoria.

Tabla comparativa:

	Crark	**Passware Kit Forensic**
Sistema Host (Linux Debian)	2302 psw/s	----
Sistema Host (Windows 7)	2182 psw/s	1200-1480 psw/s
Huésped en VMware	101 psw/s	206-225 psw/s
Huésped en VirtualBox	90 psw/s	200-248 psw/s
Huésped en KVM	100 psw/s	103 psw/s

Tras instalar la tarjeta gráfica se han repetido las pruebas del punto 8.1.1 tanto en VMware como en VirtualBox y KVM, sin embargo, ninguna de ellas es capaz de capturar la tarjeta GPU para utilizarla en los huéspedes virtuales por lo que el rendimiento obtenido es exactamente el mismo que antes.

8.1.4 Conclusiones:

Utilizando la tarjeta gráfica hemos conseguido aumentar en un orden de magnitud el rendimiento de los programas, es decir, si quisiéramos obtener el mismo rendimiento que conseguimos con una sola tarjeta gráfica utilizando CPU's deberíamos tener al menos 10 PC's con 4 procesadores cada uno trabajando al mismo tiempo.

Queda de manifiesto la importancia de utilizar las tarjetas gráficas en las máquinas virtuales ya que de esta forma se podrían utilizar las herramientas de Windows y de Linux en un mismo sistema aprovechando la mejora de rendimiento de las tarjetas gráficas en cualquiera de las herramientas de seguridad.

---

[36] Tecnologías IVT (Intel Virtualization Technology) o AMD-V (AMD Virtualization)).

[37] GPGPU o General-Purpose computation on Graphics Processing Units: Las unidades de procesamiento gráfico (GPUs) son procesadores con multitud de núcleos de alto rendimiento que pueden utilizarse para acelerar el proceso de un amplio número de aplicaciones.

8.2 Rendimiento al utilizar múltiples GPUs:

Una vez visto el rendimiento que proporciona una tarjeta gráfica frente al habitual rendimiento de la CPU cabe preguntarse si es factible la realización de un *cluster* con varias tarjetas GPU funcionando al mismo tiempo. El problema que se puede dar es que se forme un cuello de botella en el bus PCI-express al instalar varias tarjetas gráficas en la misma placa base limitándose así el rendimiento total del *cluster*. Para comprobar si existe o no este problema se han realizado una serie de pruebas utilizando el programa *Passware* en una máquina con dos tarjetas gráficas nVIDIA GTX-460 y con un procesador Intel Core i7.

8.2.1 Ataque a un archivo de Word cifrado con AES-128 bits:

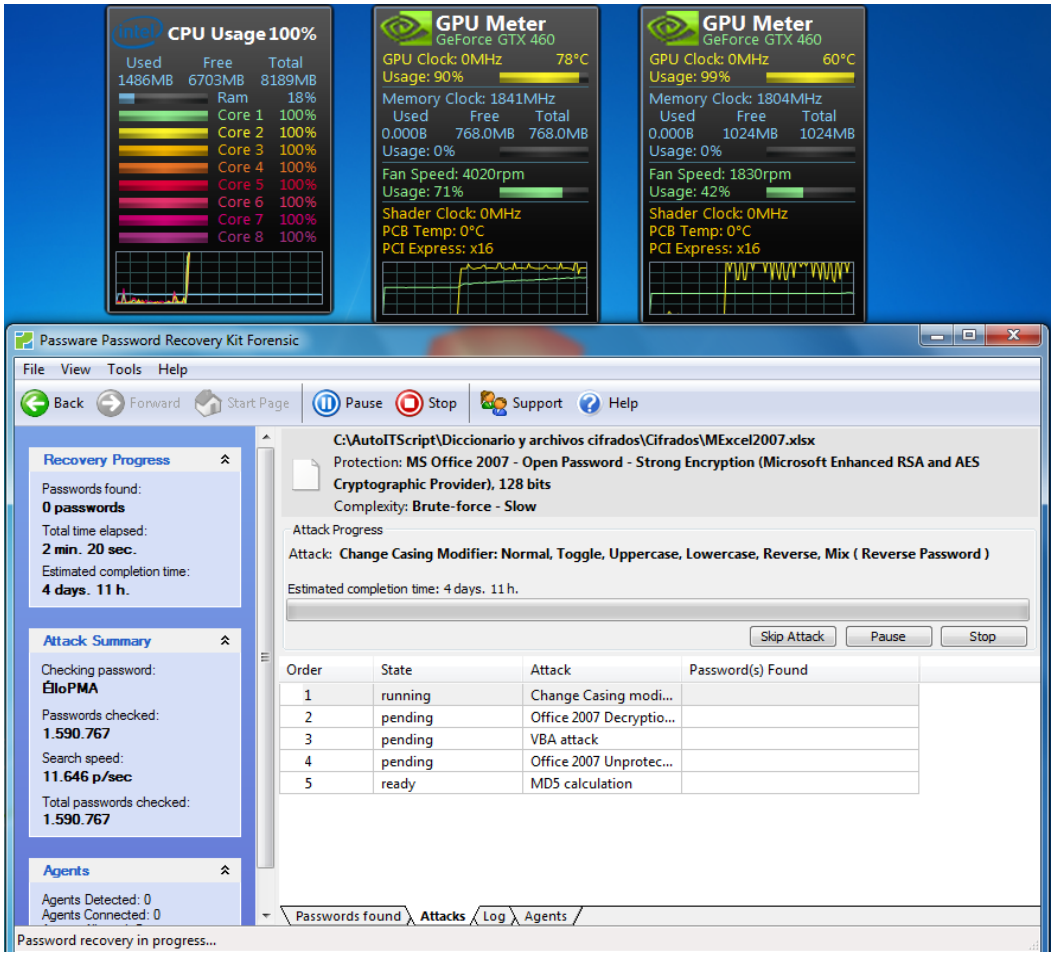


Fig. 32 – Prueba de descifrado docx

Como se puede ver en la imagen, el número de claves por segundo es de 11646, que es prácticamente el doble de lo que se obtuvo con una sola tarjeta GPU (aproximadamente 6200 psw/s) por lo que se puede afirmar que no se produce ningún cuello de botella en el bus PCI. Si es cierto, sin embargo, que se pierde algo de rendimiento debido a la gestión que debe hacer *Passware* para poder utilizar la capacidad de las dos tarjetas gráficas. Nótese que en la figura anterior sólo una de las GPUs está a su máxima capacidad mientras que la otra se encuentra solo al 90%.

8.2.2 Ataque a un archivo WinRAR con cifrado AES-256 bits:

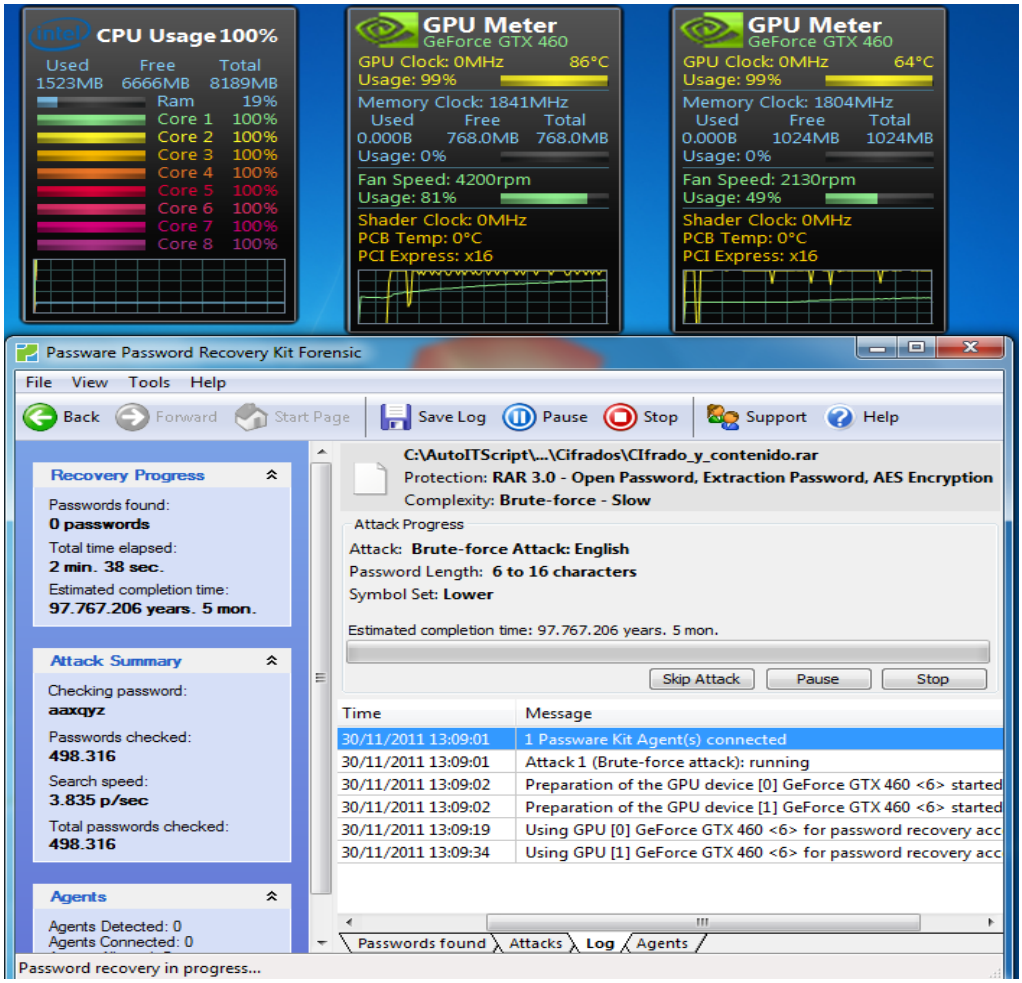


Fig. 33 – Prueba de descifrado RAR

En el caso de WinRAR las dos GPUs si funcionan a máxima potencia, sin embargo, al ser este un tipo de archivos más robusto el número de contraseñas por segundo disminuye considerablemente. Para más detalles sobre el cifrado en archivos RAR se recomienda consultar el apartado 3.5.

8.2.3 Descifrado de dos archivos utilizando dos instancias de Passware al mismo tiempo:

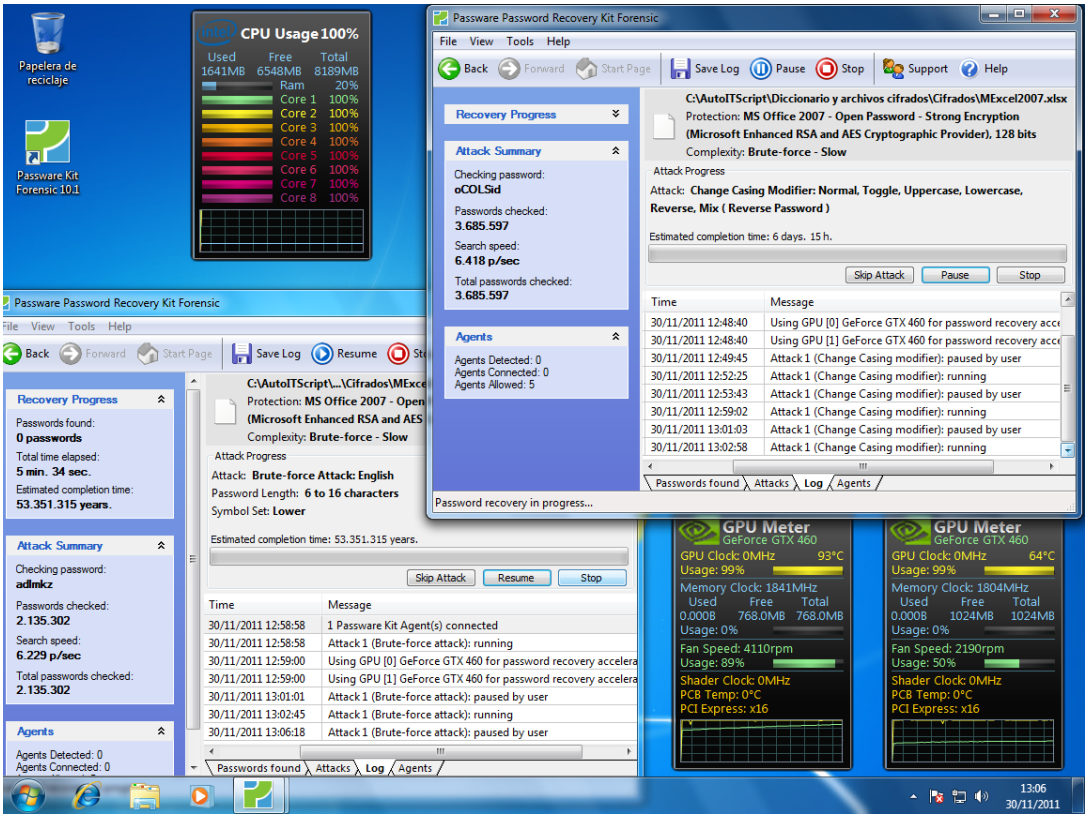


Fig. 34 – Prueba con dos instancias

Al utilizar dos instancias lo que se consigue es que las dos GPUs se utilicen al máximo en todo momento de esta forma no se desaprovecha nada de capacidad mientras dura el descifrado. Si se realizara un ataque por diccionario ante un mismo archivo cifrado, lo más eficaz sería dividir el diccionario en un número de partes igual al número de tarjetas disponibles y lanzar una instancia para cada una de las partes del diccionario obteniendo así el máximo rendimiento de las tarjetas ya que todas estarían ocupadas al 100%.

Un aspecto importante a tener en cuenta a la hora de pensar en la realización de un *cluster* con tarjetas gráficas es la necesidad de un buen sistema de refrigeración para contrarrestar las altas temperaturas que las GPUs llegan a alcanzar cuando trabajan a máxima capacidad.

9

Plataforma de Virtualización XEN

XEN es una plataforma de virtualización pionera en la utilización de *VGA-Passthrough*, Hay que tener en cuenta, no obstante, que esta característica aún no está lo suficientemente madura por lo que no es posible virtualizar cualquier tipo de tarjeta gráfica sino solo algunas específicas como se verá más adelante. Se espera que en el futuro se cubra una mayor gama de tarjetas.

A continuación, se explica detalladamente la estructura interna de XEN así como los principios de funcionamiento que hacen posible la utilización de esta característica.

Un entorno virtual de XEN consta de diferentes elementos que trabajan conjuntamente:

- *Hypervisor* XEN.
- *Domain 0 Guest* (*Dom 0*)
    - *Domain Management and Control*
- *Domain U Guest* (*Dom U*)
    - *PV Guest* (huésped PV Dom U)
    - *HVM Guest* (huésped HVM Dom U)

El diagrama básico de organización de estos componentes se resume en la siguiente imagen:

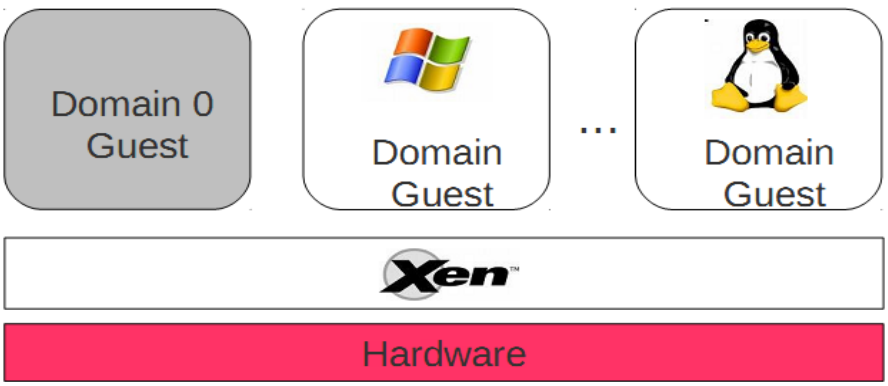


Fig. 35 – Diagrama básico Xen

9.1 Descripción de los elementos de Xen

9.1.1 *Hypervisor* XEN

El *hypervisor* de Xen es una capa básica de abstracción software que se encuentra ubicada justo encima del hardware que todo sistema operativo utiliza. Es el encargado de planificar la CPU y el particionado de memoria de las distintas máquinas virtuales que se ejecutan sobre el hardware nativo. El *hypervisor* no solo abstrae el hardware de las máquinas virtuales sino que también controla la ejecución de éstas ya que el entorno de procesos se comparte entre todas. El *hypervisor* no tiene conocimiento de los dispositivos de almacenamiento externo, de las operaciones de red ni en general de cualquier dispositivo de entrada/salida.

9.1.2 *Domain* 0

El *Domain 0*, que no es sino un kernel de Linux modificado es una máquina virtual única que corre sobre el hypervisor de Xen y que posee privilegios especiales para acceder a los recursos físicos de entrada/salida así como para interactuar con otras máquinas virtuales corriendo en el sistema (llamadas *Domain U*: *PV and HVM Guests*). Todos los entornos de virtualización Xen requieren un *Domain 0* funcionando antes de que cualquier otra máquina virtual pueda ejecutarse.

En el Domain 0 se incluyen 2 controladores para soportar operaciones de red y peticiones de acceso al disco desde los *Domain U* tanto PV (paravirtualizados) como HVM (*Hardware virtual machine* o *fully virtualized machines*). Estos controladores son el "Network Backend Driver" y el "Block Backend Driver".

El "Network Backend Driver" se comunica directamente con el hardware de red para procesar todas las peticiones de las distintas máquinas virtuales o huéspedes *Domain U*.

El "Block Backend Driver" se comunica con el disco de almacenamiento local para responder a las peticiones de lectura/escritura que los *Domain U* generan.

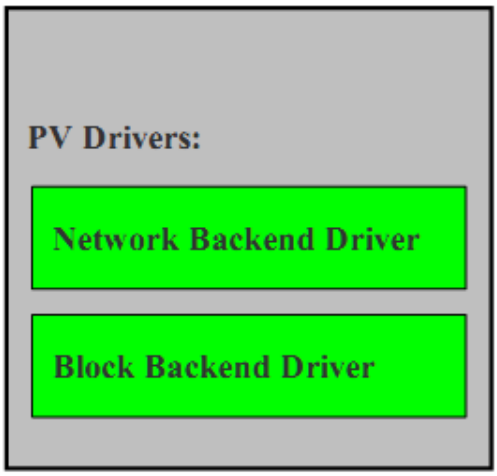


Fig. 36 – *Drivers Domain 0*

9.1.3 *Domain U*

El huésped *Domain 0* tiene acceso directo al hardware de la máquina mientras que los *Domain U* no tienen acceso alguno y necesitan del *Domain 0* para acceder a él. Podemos distinguir entre dos tipos de *Domain U*: paravirtualizado o virtualización completa.

Las **máquinas virtuales paravirtualizadas** (*PV guest*) son conscientes de que no poseen acceso directo al hardware y reconocen a otras máquinas virtuales corriendo en el mismo equipo mientras que las máquinas de virtualización completa (*HVM guest*) no son conscientes de que comparten tiempo de proceso en el *hardware* ni tampoco son conscientes de que otras máquinas están ejecutándose simultáneamente.

Todo huésped *Domain U PV* contiene dos controladores: *PV Network Driver* y *PV Block Driver*.

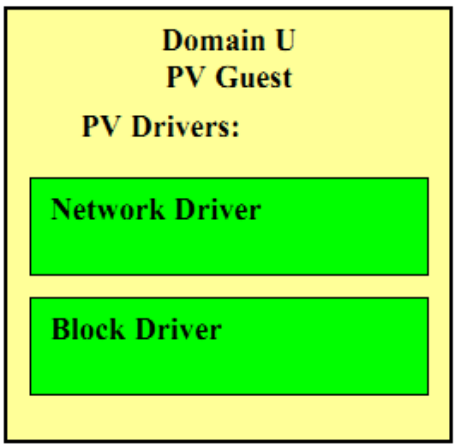


Fig. 37 – *Drivers Domain U*

Estos controladores confieren a la máquina virtual capacidades para acceder al disco y a la red, respectivamente.

Por el contrario, un *Domain U HVM* no dispone de estos controladores de paravirtualización dentro de la máquina virtual, sino que posee un demonio especial que se inicia para cada uno de los *HVM guest* dentro del *Domain 0* llamado *Qemu-dm*. *Qemu-dm* soporta las operaciones de red y las peticiones de acceso al disco de cada *Domain U HVM guest*.

Todo huésped HVM debe iniciarse como si se tratara de un sistema aislado por lo que es necesario añadir a este tipo de máquinas virtuales software adicional para emular la BIOS que un sistema operativo espera en el arranque. Esta BIOS emulada recibe el nombre de "Xen virtual firmware".

El esquema de funcionamiento de los *HVM guest* se muestra en la siguiente figura:

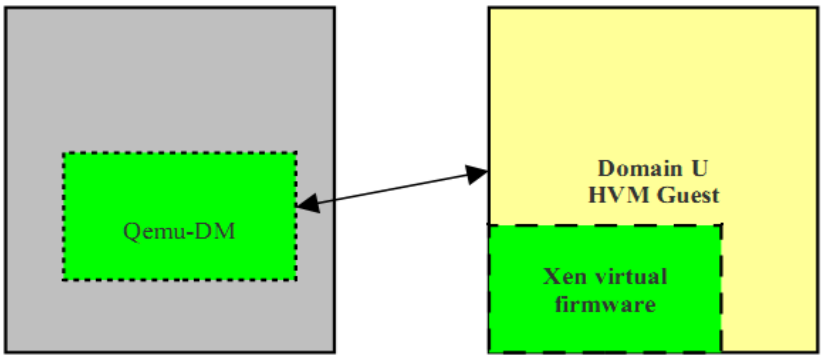


Fig. 38 – Funcionamiento de *HVM Guest*

9.1.4 Domain Management y Control

Se conoce como "Domain Management and Control" a los servicios que soportan toda la administración y control del entorno de virtualización y se encuentran ubicados en el interior de la máquina virtual principal, el *Domain 0*. En los diagramas que aparecen en los apartados siguientes, se muestran estos demonios fuera del *Domain 0* para un mejor entendimiento de la arquitectura.

9.1.5 Xend

Xend es un demonio escrito en python que se considera el administrador del sistema en un entorno Xen. Aprovecha la librería "libxenctrl" (ver más abajo) para realizar peticiones al *hypervisor* Xen. Todas las peticiones procesadas por el demonio Xend son entregadas al hypervisor a través de la interfaz XML RPC que proporciona la herramienta Xm.

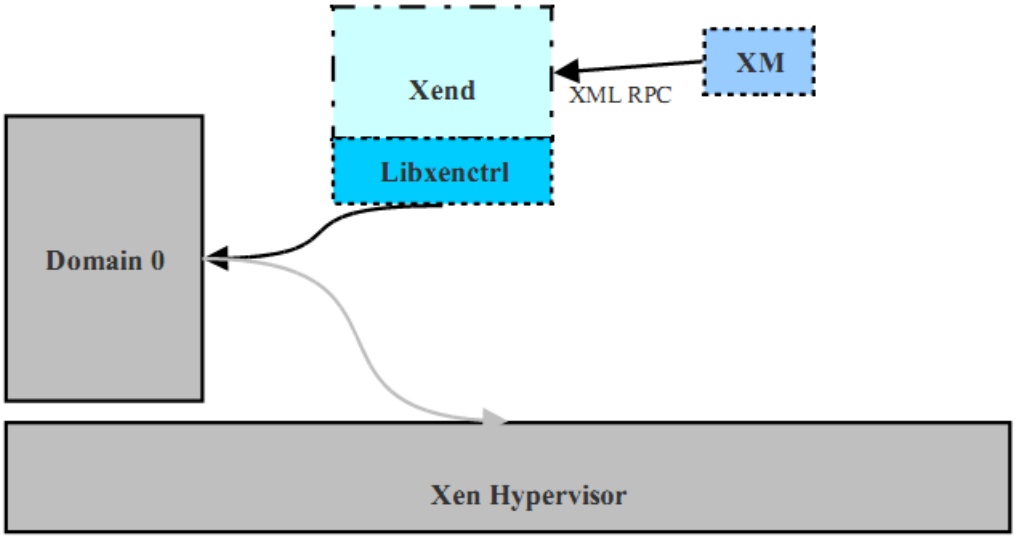


Fig. 39 – Funcionamiento Xend

9.1.6 Xm

Xm es la herramienta de línea de órdenes que recoge la entrada del usuario y la transmite al demonio Xend vía XML RPC.

9.1.7 Xenstored

Xenstored es un demonio que mantiene un registro de información, incluyendo memoria y enlaces al canal de eventos (*event channel links*), entre el *Domain 0* y todas las demás máquinas huésped *Domain U guest*. La máquina virtual *Domain 0* aprovecha este registro para configurar canales de dispositivos con otras máquinas virtuales en el sistema. Para entender mejor el funcionamiento del demonio Xend se recomienda consultar el apartado "Comunicación entre Domain 0 y Domain U".

9.1.8 Libxenctrl

Libxenctrl es una librería escrita en C que proporciona a Xend la habilidad de *hablar* con el hypervisor Xen a través del *Domain 0*. Un controlador especial incluido en el *Domain 0* llamado "privcmd" se encarga de entregar esta petición al *hypervisor*.

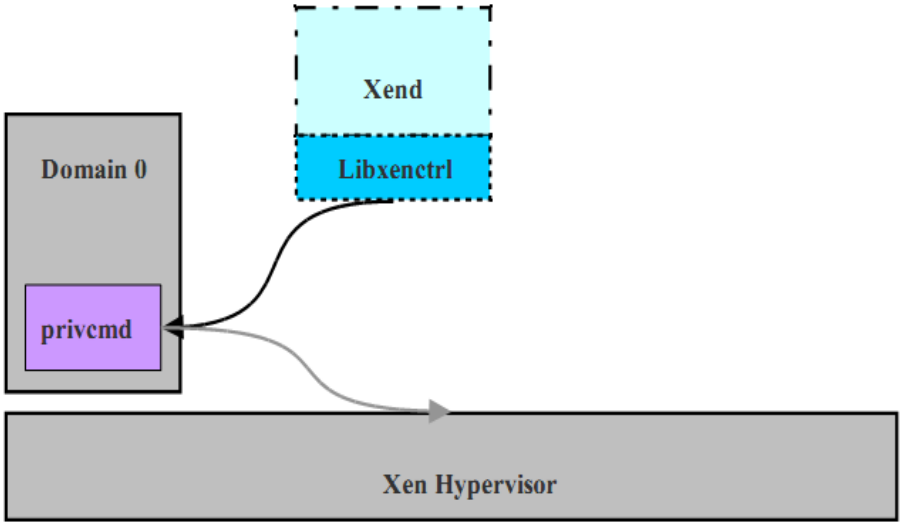


Fig. 40 – Comunicación entre libxenctrl y privcmd

9.1.9 Qemu-dm/Stub-dm

En versiones anteriores de Xen, cada *HVM Guest* dentro de un entorno Xen necesitaba su propio demonio Qemu. Esta herramienta se encargaba de todas las operaciones de red y peticiones de disco desde el *DomainU HVM Guest* para permitir la existencia de máquinas del tipo *fully virtualized* en un entorno Xen. Actualmente no se necesita un demonio Qemu para cada máquina huésped sino que se utiliza la herramienta stub-domains o stub-dm para proporcionar un conjunto de servicios disponibles para todos los huéspedes

Domain U HVM. Esta herramienta debe existir fuera del *hypervisor* de Xen debido a su necesidad de acceder a dispositivos de entrada/salida y a dispositivos de red. Por ello qemu-dm/stub-dm se encuentra incluido en el *Domain 0*.

9.1.10 Xen *Virtual Firmware*

El Xen *Virtual Firmware* es una BIOS virtual que se inserta en cada uno de los huéspedes *Domain U HVM* para asegurar que el sistema operativo recibe todas las instrucciones de arranque que tienen lugar durante un arranque normal de una máquina, proporcionando el entorno software de un PC compatible.

9.2 Funcionamiento de *Xen*

Esta sección muestra como una máquina *Domain U* paravirtualizada (PV) es capaz de comunicarse con redes externas o dispositivos de almacenamiento a través del Xen *Hypervisor* y del *Domain 0*.

9.2.1 Comunicación entre *Domain 0* y *Domain U*

Como se describió anteriormente, el *hypervisor* de Xen no soporta operaciones de red o peticiones de disco, por lo tanto un huésped paravirtualizado *Domain U* debe comunicarse con el *Domain 0* a través del *hypervisor* para poder realizar este tipo de tareas. Como ejemplo, vamos a describir cómo un *Domain U PV* escribe datos en el disco duro.

Cuando el controlador de dispositivos tipo bloque o *block driver* de un huésped *Domain U* paravirtualizado (abreviado *PV Dom U*) recibe una petición para escribir en el disco, la petición se procesa y se escriben los datos a través del *hypervisor* Xen en la posición de memoria local correspondiente (que se comparte con el *Domain 0*). Existe un canal de eventos entre el *Domain 0* y el huésped *PV Dom U* que les permite comunicarse mediante interrupciones asíncronas del *hypervisor* Xen.

El *Domain 0* recibirá la interrupción del hypervisor Xen provocando que el controlador *PV Block Backend* acceda al sistema de memoria local para leer los bloques apropiados de la memoria compartida con el huésped. Después esta información se escribirá al disco duro local en una posición específica.

En la siguiente imagen, el canal de eventos se muestra como un enlace directo entre el dominio principal y el huésped. Esta imagen, sin embargo, es una versión simplificada de cómo el sistema trabaja internamente. De hecho, el canal de eventos funciona a través del *hypervisor* Xen con interrupciones específicas registradas en el demonio Xenstored permitiendo a ambos dominios compartir de manera rápida información a través de la memoria local.

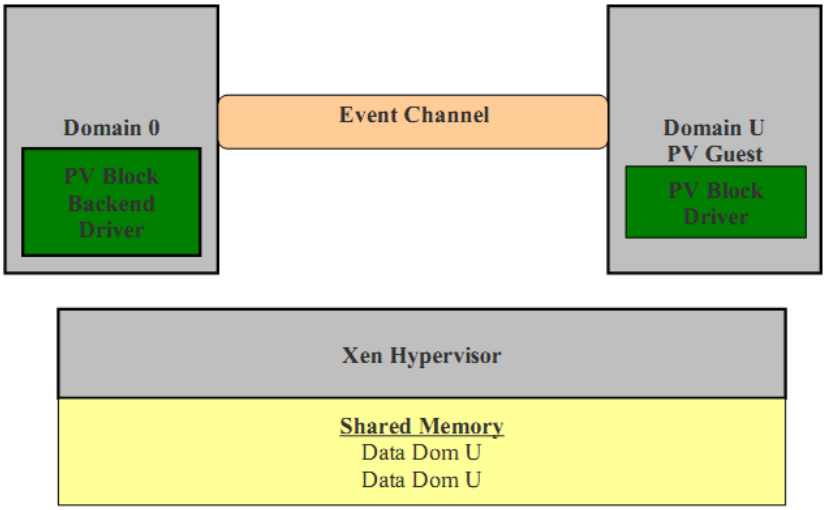


Fig. 41 - Versión simplificada del funcionamiento interno de Xen

9.3 Xen PCI Passthrough y VGA Passthrough

9.3.1 *PCI Passthrough*

Entre las características más interesantes de Xen se encuentran *PCI-Passthrough* y *VGA-Passthrough*. *PCI Passthrough* permite a los huéspedes *Domain U* tener acceso directo al hardware local sin utilizar de intermediario al *Domain 0*.

El diagrama siguiente muestra cómo funciona esta característica:

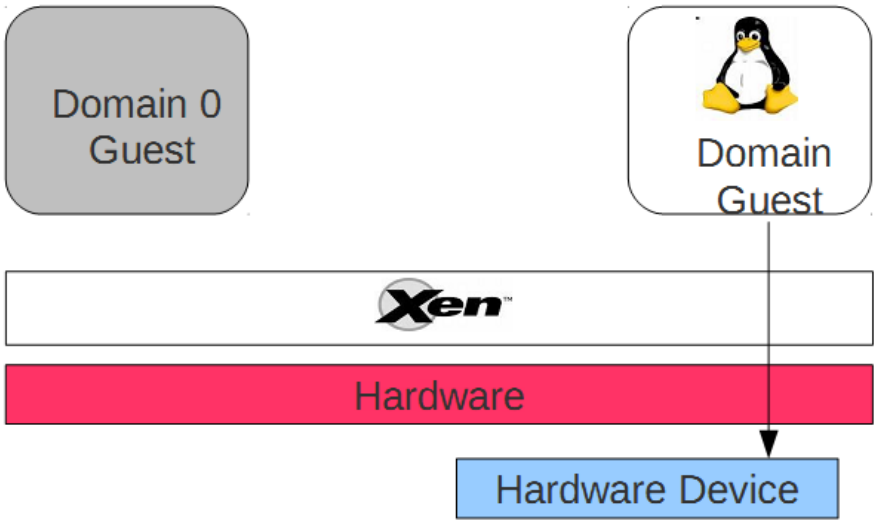


Fig. 42 – *PCI Passthrough*

El huésped *Domain U* tiene ahora privilegios para hablar directamente con un dispositivo hardware específico en vez de utilizar el canal de eventos y los *Backend drivers* para comunicarse con él ya que la comunicación con los dispositivos PCI se hacía de forma similar a la ilustrada en el ejemplo del apartado 9.2.1 Comunicación entre *Domain 0* y *Domain U*.

9.3.*2 VGA Passthrough*

PCI Passthrough se utiliza principalmente para dispositivos como interfaces de red y puertos USB, sin embargo, no todos los dispositivos que utilizan el bus PCI son compatibles con esta tecnología. Una de las excepciones en este tipo de dispositivos son las controladoras gráficas. Las controladoras gráficas tienen una forma especial de funcionar muy consolidada. Por ejemplo, no toda la memoria está alojada de forma dinámica sino que la tarjeta contiene su propia BIOS que necesita ser re-ejecutada.

Aunque es difícil de implementar, *VGA-Passthrough* es una de las características más importantes y novedosas que puede ofrecer la plataforma de virtualización Xen y puede ser muy útil en algunos casos.

Imaginemos a un usuario trabajando en un dominio sin privilegios, posiblemente sin ser consciente de que está utilizando un entorno virtualizado, *con VGA-Passthrough*. Este usuario podría ser capaz de hacer cualquier cosa que el pudiese hacer en un ordenador normal incluyendo el uso de todas las capacidades de su tarjeta gráfica. Por debajo el *Domain 0* tendría el control del sistema operativo virtualizado siendo capaz de protegerlo de las acciones maliciosas.

Gracias a esta característica será posible realizar la plataforma de descifrado comentada en los objetivos del proyecto obteniendo al mismo tiempo todas las ventajas que proporcionan las herramientas comerciales criptográficas, el sistema operativo Linux y la mejora de rendimiento que ofrecen las tarjetas gráficas modernas.

9.3.3 Distribución de la memoria y *passthrough*

Para entender lo que ocurre cuando se utiliza la característica *PCI passthrough* necesitamos comprender como la memoria de la máquina se construye cuando se ejecuta Xen. La figura 43 muestra un bosquejo de cómo se organiza la memoria en un equipo que se ejecuta con Xen. Además de la BIOS, hay un rango reservado para la comunicación con los dispositivos. Este área es utilizada entre otros por las interfaces de red o los controladores de disco.

Después de estas direcciones de memoria, se encuentra Xen y las máquinas virtualizadas. Cada máquina virtual muestra un diseño exactamente igual al que mostraría un equipo normal. La VM sólo conoce que su memoria comienza en la dirección cero; son Xen y la CPU los que traducen estas direcciones a sus correspondientes posiciones en la memoria física. Del mismo modo, la VM posee su propia BIOS que, debido a su naturaleza estática, es tan sólo una copia exacta de la BIOS original.

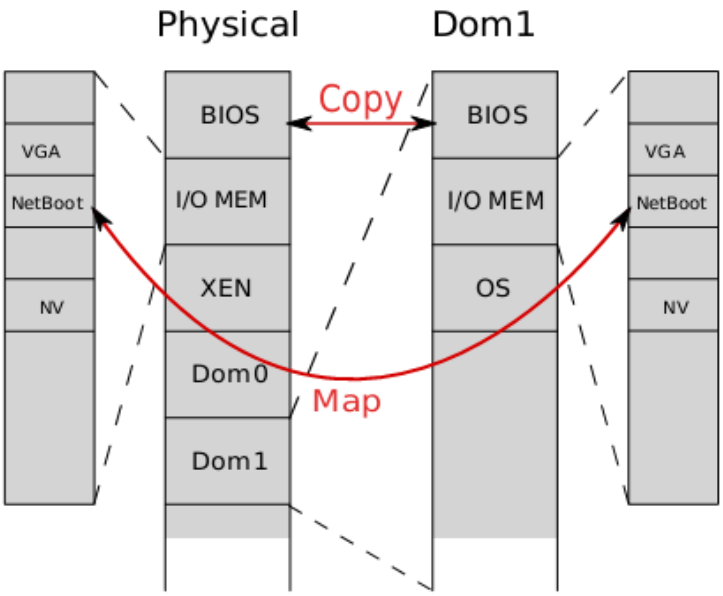


Fig. 43 – Visión esquemática de un posible uso de la memoria durante el funcionamiento de Xen

Cuando se aplica *PCI passthrough*, ciertas áreas de la memoria física de la máquina son asignadas a la máquina virtual. Esto puede verse en la figura anterior con el dispositivo *NetBoot*. El área de memoria de la VM ocupada por *NetBoot* está realmente en una posición fuera de su rango de memoria. Cuando un sistema operativo huésped escribe en una de esas direcciones de memoria, Xen se asegura de que se está escribiendo en las direcciones apropiadas. Esto implica que ninguna otra VM puede utilizar ese mismo dispositivo**.**

9.4 Antecedentes de *VGA passthrough*

9.4.1 Parche para tarjetas *Intel Graphics*

Aunque *VGA passthrough* está lejos de estar disponible para todos los usuarios, sí que se ha hecho un intento de lograr acceso directo a la VGA nativa en un dominio sin privilegios (máquina virtual). En mayo del 2008, Jean Guyader[38] presentó un parche para Xen permitiendo el uso de *passthrough*. A pesar de que el código no es muy genérico puede servir bien como prueba de concepto. En esta sección se analizan los pasos que este tipo de parches necesita y el por qué y el cómo de cada una de las partes.

[38] El parche puede encontrarse en la siguiente url
http://article.gmane.org/gmane.comp.emulators.xen.devel/51194

Hay cuatro pasos importantes que se deben realizar para lograr *VGA passthrough*:

1. Mapear el *buffer* de memoria VGA al huésped.
2. Copiar la VGA BIOS al huésped.
3. Mapear los puertos VGA I/O.
4. Deshabilitar el código VGA de Xen.

9.4.1.1 Mapear VGA *framebuffer* al huésped

Un adaptador VGA estándar puede tener una memoria de hasta 256 KiB, de esta memoria unos 128 KiB son usados en los modos normal VGA/EGA, monocromático y pantalla CGA. Esta memoria se llama "framebuffer" y se asigna normalmente de la dirección `0xA0000` hasta la `0xC0000` de la memoria principal de las máquinas. Este rango de direcciones deben ser asignadas a la memoria de la VM para que el sistema operativo pueda direccionar y acceder al adaptador de video.

Los PCs tienen un mecanismo para informar sobre qué posiciones de memoria son usadas por la BIOS y cuáles están libres para que las pueda utilizar el sistema operativo. Este mecanismo se conoce con el nombre de BIOS-e820[39] Las direcciones de memoria VGA y de la BIOS VGA no se obtienen mediante este mecanismo sino que están fijadas en una posición de memoria. Cada sistema operativo conocen esas direcciones.

Esto tiene dos implicaciones. La primera es que la memoria VGA no puede ubicarse dentro de la memoria de la máquina virtual. Cuando el sistema operativo de la VM intenta leer o escribir en ese rango específico de memoria, las direcciones lógicas no deben ser traducidas por Xen a direcciones físicas dentro del espacio de memoria de la VM.

En segundo lugar, como el adaptador VGA no especifica su rango de direcciones de forma dinámica, Xen no puede utilizar su mecanismo *passthrough* de forma normal. Las direcciones VGA deben estar siempre codificadas o grabadas en el código**.**

La mayor parte de estas tareas está contenida en los archivos *tools/libxc/xc_hvm_build.c* y *tools/libxc_linux.c* del directorio de Xen. Son precisamente estos ficheros los que se modifican en el parche de Jean Guyader.

Tras estos primeros pasos, la BIOS VGA se debe copiar a la memoria del huésped (siguiente apartado) y se le debe asignar la memoria VGA.

[39] En los sistemas basados en la arquitectura x86 el valor hexadecimal e820 se debe colocar en el registro AX para conocer los rangos de memoria disponibles y los reservados por la BIOS.

9.4.1.2 Copiar la BIOS VGA al huésped

Al igual que la memoria *framebuffer*, la BIOS VGA se encuentra en un rango fijo de memoria `0xC0000-0xCAFFF`. La BIOS proporciona un conjunto de funciones para controlar el adaptador de video por lo que debe copiarse al espacio de memoria del huésped y ejecutarse.

Este parche introduce en *tools/firware/hvmloader.c* una comprobación del fabricante y el ID del dispositivo de la tarjeta gráfica para decidir qué tipo de adaptador se está utilizando y así poder copiar el rango de memoria de la BIOS en cuestión. Para asegurar que la BIOS tiene espacio suficiente se han desplazado 8KiB las posiciones de memoria de "Etherboot". Además, dado que Xen utiliza los rangos de memoria de la BIOS VGA como espacio de almacenamiento temporal para peticiones relacionadas con la BIOS, es necesario reubicar esta zona de memoria. Estos cambios se añaden en */tools/firmaware/config.h*.

Finalmente, se necesita que las instrucciones de la BIOS sean ejecutadas para inicializar el adaptador gráfico, esto se hace en el archivo */tools/ioemu/hw/pc.c*.

9.4.1.3 Mapear los puertos VGA I/O

Los puertos de entrada/salida de la tarjeta gráfica sirven para mejorar la comunicación entre los programas del ordenador y el adaptador VGA. Para comunicar un mensaje al adaptador gráfico, es suficiente con escribir un valor en una dirección concreta de memoria. El adaptador leerá esa dirección del bus comportándose en consecuencia.

Como el hardware de gráficos está *escuchando* en una dirección específica de memoria, Xen debe facilitar el mapeo de las direcciones virtuales a este rango de memoria física. En el parche se modifica el archivo *tools/libxc/xc_hvm_build.c* para mapear el rango de memoria `0x3C0-0x3E0` al sistema operativo del huésped.

9.4.1.4 Desabilitar el código VGA de Xen

Para utilizar VGA *passthrough* se requiere que Xen evite la captura y utilización del adaptador VGA por parte del *Domain 0*.

Para realizar esta tarea habría que modificar el archivo hvm.c. No obstante este parche no realiza dichos cambios por lo que es tarea del usuario evitar que el *Domain 0* capture el adaptador de video.

9.4.1.5 Limitaciones

El parche resulta ser bastante rígido debido sobre todo a que todas las posiciones de memoria están escritas en el código del programa lo que dificulta su adaptación. Tras haber sido probado por la comunidad de Xen, parece que el parche sólo funciona correctamente en sistemas con chipset de Intel y con tarjeta gráfica integrada Intel.

El principal problema que impide que el parche funcione en otros entornos es que parece que se basa en que la BIOS de la tarjeta gráfica se re-ejecute. Este mecanismo no parece funcionar de manera fiable con ninguna otra combinación de marcas de hardware.

Otro de los problemas es que el parche solo funciona para la tarjeta gráfica principal, por lo que no es posible utilizar múltiples tarjetas con *VGA-passthrough*.

9.4.2 Parches para tarjetas con problemas de virtualización

Pese a las limitaciones de la familia de parches anteriores, la iniciativa sirvió de guía a otros desarrolladores. En Agosto de 2008 el desarrollador de Intel Han Weidong publicaba unos parches especializados para una gran cantidad de tarjetas que requerían la extracción previa de la BIOS gráfica de forma manual por parte del usuario. Estos parches permitían utilizar una amplia gama de tarjetas gráficas de virtualización difícil y para las que deben solucionarse varios problemas como pueden ser la capacidad de FLR[40], la necesidad de asignar registros pBAR=vBAR, problemas de re-ejecución [41]de la BIOS, etc.

Para entender la expresión "pBAR=vBAR" es necesario saber cómo se direcciona un dispositivo PCI. Para direccionar un dispositivo PCI este debe asignarse dentro del espacio de memoria de los puertos de entrada/salida o en el espacio de memoria asignado al sistema operativo. El controlador del dispositivo o el sistema operativo programarán los registros base (*Base Address Registers* o BARs) para informar al dispositivo PCI de las direcciones de memoria asignadas en la máquina real. La asignación pBAR=vBAR se refiere a que las direcciones virtuales del dispositivo (las que conocerá el huésped) deben ser exactamente las mismas que las reales y además el hypervisor de Xen no debe realizar traducción alguna en este caso.

Estos parches solucionan el problema de asignación de registros pBAR=vBAR de una manera poco elegante. Es por ello por lo que todavía no han sido incluidos en el código oficial de Xen, a la espera de obtener una solución más flexible y cómoda para este problema. En estos parches se incluye además soporte para cargar la BIOS de la tarjeta gráfica extraída previamente por el usuario lo que soluciona en gran medida el problema de re-ejecución de BIOS en el huésped.

Para utilizar estos parches, realizados en 2008, se necesitaba la versión 3.5 de Xen. En el momento de realización de este trabajo fin de carrera, en 2011, Xen se encontraba en su versión 4.2-unstable, por lo que hay muchas características nuevas en Xen, desarrolladas desde entonces, que no pueden utilizarse si se desean utilizar estos parches. Uno de los aspectos más importantes a tener en cuenta es que se han realizado bastantes mejoras

[40] **FLR capability:** Function Level Reset Support (FLR). FLR puede ser utilizado a la vez que la tecnología de virtualización Intel permitiendo a un sistema virtualizado tener control total sobre un dispositivo, incluyendo su inicialización sin interferir con el resto de la plataforma.

[41] **Re-execution BIOS:** Cuando se utiliza VGA-passthrough se necesitan emular puertos de E/S, varios rangos de memoria heredados de la arquitectura x86 (legacy ranges) y en algunos adaptadores de video es necesario re-ejecutar la BIOS gráfica en el huésped. Algunas tarjetas sin embargo, presentan el problema de truncar bits de la BIOS o modificarlos tras ser inicializada en el sistema host. En palabras de los desarrolladores de Xen: "vBIOS bits may be truncated or modified after initialization in host, thus it might result in re-execution issue"

que facilitan el uso de la característica VGA-passthrough o que implementan mejoras de compatibilidad con las tecnologías de virtualización directa (VT-d).

Para poder disfrutar de todo el potencial de Xen además de virtualizar la tarjeta gráfica nativa es necesario actualizar estos parches a la versión 4.2. En el siguiente apartado se muestra paso a paso como se han modificado estos parches para hacer que encajen en el nuevo código.

10

Implementación de una Solución de *VGA-Passthrough* en Xen

En este apartado se presentan las modificaciones necesarias que deben realizarse sobre el código actual de Xen para incluir la misma funcionalidad que añadían los parches publicados en 2008. El producto final serán unos nuevos parches funcionales que encajen en el código de la actual versión de Xen.

Para adaptar unos parches antiguos al código actual de un programa se necesita por un lado el código de los parches y por otro lado el código actual del *software* en el que se deben incluir las modificaciones que los parches realizaban. En el caso de Xen, la versión actual del *software* es la versión Xen-4.2-unstable disponible a través de su repositorio de desarrollo. Al ser una rama de desarrollo, el código está en constante cambio registrándose cada modificación de código mediante un número. Concretamente, el código de Xen utilizado es la revisión número 23350 y se puede obtener de la siguiente forma (usando mercurial para acceder al repositorio de código):

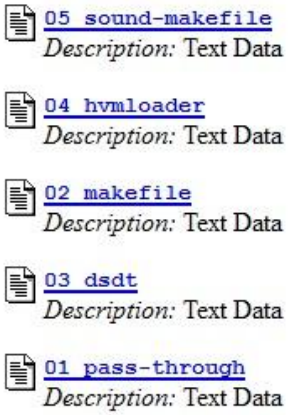
```
hg clone -r 23350 http://xenbits.xensource.com/hg/staging/xen-unstable.hg/ Xen-23350
```

De este modo, obtendremos el código fuente de la revisión 23350 de Xen en un directorio llamado Xen-23350. Los parches originales pueden encontrarse en diferentes enlaces:

- Los parches originales creados por Han Weidong, agrupados según la modificación que realizan, pueden encontrarse al final de la siguiente página[42]:

[42] http://old-list-archives.xen.org/archives/html/xen-devel/2009-08/msg01176.html

- Parches agrupados según el nombre del archivo del código fuente de Xen que se modifica[43]:



Los parches mostrados en las dos imágenes de arriba incluyen las mismas modificaciones. La diferencia entre unos y otros está en la forma en la que se ha distribuido el código en los diferentes archivos. Podemos obtener información valiosa de ambos enlaces. Del primer enlace, observando los nombres de los archivos adjuntos, podemos conocer la función que realiza el código de cada parche, lo que es útil si se pretende añadir alguna funcionalidad adicional al código o si se quieren conocer los detalles de cómo se realizan las modificaciones. El segundo enlace ofrece los mismos parches mezclando las funciones que se realizan pero recogiendo todas las modificaciones que se deben llevar a cabo sobre un mismo archivo del código fuente en un solo archivo de parche. Esto es bastante cómodo a la hora de añadir todas las modificaciones necesarias al código fuente actual de Xen.

[43] http://old-list-archives.xen.org/archives/html/xen-devel/2010-05/msg00441.html

En los siguientes apartados se muestra cómo se han ido generando los nuevos parches que permiten la utilización de *VGA-passthrough* en el entorno de virtualización Xen versión 4.2.

10.1 Passthrough.patch

El parche "Qemu-change-for-vBAR-pBAR" modificaba el archivo *pass-through.c* del código de Xen para solucionar una parte del problema de mapeo de direcciones comentado en el apartado sobre tarjetas con problemas de virtualización. El primer paso para adaptar este parche es encontrar la ruta del archivo a modificar.

Observando las primeras líneas del parche 'Qemu-change-for-vBAR-pBAR.patch' podemos una obtener una parte de la ruta:

```
11 diff --git a/hw/pass-through.c b/hw/pass-through.c
12 index 4a9e03a..dfd592b 100644
13 --- a/hw/pass-through.c
14 +++ b/hw/pass-through.c
```

Fig. 44 - Primeras líneas del parche Qemu-change

No obstante, queda mucho más claro si observamos la ruta en el parche correspondiente del segundo enlace 'pass-thhrough.patch':

```
1 --- tools/ioemu-remote/hw/pass-through.c.org    2010-05-10 15:24:52.115083489 +0200
2 +++ tools/ioemu-remote/hw/pass-through.c        2010-05-10 16:02:19.517997970 +0200
3 @@ -1865,6 +1865,75 @@
```

Fig. 45 - Primeras líneas del parche pass-through

La ruta es, por tanto, *Xen-23350/tools/ioemu-remote/hw/pass-through.c*. Sin embargo, esta ruta no existe en el código de Xen que se ha descargado anteriormente. Es necesario realizar una primera compilación para que se generen todos los directorios y archivos de código fuente que faltan. Para que se genere el directorio 'ioemu-remote' es necesario hacer una compilación previa del código, que a su vez precisa de una serie de librerías y herramientas. A continuación se muestran todos los pasos necesarios para compilar Xen.

En primer lugar, obtenemos el código de Xen del repositorio.

```
hg clone -r 23350 http://xenbits.xensource.com/hg/staging/xen-unstable.hg/ Xen-23350
```

A continuación, instalamos un conjunto de paquetes necesarios para poder compilar el código obtenido:

```
Xen-23350/# apt-get install bcc bin86 binutils bison bridge-utils build-essential bzip2 debhelper debootstrap dpkg-dev e2fslibs-dev flex gawk gcc gettext git-core gitk iasl iproute

Xen-23350/# apt-get install libbz2-dev libc6-dev libcurl3 libcurl4-openssl-dev libjpeg62-dev libncurses5-dev libsdl1.2debian-all libsdl1.2-dev libsdl-dev libssl-dev libvncserver0 libvncserver-dev libx11-dev libx86-dev

Xen-23350/# apt-get install make mercurial module-init-tools ocaml patch pciutils-dev python python-dev python-pam python-twisted
```

```
screen ssh texinfo texlive-fonts-extra texlive-fonts-recommended
texlive-latex-base texlive-latex-recommended tgif transfig udev
uuid-dev x-dev xorg zlib1g-dev
```

Finalmente, nos movemos al directorio “tools” y realizamos la compilación:

```
Xen-23350/# cd tools
Xen-23350/# make
Xen-23350/# make clean
```

Tras realizar estas operaciones aparecen varios directorios con el código fuente que se necesita parchear.

Para este parche en concreto, el proceso a seguir es simple ya que ninguna línea de código de este parche ha sido incluida en la versión actual de Xen por lo que tan sólo es necesario buscar el lugar donde se deben incluir las modificaciones y añadirlas manualmente. Analizando el parche podemos identificar las dos modificaciones que se realizaban:

```
--- a/hw/pass-through.c
+++ b/hw/pass-through.c
@@ -3174,6 +3174,7 @@ static int pt_cmd_reg_read(struct pt_dev
*ptdev,
 }

 /* read BAR */
+static int gfx_first_read_BAR[7] = {1, 1, 1, 1, 1, 1, 1};
 static int pt_bar_reg_read(struct pt_dev *ptdev,
         struct pt_reg_tbl *cfg_entry,
         uint32_t *value, uint32_t valid_mask)
@@ -3196,6 +3197,16 @@ static int pt_bar_reg_read(struct pt_dev
*ptdev,
     /* use fixed-up value from kernel sysfs */
     *value = ptdev->pci_dev->base_addr[index];

+    if ( ptdev->pci_dev->device_class == 0x300 )
+    {
+        if ( gfx_first_read_BAR[index] == 1 )
+        {
+            gfx_first_read_BAR[index] = 0;
+            PT_LOG("first read BARs of gfx\n");
+            return 0;
+        }
+    }
+
     /* set emulate mask depend on BAR flag */
     switch (ptdev->bases[index].bar_flag)
     {
--
1.6.0.4
```

Fig. 46 - Antiguo parche *passthrough*

Las líneas que comienzan con “@@” indican la posición donde se realizarán las modificaciones del código, las líneas que comienzan con el signo “+” son líneas que se añadirán tras aplicar el parche, las línea que comienzan con el signo “-“ son líneas que se eliminarán y las líneas de código sin marcar son líneas presentes en el código (antes de la aplicación del parche) que sirven de guía para saber donde introducir las modificaciones.

Para la primera modificación se debe buscar en el código actual la cadena que le precede, /*read BAR*/ y añadir la modificación correspondiente:

```
3245 /* read BAR */
3246 static int pt_bar_reg_read(struct pt_dev *ptdev,
3247         struct pt_reg_tbl *cfg_entry,
3248         uint32_t *value, uint32_t valid_mask)
3249 {
3250     struct pt_reg_info_tbl *reg = cfg_entry->reg;
3251     uint32_t valid_emu_mask = 0;
3252     uint32_t bar_emu_mask = 0;
3253     int index;
```

Fig. 47 - Código *passthrough* antes de ser modificado

```
3245 /* read BAR */
3246 /*PARCHE1*/
3247 static int gfx_first_read_BAR[7] = {1, 1, 1, 1, 1, 1, 1};
3248 /*PARCHE_END*/
3249 static int pt_bar_reg_read(struct pt_dev *ptdev,
3250         struct pt_reg_tbl *cfg_entry,
3251         uint32_t *value, uint32_t valid_mask)
3252 {
```

Fig. 48 - Código *passthrough* después de ser modificado

Para la segunda parte de código se procede de la misma forma:

Antes:

```
3268     /* use fixed-up value from kernel sysfs */
3269     *value = ptdev->pci_dev->base_addr[index];
3270
3271     /* set emulate mask depend on BAR flag */
3272     switch (ptdev->bases[index].bar_flag)
3273     {
3274     case PT_BAR_FLAG_MEM:
3275         bar_emu_mask = PT_BAR_MEM_EMU_MASK;
3276         break;
```

Fig. 49 - Segunda sección del código *passthrough.c* a modificar

Después:

```
3268     /* use fixed-up value from kernel sysfs */
3269     *value = ptdev->pci_dev->base_addr[index];
3270 /*PARCHE2*/
3271     if ( ptdev->pci_dev->device_class == 0x300 )
3272     {
3273         if ( gfx_first_read_BAR[index] == 1 )
3274         {
3275             gfx_first_read_BAR[index] = 0;
3276             PT_LOG("first read BARs of gfx\n");
3277             return 0;
3278         }
3279     }
3280 /*PARCHE_END*/
3281     /* set emulate mask depend on BAR flag */
3282     switch (ptdev->bases[index].bar_flag)
3283     {
```

Fig. 50 - Segunda sección del código *passthrough.c* ya modificada

Si se observa el parche original se puede ver como las modificaciones realizadas son exactamente las mismas, lo único que ha cambiado es el lugar donde estas nuevas líneas

han sido introducidas. Esto es debido a que el código del archivo pass-through.c no ha variado demasiado a lo largo del tiempo, lo que facilita nuestra tarea.

10.1.1 Creación del nuevo parche

Se debe renombrar el archivo original del código fuente, por ejemplo:

```
Xen-23350/tools/ioemu-remote/hw# mv pass-through.c pass-
through.c.orig
```

Ahora se guarda el archivo modificado (según se explicó arriba) con el nombre original del archivo, es decir, *pass-through.c*. Una vez hecho esto se crea el parche con la utilidad *diff*:

```
Xen-23350/# diff -Naur tools/ioemu-remote/hw/pass-through.c.orig
tools/ioemu-remote/hw/pass-through.c > pass-through.patch
```

El parche final queda como sigue:

```
--- tools/ioemu-remote/hw/pass-through.orig     2011-05-16 17:20:16.789438520 +0200
+++ tools/ioemu-remote/hw/pass-through.c        2011-05-16 17:33:09.073700625 +0200
@@ -3243,6 +3243,9 @@
 }

 /* read BAR */
+/*PARCHE1*/
+static int gfx_first_read_BAR[7] = {1, 1, 1, 1, 1, 1, 1};
+/*PARCHE_END*/
 static int pt_bar_reg_read(struct pt_dev *ptdev,
         struct pt_reg_tbl *cfg_entry,
         uint32_t *value, uint32_t valid_mask)
@@ -3265,6 +3268,18 @@
     /* use fixed-up value from kernel sysfs */
     *value = ptdev->pci_dev->base_addr[index];

+/*PARCHE2*/
+    if ( ptdev->pci_dev->device_class == 0x300 )
+    {
+        if ( gfx_first_read_BAR[index] == 1 )
+        {
+            gfx_first_read_BAR[index] = 0;
+            PT_LOG("first read BARs of gfx\n");
+            return 0;
+        }
+    }
+/*PARCHE_END*/
+
     /* set emulate mask depend on BAR flag */
     switch (ptdev->bases[index].bar_flag)
     {
```

Fig. 51 - Parche *passthrough.patch*

10.2 Makefile.patch

El parche “Xen-load-vbios-file” realizaba modificaciones en dos archivos del código fuente, *tools/firmware/hvmloader/Makefile* y *tools/firmware/hvmloader/hvmloader.c* Este es el único de todos los parches que realiza modificaciones en el archivo *Makefile* de la carpeta *hvmloader* por lo que a partir de este se pueden crear el parche *Makefile.patch* completo y parte del parche *hvmloader.patch* explicado más adelante.

En este caso, la actualización del parche no es tan sencilla como en el caso anterior ya que la sintaxis utilizada en el código actual del archivo *Makefile* ha cambiado con respecto al anterior:

```
1  diff -r 4f13590588e3 tools/firmware/hvmloader/Makefile
2  --- a/tools/firmware/hvmloader/Makefile Fri Aug 28 14:41:36 2009 +0800
3  +++ b/tools/firmware/hvmloader/Makefile Mon Aug 31 12:39:45 2009 +0800
4  @@ -50,6 +50,7 @@ roms.h: ../rombios/BIOS-bochs-latest ../
5   roms.h: ../rombios/BIOS-bochs-latest ../vgabios/VGABIOS-lgpl-latest.bin \
6           ../vgabios/VGABIOS-lgpl-latest.cirrus.bin ../etherboot/eb-roms.h
7           sh ./mkhex rombios ../rombios/BIOS-bochs-latest > roms.h
8  +        sh ./mkhex vgabios_pt ../vgabios/vgabios-pt.bin >> roms.h
9           sh ./mkhex vgabios_stdvga ../vgabios/VGABIOS-lgpl-latest.bin >> roms.h
10          sh ./mkhex vgabios_cirrusvga \
11                  ../vgabios/VGABIOS-lgpl-latest.cirrus.bin >> roms.h
```

Fig. 52 - Parche *Xen-load-vbios-file*, parte que modifica el archivo *makefile*

Para adaptar estas modificaciones al *Makefile* actual se deben buscar las líneas de código de la imagen anterior, por ejemplo una parte de la línea 7 “`sh ./mkhex rombios`”:

```
72 roms.inc: $(ROMBIOS_ROM) $(SEABIOS_ROM) $(STDVGA_ROM) $(CIRRUSVGA_ROM) ../etherboot/eb-roms.h
73         echo "/* Autogenerated file. DO NOT EDIT */" > $@.new
74
75 ifneq ($(ROMBIOS_ROM),)
76         echo "#ifdef ROM_INCLUDE_ROMBIOS" >> $@.new
77         sh ./mkhex rombios $(ROMBIOS_ROM) >> $@.new
78         echo "#endif" >> $@.new
79 endif
80
81 ifneq ($(SEABIOS_ROM),)
82         echo "#ifdef ROM_INCLUDE_SEABIOS" >> $@.new
83         sh ./mkhex seabios $(SEABIOS_ROM) >> $@.new
84         echo "#endif" >> $@.new
85 endif
```

Fig. 53 - Código actual del archivo *makefile*

Como se puede observar en la imagen, la sintaxis ha cambiado bastante por lo que el código del parche debe analizarse con detenimiento para que la nueva adaptación termine realizando la misma función que el original.

El *Makefile* original creaba un archivo llamado *roms.h* en el que se incluían todos los firmwares o BIOS necesarios para el funcionamiento correcto de Xen. Entre otros se pueden observar *rombios*, *vgabios_stdvga*, *vgabios_cirrusvga* etc. En el código actual de Xen, no se crea ningún archivo llamado *roms.h* sino que en su lugar se crea el archivo *roms.inc*.

Utilizando la sintaxis del *Makefile* actual se pueden realizar las mismas modificaciones que se realizaban con el antiguo parche. Estas operaciones sirven para copiar el contenido del *firmware vgabios_pt* en el archivo *roms.inc* de Xen. Este *firmware* es el de la BIOS de la

tarjeta que se pretende virtualizar utilizando *passthrough*. Más adelante se explicará cómo extraer la BIOS de la tarjeta gráfica para utilizarla en Xen.

El primer paso para adaptar el parche es definir la referencia a la nueva BIOS:

```
53 STDVGA_ROM    := ../vgabios/VGABIOS-lgpl-latest.bin
54 ifeq ($(CIRRUSVGA_DEBUG),y)
55 CIRRUSVGA_ROM := ../vgabios/VGABIOS-lgpl-latest.cirrus.debug.bin
56 else
57 CIRRUSVGA_ROM := ../vgabios/VGABIOS-lgpl-latest.cirrus.bin
58 endif
59 #PARCHE#
60 PTVGA_ROM := ../vgabios/vgabios-pt.bin
61 #PARCHE END1#
```

Fig. 54 - Primera modificación del archivo *makefile*

El segundo paso es añadir el nuevo *firmware* en la definición de *roms.inc*:

```
75 #PARCHE2#
76 roms.inc: $(ROMBIOS_ROM) $(SEABIOS_ROM) $(PTVGA_ROM) $(STDVGA_ROM) $(CIRRUSVGA_ROM) ../etherboot/eb-roms.h
77        echo "/* Autogenerated file. DO NOT EDIT */" > $@.new
78 #
79 #roms.inc: $(ROMBIOS_ROM) $(SEABIOS_ROM) $(STDVGA_ROM) $(CIRRUSVGA_ROM) ../etherboot/eb-roms.h
80 #       echo "/* Autogenerated file. DO NOT EDIT */" > $@.new
81 #PARCHE2END#
```

Fig. 55 - Segunda modificación del archivo *makefile*

Finalmente, se añade la BIOS al archivo *roms.inc* respetando la sintaxis actual del código (ver figura 53):

```
 95 #PARCHE3#
 96 ifneq ($(PTVGA_ROM),)
 97         echo "#ifdef ROM_INCLUDE_VGABIOS" >> $@.new
 98         sh ./mkhex vgabios_pt $(PTVGA_ROM) >> $@.new
 99         echo "#endif" >> $@.new
100 endif
101 #PARCHE3END#
```

Fig. 56 - Tercera y última modificación del archivo *makefile*

Una vez realizadas las modificaciones se puede generar el parche, tal y como se explica en el caso anterior:

```
Xen-23350/#   diff   -Naur   tools/firmware/hvmloader/Makefile.orig
tools/firmware/hvmloader/Makefile > Makefile.patch
```

10.3 Rombios.patch

Los antiguos parches “Xen-load-vbios-file” y “Xen-vBAR-pBAR” entre otras modificaciones realizaban cambios de código en el archivo *hvmloader.c* del directorio *tools/firmware/* de Xen, en este apartado se explica cómo se han llevado a cabo estas mismas modificaciones en el código actual de Xen generando así tres archivos de parche diferentes: *Config.patch*, *hvmloader.patch* y *rombios.patch*.

En este caso se observa un cambio importante entre las versiones 3.5 y 4.2 de Xen dado que el archivo *hvmloader.c* ha sido dividido en la versión 4.2 en dos partes: *hvmloader.c* y *rombios.c*. Debido a esto se deberá tener especial cuidado al incorporar las modificaciones del parche antiguo al código actual.

Otra de las cuestiones a tener en cuenta es que partes del código del parche antiguo parecen haber sido incluidas en el archivo actual *rombios.c*:

```
177         /* Map the I/O memory and port resources. */
178         for ( bar = 0; bar < 7; bar++ )
179         {
180             bar_reg = PCI_BASE_ADDRESS_0 + 4*bar;
181             if ( bar == 6 )
182                 bar_reg = PCI_ROM_ADDRESS;
183
184             bar_data = pci_readl(devfn, bar_reg);
185             pci_writel(devfn, bar_reg, ~0);
186             bar_sz = pci_readl(devfn, bar_reg);
187             pci_writel(devfn, bar_reg, bar_data);
188             if ( bar_sz == 0 )
189                 continue;
```

Fig. 57 - Captura del código actual del archivo rombios.c

```
43              virtual_vga = VGA_pt;
44              gfx_bdf = devfn;
45 +
46 +            /* Make vBAR=pBAR */
47 +            printf("Make vBAR = pBAR of assigned gfx\n");
48 +            for ( bar = 0; bar < 7; bar++ )
49 +            {
50 +                bar_reg = PCI_BASE_ADDRESS_0 + 4*bar;
51 +                if ( bar == 6 )
52 +                    bar_reg = PCI_ROM_ADDRESS;
53 +                /* When first time read, it will return physical address */
54 +                bar_data = pci_readl(devfn, bar_reg);
55 +                pci_writel(devfn, bar_reg, bar_data);
56 +
57 +                /* Now enable the memory or I/O mapping. */
```

Fig. 58 - Captura del antiguo parche Xen-vBAR-pBAR

Como se puede observar, el código es bastante similar exceptuando las líneas 43 y 44 que en el código actual no existen y son las que marcan la diferencia a la hora de realizar la asignación de registros vBAR=pBAR entre las tarjetas de virtualización sencillas y las tarjetas de virtualización compleja.

El código actual realiza la asignación de registros base vBAR=pBAR para adaptadores de virtualización sencilla. Para estos adaptadores, es posible leer la información necesaria de su *firmware* sin necesidad de herramientas adicionales. Esto hace posible que tarjetas de alta gama como las tarjetas gráficas Quadro FX 3800, 4800, 5800… de nVIDIA puedan ser utilizadas sin ningún parche adicional. Cabe preguntarse, por tanto, por qué no se decidió

emplear una tarjeta de esa gama para este proyecto, teniendo en cuenta que su empleo evitaría todo el tedioso proceso de adaptación de Xen. El alto precio de estas tarjetas (del orden de los 1500€ a 3000€) y su menor eficiencia en el descifrado de archivos desaconsejó su uso en este proyecto.

El parche antiguo Xen-vBAR-pBAR añadía esta misma modificación pero para tarjetas gráficas de las que se hubiese extraído el firmware previamente y se hubiese guardado en la carpeta hvmloader con el nombre *vgabios-pt.bin*. Por lo tanto aunque en el *rombios.c* actual parece que existe el mismo código que en el parche, este no es exactamente el mismo sino que difiere ligeramente en el caso de haber extraído la BIOS gráfica. Dado que no se pretende eliminar la actual facilidad de uso de las tarjetas de alta gama para realizar passthrough, se decidió duplicar el código para añadir soporte al caso de extracción de BIOS.

En las capturas siguientes se puede ver como los parches antiguos se aplicaban siguiendo un orden, primero el parche Xen-load-vbios-file y luego el parche Xen-vBAR-pBAR. Es por ello que en la figura 58 las líneas 43 "`virtual_vga = VGA_pt;`" y 44 "`gfx_bdf = devfn;`" no tienen el símbolo '+' al comienzo pues son líneas que deben existir, sin embargo, esto no significa que existieran en el código oficial de Xen 3.5 sino que habían sido añadidas con anterioridad al utilizar el parche Xen-Load-vBios-file (figura 59).

```
25 @@ -215,7 +218,10 @@ static void pci_setup(void)
26              else if ( (vendor_id == 0x1013) && (device_id == 0xb8) )
27                  virtual_vga = VGA_cirrus;
28              else
29 +            {
30                  virtual_vga = VGA_pt;
31 +                gfx_bdf = devfn;
32 +            }
33              break;
```

Fig. 59 - Extracto del parche *Xen-Load-vBios-File*

```
41 @@ -221,6 +221,40 @@ static void pci_setup(void)
42              {
43                  virtual_vga = VGA_pt;
44                  gfx_bdf = devfn;
45 +
46 +                /* Make vBAR=pBAR */
47 +                printf("Make vBAR = pBAR of assigned gfx\n");
48 +                for ( bar = 0; bar < 7; bar++ )
49 +                {
50 +                    bar_reg = PCI_BASE_ADDRESS_0 + 4*bar;
51 +                    ⋮
```

Fig. 60 - Extracto del parche *Xen-vBAR-pBAR*

Viendo el contenido de estos dos parches, se puede intuir como realizar la modificación en el código actual de *rombios.c*:

```
147         case 0x0300:
148             /* If emulated VGA is found, preserve it as primary VGA. */
149             if ( (vendor_id == 0x1234) && (device_id == 0x1111) )
150                 virtual_vga = VGA_std;
151             else if ( (vendor_id == 0x1013) && (device_id == 0xb8) )
152                 virtual_vga = VGA_cirrus;
153             else if ( virtual_vga == VGA_none )
154                 virtual_vga = VGA_pt;
155             break;
156                                 ⋮
176
177         /* Map the I/O memory and port resources. */
178         for ( bar = 0; bar < 7; bar++ )
                                 ⋮
```

Fig. 61 - Captura del código actual *rombios.c*

```
153             else if ( virtual_vga == VGA_none )
154                 //# virtual_vga = VGA_pt;
155 //#PARCHE1#
156             {
157                 virtual_vga = VGA_pt;
158                 gfx_bdf = devfn;                    Asignación vBAR=pBAR para el caso de
159                                                     BIOS extraída manualmente.
160                 /* Make vBAR=pBAR */
161                 printf("Make vBAR = pBAR of assigned gfx\n");
162                 for ( bar = 0; bar < 7; bar++ )
163                 {
164                                 ⋮
187                     /* This is the barber's pole mapping used by Xen. */
188                     link = ((pin - 1) + (devfn >> 3)) & 3;
189                     isa_irq = pci_readb(PCI_ISA_DEVFN, 0x60 + link);
190                     pci_writeb(devfn, PCI_INTERRUPT_LINE, isa_irq);
191                 }
192                 continue;
193         }
194 //#PARCHE1END#
                                 ⋮
                                                     Asignación vBAR=pBAR del código rombios.c original.
218         /* Map the I/O memory and port resources. */
219         for ( bar = 0; bar < 7; bar++ )
220         {
221                                 ⋮
222
```

Fig. 62 - Código *rombios.c* tras la modificación

Antes de generar el parche se debe incluir la definición de la variable *gfx_bdf* en el código C. En el antiguo parche "Xen-Load-VBios-File" esta variable se incluía así:

```
19 +/* virtual BDF of pass-throughed gfx */
20 +static uint8_t gfx_bdf;
21 +
```

Fig. 63 - Definición de la variable gfx_bdf en el parche *Xen-Load-VBios-File*

Esta variable aparecía también en el parche Xen-pBAR-vBAR, y ambos parches la utilizaban en el interior del archivo *hvmloader.c*, no obstante este archivo ha sido dividido en dos en la versión 4.2 de Xen. Por un lado está el archivo *rombios.c* y por otro lado el archivo *hvmloader.c*. Es importante mantener la misma funcionalidad de antes y para ello la variable *gfx_bdf* debe ser accesible en estos dos archivos. La variable *gfx_bdf* obtiene su valor en la ejecución del archivo *rombios.c* y este valor es el que se utiliza cuando se ejecuta *hvmloader.c*. Para que esta operación sea posible y la variable no sea tratada como dos variables diferentes en cada uno de los archivos es necesario utilizar el mecanismo *extern* del lenguaje C. Como ambos archivos utilizan el fichero de cabecera *Config.h* se incluirá aquí la declaración de la variable *extern* y en el archivo *rombios.c* se incluirá su definición:

```
43 #define ROMBIOS_END             (ROMBIOS_BEGIN + ROMBIOS_SIZE)
44
45 /*PARCHE*/
46 uint8_t gfx_bdf;
47 /*PARCHE END*/
48
```

Fig. 64 - Segunda modificación del archivo *rombios.c*

Finalmente, generaremos el parche *rombios.patch*:

```
Xen-23350/#  diff  -Naur  tools/firmware/hvmloader/rombios.orig
tools/firmware/hvmloader/rombios.c > rombios.patch
```

10.4 *Config.patch*

Tras modificar ligeramente el archivo *Config.h* como se comentó arriba, se puede generar el parche como en los demás casos quedando éste tal y como se muestra en la siguiente imagen:

```
1 --- tools/firmware/hvmloader/config.orig        2011-05-16 17:07:16.705512331 +0200
2 +++ tools/firmware/hvmloader/config.h   2011-05-16 17:05:49.073488986 +0200
3 @@ -5,6 +5,9 @@
4
5  enum virtual_vga { VGA_none, VGA_std, VGA_cirrus, VGA_pt } virtual_vga;
6  extern enum virtual_vga virtual_vga;
7 +//PARCHE//
8 +extern uint8_t gfx_bdf;
9 +//PARCHE_END//
10
11  struct bios_config {
12      const char *name;
```

Fig. 65 - Parche *Config.patch* completo

10.5 *Hvmloader.patch*

Dado que todo el contenido del parche original Xen-vBAR-pBAR correspondiente al archivo *hvmloader.c* de la versión 3.5 se ha convertido en el actual parche *rombios.patch*, el parche *hvmloader.patch* contendrá sólo las modificaciones que incluía el parche *Xen-Load-vBios-File*. Este parche es fácil de adaptar ya que sólo se debe buscar el sitio adecuado en el que incluir la modificación. Las operaciones que realizaba este antiguo parche se ven a continuación:

```
39          printf("Loading VGABIOS of passthroughed gfx ...\n");
40 -        vgabios_sz =
41 -            round_option_rom((*(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS+2)) * 512);
42 +        memcpy((void *)VGABIOS_PHYSICAL_ADDRESS,
43 +               vgabios_pt, sizeof(vgabios_pt));
44 +        *(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS + sizeof(vgabios_pt)) = gfx_bdf;
45 +        vgabios_sz = round_option_rom(sizeof(vgabios_pt) + 1);
46          break;
47      default:
48          printf("No emulated VGA adaptor ...\n");
```

Fig. 66 - Extracto de código del parche *Xen-Load-VBios-File*

Buscando las líneas 40 o 41 en el código actual se encuentra el sitio donde se debe incluir la modificación:

```
434     case VGA_pt:
435         printf("Loading VGABIOS of passthroughed gfx ...\n");
436         vgabios_sz =
437             round_option_rom((*(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS+2)) * 512);
438         break;
```

Fig. 67 - Imagen del código *hvmloader.c* actual

Sustituyendo las líneas 435-437 por las del parche (42-45) tenemos el nuevo archivo *hvmloader.c,* al igual que antes se crea el parche con la herramienta *diff*:

```
Xen-23350/#  diff  -Naur  tools/firmware/hvmloader/hvmloader.orig
tools/firmware/hvmloader/hvmloader.c > hvmloader.patch
```

```
16      case VGA_pt:
17          printf("Loading VGABIOS of passthroughed gfx ...\n");
18 -        vgabios_sz =
19 -            round_option_rom((*(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS+2)) * 512);
20 +//#        vgabios_sz =
21 +//#            round_option_rom((*(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS+2)) * 512);
22 +//#PARCHE1#
23 +
24 +        memcpy((void *)VGABIOS_PHYSICAL_ADDRESS,
25 +               vgabios_pt, sizeof(vgabios_pt));
26 +        *(uint8_t *)(VGABIOS_PHYSICAL_ADDRESS + sizeof(vgabios_pt)) = gfx_bdf;
27 +        vgabios_sz = round_option_rom(sizeof(vgabios_pt) + 1);
28          break;
29      default:
30 +//#PARCHE1END#
```

Fig. 68 - Parche completo *hvmloader.patch*

10.6 *Dsdt.patch*

La otra parte del parche Xen-vBAR-pBAR modificaba el archivo *./hvmloader/acpi/dsdt.asl* que también existe en el código actual y que contiene la tabla DSDT[44] de Xen. A continuación se muestra la modificación que se realizaba en esta tabla:

```
diff -r 96b634bf65c3 tools/firmware/hvmloader/acpi/dsdt.asl
--- a/tools/firmware/hvmloader/acpi/dsdt.asl    Mon Aug 31 13:14:47 2009 +0800
+++ b/tools/firmware/hvmloader/acpi/dsdt.asl    Mon Aug 31 16:03:27 2009 +0800
@@ -175,6 +175,34 @@ DefinitionBlock ("DSDT.aml", "DSDT", 2,
                         0x000BFFFF,
                         0x00000000,
                         0x00020000)
+
+                    /* reserve MMIO BARs of gfx for 1:1 mapping */
+                    DWordMemory(
+                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
+                        Cacheable, ReadWrite,
+                        0x00000000,
+                        0xE0000000,
+                        0xEFFFFFFF,
+                        0x00000000,
+                        0x10000000)
+
+                    DWordMemory(
+                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
+                        NonCacheable, ReadWrite,
+                        0x00000000,
+                        0xC0000000,
+                        0xC1FFFFFF,
+                        0x00000000,
+                        0x02000000)
+
+                    DWordMemory(
+                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
+                        NonCacheable, ReadWrite,
+                        0x00000000,
+                        0xC2000000,
+                        0xC2FFFFFF,
+                        0x00000000,
+                        0x01000000)

                     DWordMemory(
                         ResourceProducer, PosDecode, MinFixed, MaxFixed,
--- a/tools/firmware/hvmloader/hvmloader.c      Mon Aug 31 13:14:47 2009 +0800
```

Fig. 69 - Modificación del archivo dsdt.asl que realizaba el parche Xen-vBAR-pBAR

Lo que realiza esta modificación es simplemente añadir los registros base de la tarjeta gráfica que se desea virtualizar a la tabla DSDT de Xen. Sin embargo, estos registros son diferentes para cada tarjeta gráfica y para cada máquina física. Por lo tanto no es posible crear un parche que funcione a nivel general sino que cada usuario debe modificar la tabla DSDT añadiendo manualmente los registros de su tarjeta gráfica si pretende utilizarla en un huésped virtual de Xen.

Lo que sí se puede utilizar de este parche es la posición en la que se deben incluir los rangos de direcciones de la tarjeta gráfica, esto es justo después de los valores

[44] DSDT: "Differentiated System Description Table" son una serie de tablas que suministran informaciones varias sobre la configuración de los distintos dispositivos al sistema operativo, en ellas están definidas por ejemplo, tipo de chip de sonido, salidas de video, capacidad para hibernación, reiniciado, apagado, suspensión, número de procesadores etc.

0x000BFFFF, 0x00000000, 0x00020000 que se pueden observar en las líneas 5, 6 y 7 del parche. Buscando estos valores en el código actual del archivo *dsdt.asl* se obtiene:

```
166                     /* reserve memory for pci devices */
167                     DWordMemory(
168                         ResourceProducer, PosDecode, MinFixed, MaxFixed,
169                         Cacheable, ReadWrite,
170                         0x00000000,
171                         0x000A0000,
172                         0x000BFFFF,
173                         0x00000000,
174                         0x00020000)
```

Fig. 70 - Captura del archivo dsdt.asl actual

Justo debajo es donde se deben incluir las modificaciones, lo más cómodo es utilizar un parche que copie los registros que se añadían con el parche antiguo y luego modificarlos según cada caso por lo que lo único útil del nuevo parche será la posición donde se introducirán las modificaciones, que además solo ha cambiado en dos líneas con respecto al pasado:

```
1  --- tools/firmware/hvmloader/acpi/dsdt.asl.orig 2011-05-13 15:34:47.653769584 +0200
2  +++ tools/firmware/hvmloader/acpi/dsdt.asl      2011-05-13 13:57:57.718003194 +0200
3  @@ -173,6 +173,34 @@
4                           0x00000000,
5                           0x00020000)
6
7  +                    /* reserve MMIO BARs of gfx for 1:1 mapping */
8  +                    DWordMemory(
9  +                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
10 +                        Cacheable, ReadWrite,
11 +                        0x00000000,
12 +                        0xE0000000,
13 +                        0xEFFFFFFF,
14 +                        0x00000000,
15 +                        0x10000000)
16 +
17 +                    DWordMemory(
18 +                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
19 +                        NonCacheable, ReadWrite,
20 +                        0x00000000,
21 +                        0xC0000000,
22 +                        0xC1FFFFFF,
23 +                        0x00000000,
24 +                        0x02000000)
25 +
26 +                    DWordMemory(
27 +                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
28 +                        NonCacheable, ReadWrite,
29 +                        0x00000000,
30 +                        0xC2000000,
31 +                        0xC2FFFFFF,
32 +                        0x00000000,
33 +                        0x01000000)
34 +
35                      DWordMemory(
36                          ResourceProducer, PosDecode, MinFixed, MaxFixed,
37                          Cacheable, ReadWrite,
```

Fig. 71 - Parche dsdt.patch adaptado a la versión 4.2 de Xen

Tras aplicar el parche mediante la herramienta *patch*:

```
Xen-23350# patch -p0 < dsdt.patch
patching file tools/firmware/hvmloader/acpi/dsdt.asl
```

Se debe buscar en el archivo dsdt.asl la cadena de texto `/*reserve MMIO BARS of gfx for 1:1 mapping*/` y en el bloque siguiente incluir las modificaciones. A continuación se presentan unos ejemplos que muestran como se puede obtener la información necesaria para dichas modificaciones.

Ejemplo 1:

Lo primero es obtener el identificador BDF de la tarjeta gráfica:

```
# lspci |grep VGA
01:00.0 VGA compatible controller: nVIDIA Corporation Device 0de0
(rev a1)
```

Se puede ver como la tarjeta tiene asociado el identificador 01:00.0, con este dato se pueden obtener los rangos de memoria de sus registros de entrada/salida:

```
# dmesg | grep 01:00.0 |grep mem
[    3.124546] pci 0000:01:00.0: reg 10: [mem 0xfa000000-0xfaffffff]
[    3.124565] pci 0000:01:00.0: reg 14: [mem 0xc0000000-0xcfffffff 64bit
pref]
[    3.124584] pci 0000:01:00.0: reg 1c: [mem 0xd0000000-0xd1ffffff 64bit
pref]
[    3.124609] pci 0000:01:00.0: reg 30: [mem 0xfb000000-0xfb07ffff pref]
[    3.199436] vgaarb: device added:
PCI:0000:01:00.0,decodes=io+mem,owns=io+mem,locks=none
[    3.199730] pci 0000:01:00.0: BAR 0: reserving [mem 0xfa000000-
0xfaffffff flags 0x40200] (d=0, p=0)
[    3.199732] pci 0000:01:00.0: BAR 1: reserving [mem 0xc0000000-
0xcfffffff flags 0x14220c] (d=0, p=0)
[    3.199735] pci 0000:01:00.0: BAR 3: reserving [mem 0xd0000000-
0xd1ffffff flags 0x14220c] (d=0, p=0)
```

Los registros son cuatro:

```
reg 10: [mem 0xfa000000-0xfaffffff]

reg 14: [mem 0xc0000000-0xcfffffff 64bit

reg 1c: [mem 0xd0000000-0xd1ffffff 64bit

reg 30: [mem 0xfb000000-0xfb07ffff pref]
```

Para cada uno de los registros se debe rellenar un bloque teniendo en cuenta las siguientes indicaciones.

```
/* reserve MMIO BARs of gfx for 1:1 mapping */
                    DWordMemory(          (Bloque para el primer registro)
                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
                        Cacheable, ReadWrite,
                        0x00000000,       (Dejar a cero)
                        0xfa000000,       (Valor mínimo del registro)
                        0xfaffffff,       (Valor máximo del registro)
                        0x00000000,       (Dejar a cero)
                        0x01000000)       (Rango del registro, Max-Min+1)
```

En el último bloque o registro se debe incluir una línea adicional:

```
      DWordMemory(                        (Bloque para el último registro)
                        ResourceProducer, PosDecode, MinFixed, MaxFixed,
                        Cacheable, ReadWrite,
                        0x00000000,
```

```
                0xFB000000,
                0xFB07FFFF,
                0x00000000,
                0x00080000,
                ,, _Y01)          (Línea adicional de cierre)
        })
```

Ejemplo 2:

Este ejemplo tan sólo muestra una salida ligeramente diferente del programa *dmesg*, la tarjeta gráfica tiene asociado el identificador BDF 08:00.0:

```
# dmesg | grep 08:00.0
pci 0000:08:00.0: reg 10 32bit mmio: [0xf8000000-0xf8ffffff]
pci 0000:08:00.0: reg 14 64bit mmio pref: [0xd0000000-0xdfffffff]
pci 0000:08:00.0: reg 1c 64bit mmio: [0xf6000000-0xf7ffffff]
pci 0000:08:00.0: reg 24 io port: [0xec00-0xec7f]
pci 0000:08:00.0: reg 30 32bit mmio pref: [0xf5f80000-0xf5ffffff]
vgaarb: device added: PCI:0000:08:00.0,decodes=io+mem,owns=none,locks=none
```

En este caso se observan cuatro registros y un puerto de entrada/salida, se procederá de la misma forma que en el ejemplo anterior, teniendo en cuenta solamente los registros MMIO[45]. Por tanto, habrá que rellenar los bloques ignorando el puerto I/O utilizando únicamente los rangos siguientes:

```
reg 10 32bit mmio: [0xf8000000-0xf8ffffff]
reg 14 64bit mmio pref: [0xd0000000-0xdfffffff]
1c 64bit mmio: [0xf6000000-0xf7ffffff]
reg 30 32bit mmio pref: [0xf5f80000-0xf5ffffff]
```

Nota: Todos los parches pueden encontrarse en el CD-ROM adjunto al proyecto, o bien obtenerse de la siguiente URL:

http://old-list-archives.xen.org/archives/html/xen-devel/2011-05/msg01426.html

[45] MMIO: Memory-mapped I/O y port I/O son dos métodos complementarios para realizar operaciones de entrada/salida entre la CPU y los dispositivos periféricos en un ordenador.

11

Configuración de un Entorno Xen con Soporte *VGA-Passthrough*

11.1 Extracción de la BIOS de la tarjeta gráfica

Como ya se ha comentado anteriormente, uno de los problemas que se debe solventar cuando se realiza *VGA-passthrough* es realizar una lectura limpia de la BIOS gráfica, ya que después de ejecutarse por primera vez algunos bits de esta zona de memoria pueden ser parcialmente truncados. Este problema se conoce como *re-execution issue* y sólo afecta a algunas tarjetas gráficas.

La mejor forma de extraer la BIOS VGA es hacerlo desde un sistema operativo que la utilice lo menos posible como, por ejemplo, MS-DOS. En el caso de la tarjeta Geforce GTX-460 necesitamos utilizar la herramienta que proporciona nVIDIA para actualizar la BIOS de las tarjetas gráficas llamada 'nvflash'.

Los pasos a seguir para extraer la BIOS utilizando esa herramienta son los siguientes:

1. Descargar los archivos necesarios de un disco de arranque en MS-DOS desde la siguiente url: http://files.extremeoverclocking.com/file.php?f=196

2. Descargar la herramienta HP USB Disk Storage Format Tool, que puede encontrase en la siguiente dirección: http://files.extremeoverclocking.com/file.php?f=197

3. Tras instalar el programa, introducir un USB en el PC y ejecutando el programa como administrador se debe seleccionar la opción 'Create a DOS startup disk'. Presionar el botón (...) para buscar los archivos descargados en el paso '1' Y seleccionar los archivos de arranque:

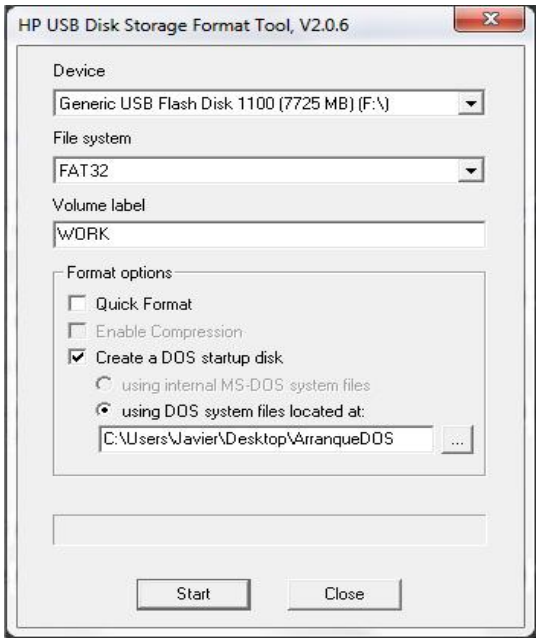


Fig. 72 - Programa HP USB Disk Storage Format Tool

4. Tras tener el USB de arranque se debe descargar la herramienta nvflash de nVIDIA y copiarla en el USB. Se puede descargar en la siguiente dirección: http://downloads.guru3d.com/NVFlash-5.95.0.1-download-2590.html

5. Reiniciar el PC arrancando desde el USB, para ello asegurarse de que en la BIOS se cambia la prioridad de arranque.

6. Ejecutar la orden de respaldo de la BIOS tal y como se muestra en la imagen de abajo:

```
nvflash -b oldbios.rom
```

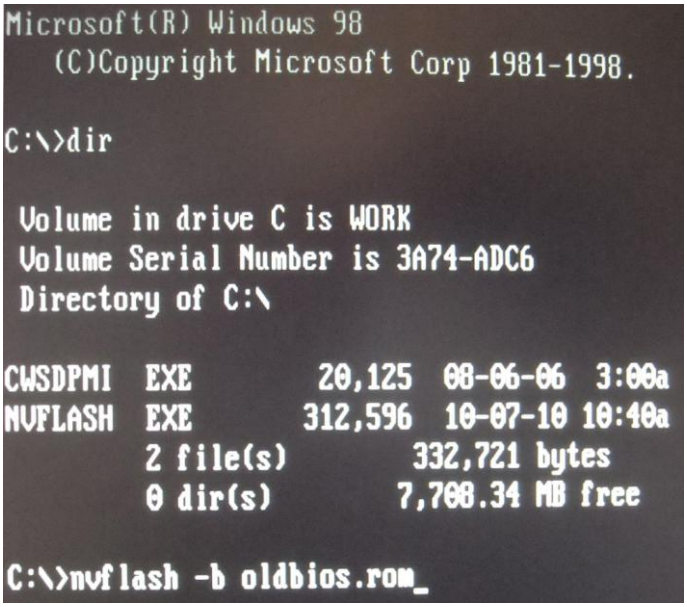


Fig. 73 - Ejecución de nvflash

7. Tras ejecutar la orden, si todo sale bien se observará una salida similar a esta:

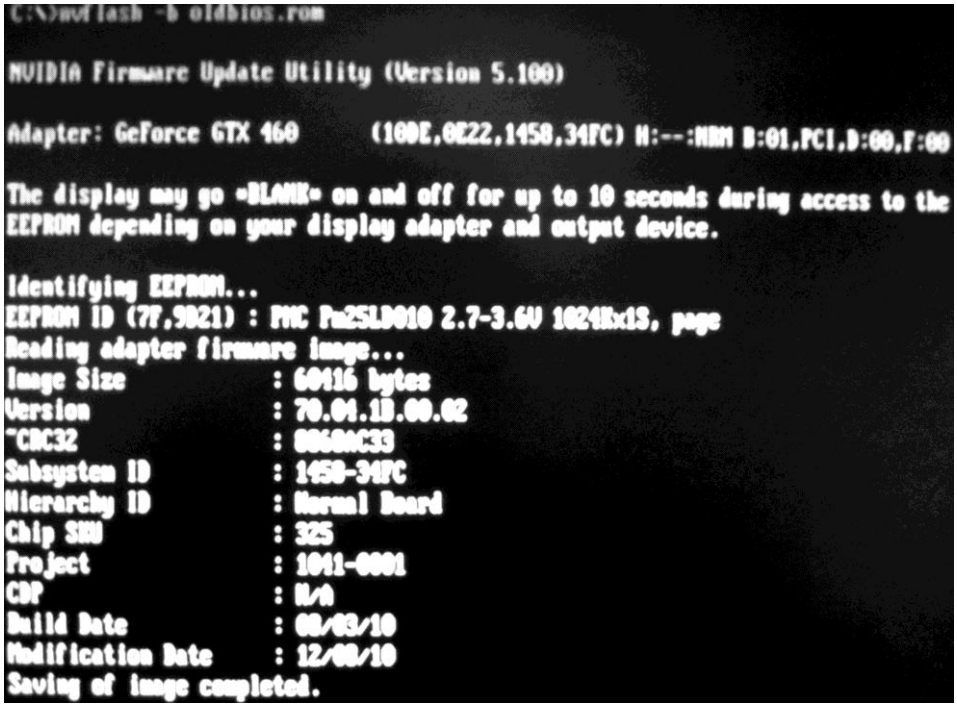


Fig. 74 - Nvflash, Saving of image completed

Tras este último paso, se puede comprobar cómo en el USB ha quedado almacenado la BIOS gráfica con el nombre oldbios.rom, tal y como se muestra en la siguiente figura:

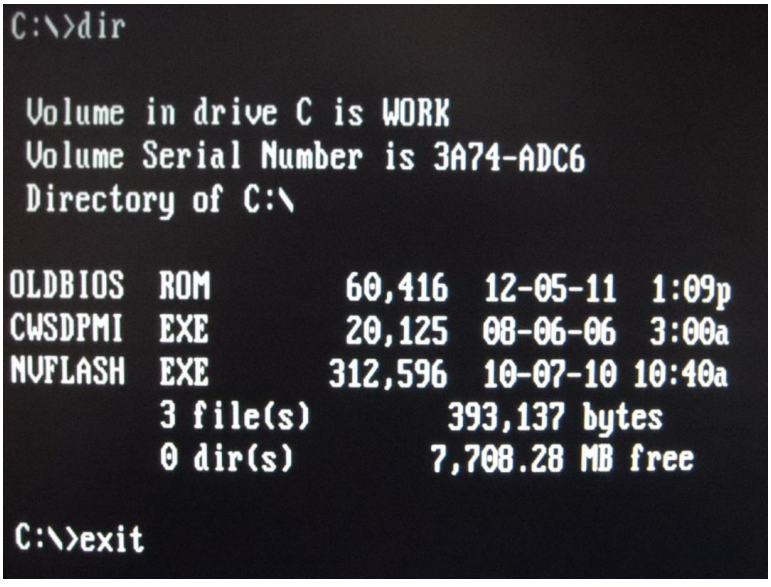


Fig. 75 - Bios gráfica extraída

11.2 Instalación de XEN para utilizar *VGA-Passthrough*

11.2.1 Requisitos

Lo primero que se debe hacer si se pretende utilizar la característica *passthrough* de Xen es verificar si se tienen los elementos necesarios para ello. Se necesita un procesador con extensiones de virtualización Intel (Vt-d) o AMD (IOMMU) y una placa base capaz de utilizar las extensiones. Una vez conseguido esto se necesita instalar el sistema operativo Debian en la máquina en cuestión para posteriormente instalar Xen.

Para comprobar si las extensiones de virtualización están presentes en la máquina se puede ejecutar la siguiente orden:

```
# grep -E –color 'svm|vmx' /proc/cpuinfo
```

Si el comando anterior produce alguna salida entonces nuestro procesador soporta virtualización por hardware, de lo contrario no se podrá utilizar ese ordenador para utilizar *VGA-Passthrough*.

Las extensiones de virtualización se deben habilitar en la BIOS del sistema. Es importante que existan no solo tecnologías de virtualización sino también extensiones de virtualización (VT-d para Intel o IOMMU para AMD). En la siguiente imagen puede verse un ejemplo de lo que la BIOS debe mostrar en un ordenador Intel capacitado para esta tarea:

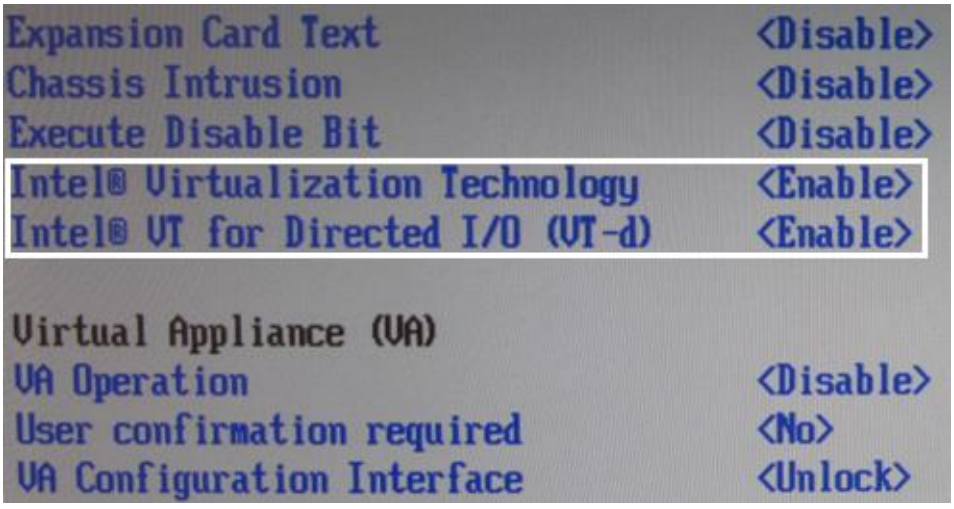


Fig. 76 - Imagen BIOS de la placa base Intel DQ35MPE

Para poder instalar Xen se necesita que el kernel de Linux utilizado incluya soporte para Xen, de lo contrario no se podrá ejecutar el *Dom 0* y el entorno virtual no funcionará. Existen kernels precompilados de Xen pero estos no incluyen todas las características

necesarias para la utilización de *VGA-passthrough,* por lo que lo más sencillo es recompilar el kernel añadiéndole soporte para Xen y las características que interesen.

Si se desea un kernel más reciente que el utilizado por el sistema operativo se puede obtener cualquiera de ellos desde los repositorios git[46].

Si se pretende utilizar la tarjeta gráfica nativa en un huésped con Windows 7 se debe utilizar un kernel de versión superior o igual a 3.1-rc9 de lo contrario solo será posible utilizarla en máquinas virtuales WinXP.

Para modificar el kernel actual utilizado por Debian, se deben descargar las fuentes de este. En los repositorios Debian se pueden encontrar con el nombre kernel-source:

```
# apt-get install kernel-source
```

Con esto se descargará el archivo kernel-source-2.X.X.bz2 en el directorio `/usr/src`.

11.2.2 Recompilar el kernel para incluir el soporte a Xen:

Una vez obtenido el kernel a modificar se accede a su directorio y se ejecuta la herramienta *menuconfig*:

```
# make menuconfig
```

Tras ejecutar el comando se mostrará el menú de configuración del kernel:

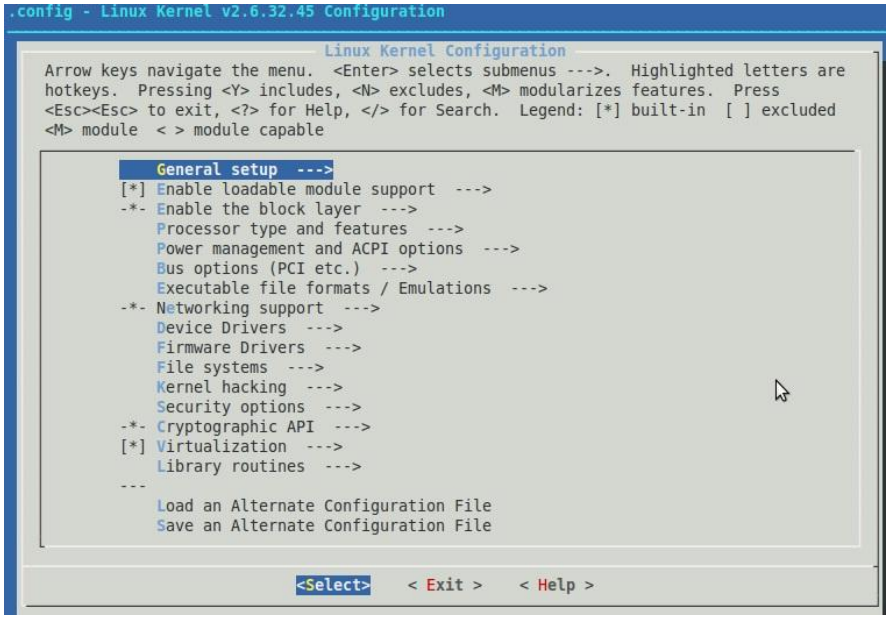


Fig. 77 – Menuconfig del Kernel

[46] Kernels de Linux en github: https://github.com/mirrors/linux-2.6/tags

Para que las opciones de Xen estén disponibles en el menú, se deben habilitar otras antes de las que depende el funcionamiento de Xen. En concreto hay dos opciones que una vez incluidas en el kernel provocan la aparición de nuevos menús para configurar Xen. La primera de ellas es la llamada *PAE (Physical Address Extension)* que provoca la aparición de la segunda, *High Memory Support*, que se ubica dentro del submenú *Processor type and features* y en la que se debe seleccionar 64 GB tal y como se muestra en la siguiente figura:

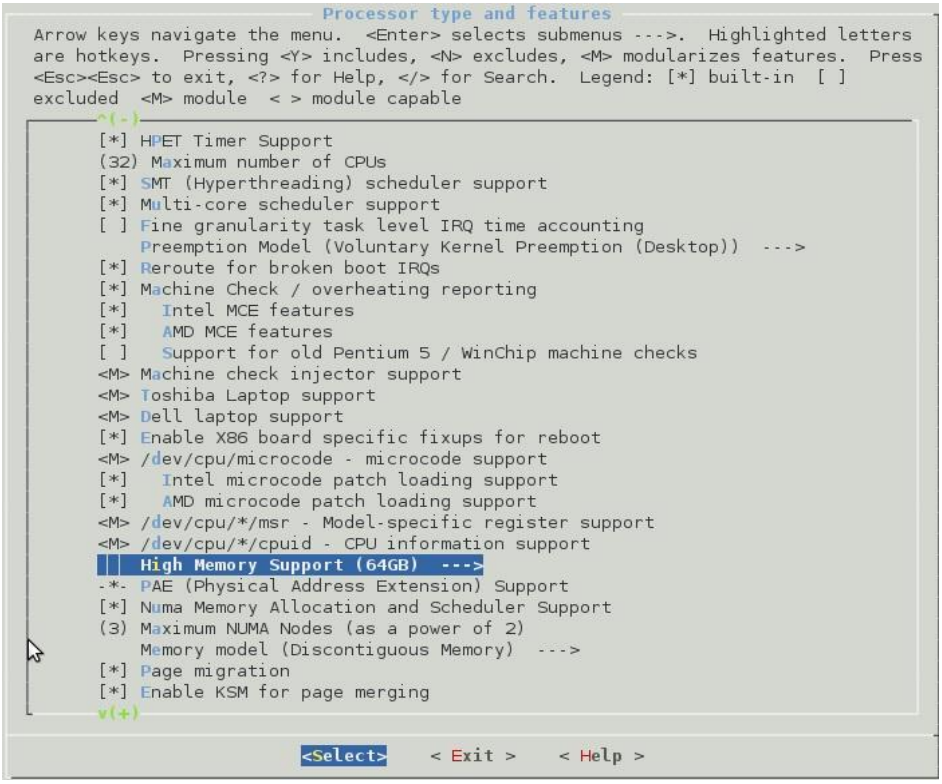


Fig. 78 - Requisito para Xen, activar PAE

Tras activar esta opción, en el submenú *Paravirtualized guest support* dentro del actual menú *Processor type and features* aparece la segunda opción que condiciona el uso de Xen, *Xen guest support* **:**

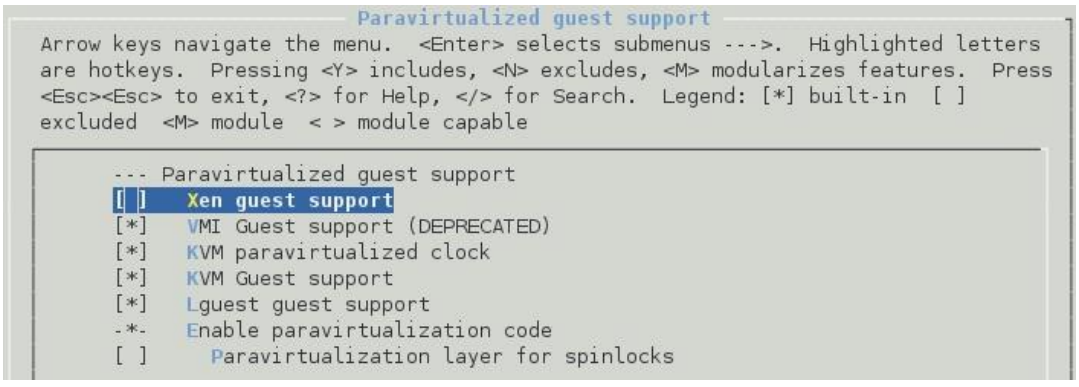


Fig. 79 – Menuconfig, activar Xen guest support

Tras activar esta opción, aparecerán todas las que están relacionadas con Xen que serán añadidas automáticamente como cadenas de texto al archivo *.config.* Para terminar de editar la configuración del kernel que pretendemos crear, podemos seguir empleando el menú o editar manualmente el archivo *.config*. A continuación, se muestran algunas capturas de cómo debe quedar el archivo de configuración haciendo hincapié en las opciones clave que se deben incluir:

```
CONFIG_PARAVIRT_GUEST=y
CONFIG_XEN=y
CONFIG_XEN_PVHVM=y
CONFIG_XEN_MAX_DOMAIN_MEMORY=128
CONFIG_XEN_SAVE_RESTORE=y
CONFIG_XEN_DEBUG_FS=y
CONFIG_SWIOTLB_XEN=y
CONFIG_MICROCODE_XEN=y
CONFIG_XEN_DOM0=y
CONFIG_XEN_PRIVILEGED_GUEST=y
CONFIG_XEN_DOM0_PCI=y
CONFIG_XEN_PCI_PASSTHROUGH=y
CONFIG_VMI=y
# CONFIG_KVM_CLOCK is not set
# CONFIG_KVM_GUEST is not set
```

La mayoría de estas características se añaden por defecto al habilitar Xen pero hay que asegurarse que la opción *CONFIG_XEN_PCI_PASSTHROUGH* está incluida en el kernel.

Otra de las opciones necesarias para realizar passthrough es el módulo *PCI_STUB* que se utilizará para ocultar la tarjeta gráfica al Dominio 0:

```
CONFIG_PCI_STUB=m
CONFIG_XEN_PCIDEV_FRONTEND=y
```

Finalmente hay que asegurarse de que el kernel Dom0 de Linux incluye las opciones *CONFIG_XEN_DEV_EVTCHN* y *CONFIG_XEN_PCIDEV_BACKEND_PASS*[47] y de que la opción *CONFIG_XEN_PCIDEV_BACKEND_VPCI* esté deshabilitada, tal y como se muestra en la siguiente imagen.

[47] El modo `BACKEND_PASS` implica que los identificadores PCI de los dispositivos PCI (como la tarjeta gráfica) serán exactamente los mismos en la máquina virtual y en el Dom0/host.

```
CONFIG_XEN_BALLOON=y
CONFIG_XEN_SCRUB_PAGES=y
CONFIG_XEN_DEV_EVTCHN=y
CONFIG_XEN_BACKEND=y
CONFIG_XEN_NETDEV_BACKEND=y
CONFIG_XEN_BLKDEV_BACKEND=y
CONFIG_XEN_BLKDEV_TAP=m
CONFIG_XEN_BLKBACK_PAGEMAP=y
CONFIG_XEN_PCIDEV_BACKEND=y
# CONFIG_XEN_PCIDEV_BACKEND_VPCI is not set
CONFIG_XEN_PCIDEV_BACKEND_PASS=y
# CONFIG_XEN_PCIDEV_BACKEND_SLOT is not set
# CONFIG_XEN_PCIDEV_BACKEND_CONTROLLER is not set
# CONFIG_XEN_PCIDEV_BE_DEBUG is not set
CONFIG_XENFS=y
CONFIG_XEN_COMPAT_XENFS=y
CONFIG_XEN_SYS_HYPERVISOR=y
CONFIG_XEN_XENBUS_FRONTEND=y
CONFIG_XEN_GNTDEV=y
CONFIG_XEN_S3=y
CONFIG_ACPI_PROCESSOR_XEN=m
CONFIG_XEN_PLATFORM_PCI=y
```

El kernel 2.6.32 de Debian Squeeze, que fue el que se empleó para este desarrollo, incluye por defecto el modo *PCIDEV_BACKEND* equivocado por lo que se debe modificar como se ha indicado y recompilarlo.

Una vez modificado, se debe compilar el kernel. Para compilar la configuración actual basta con ejecutar la orden *make* y esperar.

```
# make –j4
# make –j4 install && make –j4 modules_install
# depmod 2.6.32.45
# mkinitramfs -k -o /boot/initramfs.img-2.6.32.45 2.6.32.45
# update-grub
```

Nota: Para acelerar la compilación utilizar el parámetro *–jNumeroDeProcesadores*. Nótese que debe coincidir la versión del kernel con la de *initramfs* para que Grub2 los relacione adecuadamente.

11.2.3 Descargar el código fuente de Xen y aplicar los parches

En este apartado se describe paso a paso el proceso que debe seguirse para aplicar los parches al código de Xen permitiendo habilitar la característica *VGA Passthrough*.

1. El primer paso es extraer la BIOS de la tarjeta gráfica tal y como se explica en el apartado “11.1 Extracción de la BIOS de la tarjeta gráfica”.
2. Descargar la revisión 23350 del código de Xen desde el sitio oficial:

   ```
   # hg clone -r 23350 http://xenbits.xensource.com/staging/xen-unstable.hg/ xen-unstable.hg-rev-23350
   ```

3. Realizar una compilación previa para generar todos los archivos de código fuente:

   ```
   # cd xen-unstable.hg-rev-23350/
   # cd tools/
   ```

```
# make
# make clean
# cd ..
```

4. Obtener los parches creados en el apartado "10 Implementación de una Solución de *VGA-Passthrough* en Xen". Estos se pueden obtener del CD-ROM adjunto o descargarlos de la lista de Xen-dev[48].
5. Aplicar los parches:

```
# for file in $(ls gfx-passthrough-patchs-23350/*);do patch -p1
< $file;done
```

   El comando anterior supone que los parches se han copiado dentro de un directorio llamado *gfx-passthrough-patchs-23350* dentro de la carpeta del código fuente de Xen.

8. Modificar los rangos de memoria asignados a la tarjeta gráfica en el archivo *tools/firmware/hvmloader/acpi/dsdt.asl* tal y como se explica en el apartado "10.6 *Dsdt.patch*".
6. Copiar la BIOS gráfica *vgabios-pt.bin* en el directorio *tools/firmware/vgabios/* .
7. Finalmente compilar e instalar Xen:

```
# make xen && make tools && make stubdom
# make install-xen && make install-tools PYTHON_PREFIX_ARG= &&
make install-stubdom && reboot
```

Tras realizar estas operaciones, se debe iniciar Xen y realizar la siguiente comprobación para verificar el correcto funcionamiento de este:

```
# xm list
```

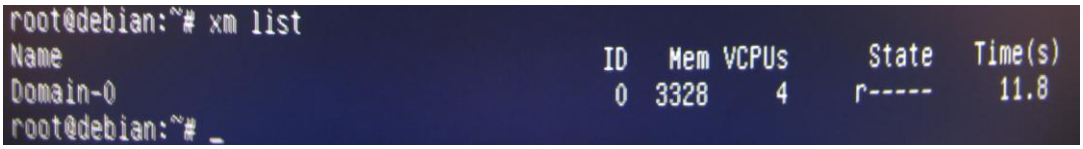


Fig. 80 - Salida normal del comando xm list, funcionamiento correcto.

La salida del comando *xm list* muestra todos los dominios Xen funcionando. Tras iniciar Xen se debe comprobar que aparece ejecutándose el *Domain 0*. Si este no puede ejecutarse ocurrirá un error. En el Anexo II "Problemas comunes en Xen" se detallan los problemas más comunes y sus posibles soluciones.

[48] Tras actualizar los parches se publicaron en la lsita de desarrollo de Xen:

http://old-list-archives.xen.org/archives/html/xen-devel/2011-05/msg01426.html

Se recomienda ignorar el contenido del mensaje por estar desactualizado, sólo se deben descargar los parches.

11.2.4 Habilitar VT-d en el Grub y comprobar su funcionamiento

Una vez que Xen funciona correctamente se debe activar el uso de las extensiones de virtualización en el sistema Xen. Para hacer esto se debe cambiar la entrada Grub de Xen incluyendo la opción *iommu=1* o *iommu=force* , tal y como se muestra en la siguiente imagen:

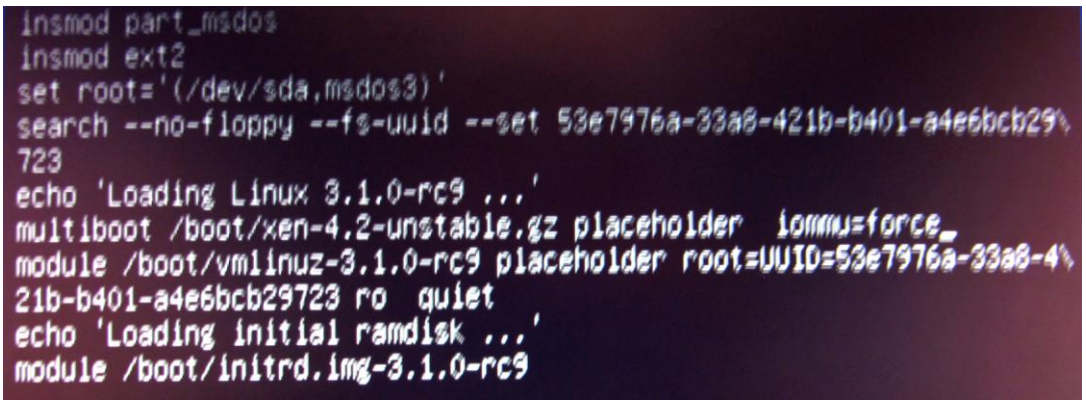


Fig. 81 - Modificación de la entrada del Grub de Xen para habilitar VT-d

La opción *iommu=force* se asegura que el sistema sólo arranque si es posible utilizar la característica IOMMU (VT-d) necesaria para utilizar *PCI-passthrough*, de lo contrario se mostrará un error por pantalla y se reiniciará el PC.

Si se produce algún error la única opción posible es actualizar la BIOS del PC a su versión más reciente y esperar que el error se solucione**.** Si el error persiste quiere decir que la BIOS en cuestión no es capaz de utilizar las partes de VT-d que se necesitan. Los desarrolladores de Xen han implementado soluciones para muchas de las placas base del mercado que solucionan distintos fallos en la implementación de VT-d. Todos estos casos con solución se comprueban al forzar el uso de VT-d mediante la opción *iommu=force*.

Si se utiliza la opción *iommu=1* , se pueden comprobar los resultados tal y como se muestra en la siguiente imagen:

```
root@debian:~# xm dmesg | grep I/O
(XEN) Enabling APIC mode:  Flat.  Using 1 I/O APICs
(XEN) I/O virtualisation enabled
root@debian:~# xm dmesg | grep VT
(XEN) Intel VT-d Snoop Control not enabled.
(XEN) Intel VT-d Dom0 DMA Passthrough not enabled.
(XEN) Intel VT-d Queued Invalidation not enabled.
(XEN) Intel VT-d Interrupt Remapping not enabled.
(XEN) Intel VT-d Shared EPT tables not enabled.
root@debian:~#
```

Fig. 82 - Comprobación VT-d en Xen

Como se puede ver en la imagen, la virtualización está habilitada (*I/O virtualisation enabled*), sin embargo, no es posible habilitar ninguna de las características Vt-d; todas ellas aparecen deshabilitadas (*not enabled*).

Si se intentan arreglar los posibles problemas mediante la opción *iommu=force* se obtiene un error que como ya se ha explicado es irrecuperable:

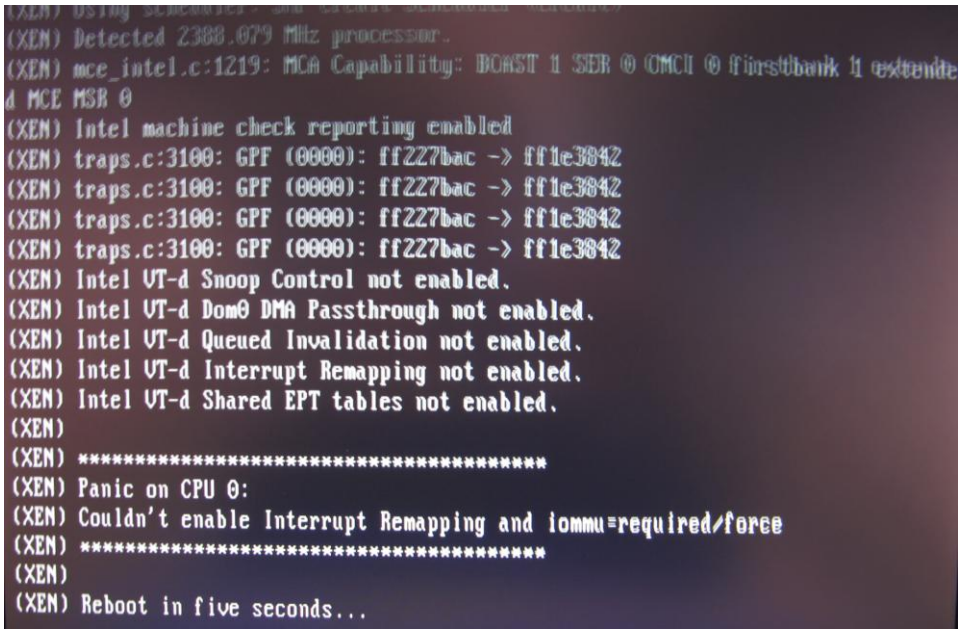


Fig. 83 – Error al habilitar la utilización de las extensiones de virtualización VT-d

Una última opción que ofrece una solución para algunas situaciones consiste en utilizar la opción alternativa *iommu= workaround_bios_bug.*

Aunque no es necesario poder activar todas y cada una de las diferentes opciones VT-d, tal y como se puede observar en la imagen, sí que es necesario disponer de la característica VT-d *Interrupt Remapping*. Un ejemplo de salida que si permite la utilización de VGA-passthrough sería esta:

```
xm dmesg | grep VT-d

(XEN) Intel VT-d Snoop Control not enabled.
(XEN) Intel VT-d Dom0 DMA Passthrough not enabled.
(XEN) Intel VT-d Queued Invalidation enabled.
(XEN) Intel VT-d Interrupt Remapping enabled.
(XEN) Intel VT-d Shared EPT tables not enabled.
```

Si se desea adquirir un procesador con tecnología VT-d que incluya además *VT-d Interrupt Remapping*, lo mejor es cerciorarse leyendo la hoja de especificaciones del mismo ya que la información que se ofrece por defecto no especifica que partes de VT-d han sido implementadas y cuáles no. De hecho, según la información ofrecida por la comunidad Intel actualmente no hay ningún procesador que soporte todas y cada una de las opciones VT-d.[49]

[49] Hilo "Complete VT-d" Junio 2011- http://communities.intel.com/message/129595

11.2.5 Creación y configuración de la máquina virtual

Si no se ha obtenido ningún error como los mencionados anteriormente se puede continuar con el proceso creando la máquina virtual.

El primer paso consiste en crear el archivo que servirá como imagen de la máquina virtual. Para ello lo más sencillo es utilizar el programa *dd*. Se muestra como crear un archivo de nombre "disk.img" con un tamaño de 8GB (8192 bloques de 1MB).

```
# mkdir -p /xen/domains/win01
# dd if=/dev/zero of=/xen/domains/win01/disk.img bs=1M count=8192
```

Tras esto, hay que crear manualmente el archivo de configuración de la máquina virtual */etc/xen/win01.cfg*. Hay mucha información disponible para crear y editar estos ficheros de configuración, el que se muestra a continuación es solo un ejemplo más:

```
kernel = '/usr/lib/xen/boot/hvmloader'
builder = 'hvm'
memory = '512'
device_model='/usr/lib/xen/bin/qemu-dm'
#Number of CPU's
vcpus = 1
# Disks
disk = [ 'file:/xen/domains/win01/disk.img,ioemu:hda,w',
'file:/home/WindowsXP-SP2/image.iso,ioemu:hdc:cdrom,r' ]
# Hostname
name = 'win01'
# Networking
vif = ['type=ioemu, bridge=xenbr0']
#----------------------------------------------------------------
# boot on floppy (a), hard disk (c) or CD-ROM (d)
# default: hard disk, cd-rom, floppy
#----------------------------------------------------------------
boot='cd'
vnc=1
vncviewer=1
acpi = 1
sdl=0
serial='pty'
on_poweroff = 'destroy'
on_reboot = 'restart'
on_crash = 'restart'
```

Para permitir las conexiones VNC desde otras máquinas que no sean *localhost* se debe editar el archivo */etc/xen/xend-config.sxp* y cambiar la línea:

```
(vnc-listen '127.0.0.1')
```

Por esta otra:

```
(vnc-listen '0.0.0.0')
```

Para que el cambio tenga efecto, se debe reiniciar el *hypervisor*:

```
# /etc/init.d/xend restart
```

Se debe tener cuidado al habilitar el acceso VNC ya que no está protegido por contraseña (a menos que se especifique), y cualquiera con acceso al servidor podría conectarse.

Para lanzar la instalación del sistema operativo huésped (en este caso se verá el huésped a través de VNC) ejecutar el comando:

```
# xm create win01.cfg
```

11.2.6 Últimos pasos para utilizar la tarjeta gráfica

Una vez que el sistema ha completado la instalación del sistema operativo y que todo funciona correctamente se puede apagar el huésped y configurar el uso de la tarjeta gráfica.

Para que el huésped sea capaz de utilizar la tarjeta gráfica, ésta se debe ocultar al Dom0 e incluirse en el archivo de configuración del huésped.

Para añadirla al huésped basta con incluir el identificador PCI de la tarjeta gráfica e indicar que se va a realizar *passthrough*:

```
gfx_passthru=1
pci  = [ '01:00.0' ]
```

El identificador pci puede obtenerse con la utilidad *lspci*:

```
# lspci | grep VGA
01:00.0 VGA compatible controller: nVIDIA Corporation Device 0e23
(rev a1)
```

Para ocultar la tarjeta al Dom0 se utilizará el módulo PCI-STUB[50]. Para no tener que realizar siempre estas operaciones lo más cómodo es crear un script, el script variará en función de la salida que muestre la utilidad *lspci*.

En este caso:

```
# lspci -n | grep "01:00.0"
01:00.0 0300: 10de:0e23 (rev a1)
```

Teniendo en cuenta la salida anterior el *script* start_windows.sh es como sigue:

```
# grep -vE '^(#|$)' start_windows.sh
echo "10de 0e23">/sys/bus/pci/drivers/pci-stub/new_id
echo "0000:01:00.0">/sys/bus/pci/devices/0000:01:00.0/driver/unbind
echo "0000:01:00.0">/sys/bus/pci/drivers/pci-stub/bind
xm create /etc/xen/win01.cfg
```

Tras lanzar la creación de la máquina virtual puede ser que se obtenga un error similar a éste:

```
Error: pci: improper device assignment specified: pci: 0000:01:00.1
must be co-assigned to the same guest with 0000:01:00.0, but it is
not owned by pciback or pci-stub.
```

[50] Para más información ver la sección “Binding Devices to PCI-STUB” en la url http://wiki.xen.org/xenwiki/VTdHowTo

Esto se debe a que la tarjeta gráfica incorpora una parte de audio como se puede ver aquí:

```
# lspci | grep nVIDIA
01:00.0 VGA compatible controller: nVIDIA Corporation GF104
[GeForce GTX 460] (rev a1)
01:00.1 Audio device: nVIDIA Corporation GF104 High Definition
Audio Controller (rev a1)
```

Por lo tanto será necesario repetir las operaciones anteriores para el identificador 01:00.1, finalmente el script queda así:

```
#!/bin/bash
echo "10de 0e23">/sys/bus/pci/drivers/pci-stub/new_id
echo "0000:01:00.0">/sys/bus/pci/devices/0000:01:00.0/driver/unbind
echo "0000:01:00.0">/sys/bus/pci/drivers/pci-stub/bind
echo "10de 0beb">/sys/bus/pci/drivers/pci-stub/new_id
echo "0000:01:00.1">/sys/bus/pci/devices/0000:01:00.1/driver/unbind
echo "0000:01:00.1">/sys/bus/pci/drivers/pci-stub/bind
xm create /etc/xen/win01.cfg
```

Se puede comprobar que la tarjeta gráfica puede ser asignada al huésped mediante la orden:

```
# xm pci-list-assignable-devices
01:00.0
01:00.1
```

11.2.7 Instalar el controlador de nVIDIA

Una vez que se tiene la máquina virtual iniciada y que la tarjeta gráfica se ha detectado, se debe instalar la versión adecuada del controlador para que la tarjeta funcione correctamente en la máquina virtual. En nuestro caso al utilizar un huésped Windows se debe instalar la versión 275.33 del controlador pues cualquier otra versión mayor o igual a la 280.XX no funcionará. En el caso de un huésped Linux es preferible utilizar el controlador 275.28 (nVIDIA-Linux-x86_64-275.28.run).

Los controladores para las tarjetas nVIDIA GeForce se pueden encontrar en las siguientes direcciones:

http://www.nvidia.es/object/winxp-275.33-whql-driver-es.html

http://www.nvidia.es/object/winxp64-275.33-whql-driver-es.html

http://www.nvidia.es/object/win7-winvista-32bit-275.33-whql-driver-es.html

http://www.nvidia.es/object/win7-winvista-64bit-275.33-whql-driver-es.html

11.3 Evolución y mantenimiento de los parches

Tras la publicación en la lista oficial de Xen de la adaptación de los parches realizada en este proyecto[51], uno de los usuarios se hizo cargo de mantener estos parches cada vez que aparecía una nueva revisión de Xen 4.2. La adaptación inicial se realizó para la revisión 23350, actualmente los parches se encuentran disponibles hasta la rev. 24798.

En la siguiente dirección se pueden encontrar las sucesivas adaptaciones de los parches:

http://www.davidgis.fr/blog/index.php?2011/12/07/860-xen-42unstable-patches-for-vga-pass-through

También pueden encontrarse algunos datos curiosos como videojuegos testeados con la tecnología *passthrough* o problemas que se pueden dar al reiniciar y otros. En las dos imágenes de abajo se muestra la información del programa CPU-Z ejecutado en una máquina virtual Windows utilizando la tarjeta gráfica nativa:

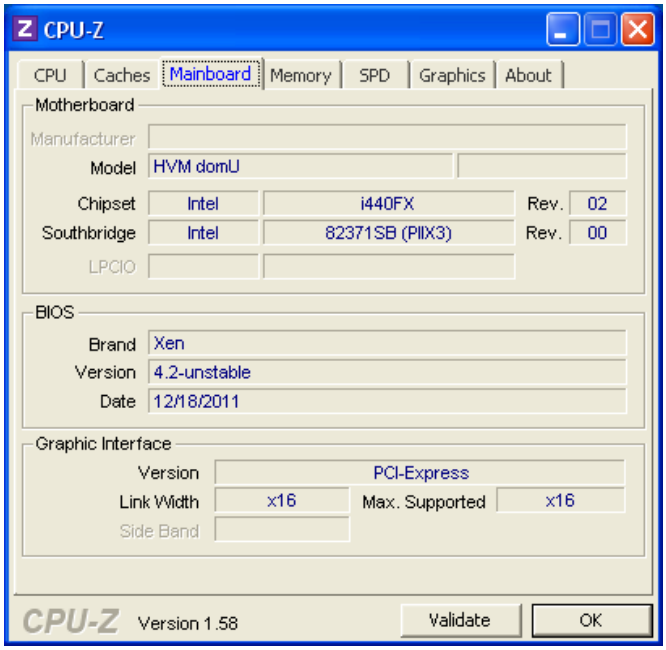


Fig. 84 - Placa base en huésped Xen

[51] Hilo completo sobre los parches actualizados publicados en la lista de Xen:

http://xen.1045712.n5.nabble.com/Patches-for-VGA-Passthrough-XEN-4-2-unstable-td4406265.html

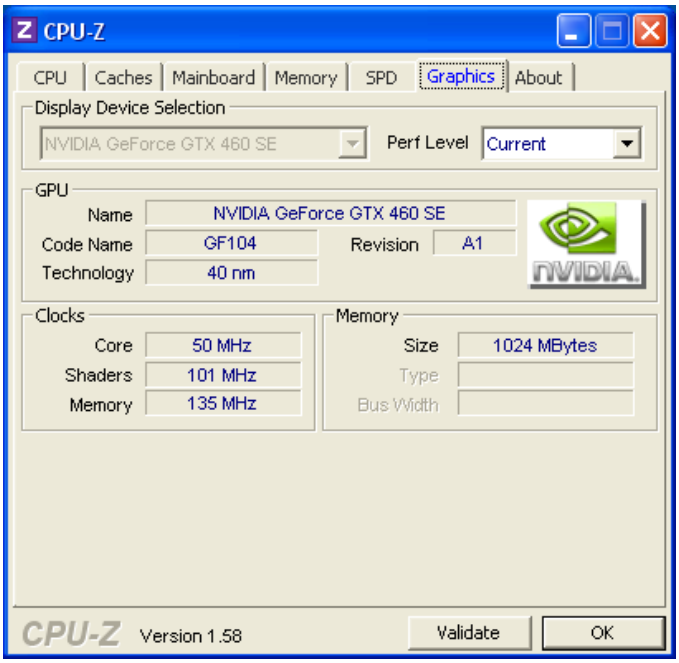


Fig. 85 - Gráficos en huésped Xen con tarjeta gráfica nVIDIA nativa.

Estos parches, no obstante, no son mantenidos de forma oficial debido a que están limitados al uso de tarjetas nVIDIA y provocan incompatibilidades con el soporte existente de las tarjetas ATI.

11.4 Consideraciones finales. ATI vs nVIDIA

Hay que remarcar que las tarjetas gráficas ATI no necesitan ningún tipo de parche para poder utilizarse en el huésped con la versión actual de Xen *unstable*. Tan solo es necesario configurar la máquina virtual tal y como se ha indicado en el apartado "11.2.6 Últimos pasos para utilizar la tarjeta gráfica". De hecho, en el caso de tarjetas ATI se puede incluso utilizar estas como tarjetas secundarias de forma que no haya que utilizar *pci-stub* para desvincular la tarjeta del host, ya que este usaría la tarjeta gráfica primaria.

No obstante, el hecho de haber realizado el esfuerzo de adaptar los parches para nVIDIA se debe sobre todo a que estas tarjetas son en muchos casos las únicas contempladas en programas de seguridad, en concreto el programa forense *Passware* solo permite utilizar tarjetas nVIDIA para acelerar los ataques de archivos cifrados.

Uno de los objetivos de este proyecto es facilitar la tarea de unificar todas las herramientas de seguridad seleccionadas en un único sistema centralizado. La plataforma virtual de Xen colabora en este objetivo al permitir utilizar los programas de seguridad desarrollados para Windows en el sistema final que estará gestionado por el sistema operativo Linux. Será necesario aplicar los parches adaptados en este proyecto para poder utilizar las tarjetas gráficas nVIDIA en el huésped y así disfrutar de todas las ventajas que ofrecen los programas de seguridad como por ejemplo la herramienta de recuperación de información cifrada *Passware*.

12

Recomendaciones para la Construcción de una Plataforma de Descifrado de Archivos

Tras realizar un estudio en profundidad de los resultados obtenidos en cada uno de los apartados del proyecto se recomiendan utilizar las siguientes herramientas para poder construir la plataforma de descifrado de la forma más completa posible:

- La herramienta de código libre ClamAV: Esta herramienta se utilizará para realizar búsquedas de archivos protegidos en el disco. Se recomienda utilizar la futura versión 0.97.5 pues incluirá la detección de archivos de Office mientras que la actual versión 0.97.4 solo detecta archivos comprimidos y archivos PDF cifrados. Al ser una herramienta desarrollada para su utilización por consola es posible configurar la búsqueda con multitud de parámetros por lo que se recomienda su uso frente a la función de búsqueda de *Passware*.
- La herramienta comercial *Passware Kit Forensic*: Como se vió en el apartado 5, el uso de una herramienta comercial de descifrado de archivos sigue siendo necesario y la herramienta *Passware* es una de las soluciones más completas y eficientes del mercado. Esta herramienta servirá tanto para descifrar archivos como para realizar búsquedas de archivos cifrados (en el caso de no disponer de la futura versión de ClamAV). Utilizando los *scripts* AutoIT desarrollados en el proyecto será posible utilizar el programa desde la línea de órdenes paliando así una de las defciencias del programa.
- El programa cRARk: En el caso de querer realizar ataques de ferza bruta contra arhivos comprimidos RAR (por ejemplo para descartar las posibles contraseñas de longitud de 1 a 6 o 7 caracteres) se recomienda utilizar el programa cRARk ya que como se comprobó en el apartado 5.2.3.1, este obtiene rendimientos que la herramienta *Passware*.
- La herramienta de recuperación de contraseñas Hashkill: Esta herramienta servirá tanto para la ruptura de funciones hash como para descifrar archivos comprimidos. Para poder descifrar archivos es necesario utilizar la versión 0.3.0 del programa que no estará disponible hasta verano del 2012.
- Las versiones 'Plus' y 'Lite' de la herramienta OCLHashcat: Estas herramientas junto con el programa Hashkill servirán para realizar ataques contra multitud de funciones hash diferentes con un alto rendimiento. Es importante incluir este tipo

de herramientas en la plataforma de descifrado ya que serán de gran utilidad para la generación de diccionarios personalizados.

- Entorno de virtualización Xen: Para poder unificar todas estas herramientas en un mismo sistema se recomienda utilizar el entorno de virtualización Xen utilizando la implementación de VGA-passthrough realizada en el apartado 10 del proyecto. Remarcar que es importante disponer de una placa base que sea capaz de utilizar correctamente[52] las extensiones de virtualización (VT-d) para que se pueda utilizar la característica VGA-passthrough de Xen.

Para instalar adecuadamente todas estas herramientas en la plataforma de descifrado, se recomienda seguir el siguiente procedimiento:

1) Instalar el sistema operativo Linux distribución Debian Squeeze.
2) Instalar el entorno de virtualización Xen aplicando los parches desarrollados en el proyecto siguiendo las instrucciones del apartado 11.
3) Instalar los drivers de nVIDIA en el sistema operativo Linux.
4) Instalar la herramienta de seguridad Hashkill (versión 0.3.0 o superior) tanto para el descifrado de archivos comprimidos como para la ruptura de hashes.
5) Instalar las herramientas OCLHashcat-plus y OCLHashcat-lite para la ruptura de hashes.
6) Instalar la herramienta cRARk por si se desean realizar ataques de fuerza bruta de forma más eficiente que con el programa *Passware* (deberá compararse el rendimiento con la futura versión 0.3.0 de Hashkill).
7) Instalar la herramienta ClamAV para utilizar su motor de búsqueda de archivos cifrados.
8) Crear la máquina virtual de Windows XP en el entorno de virtualización Xen.
9) Instalar la versión 275.33 del driver nVIDIA en el huésped Windows.
10) Instalar la herramienta de seguridad *Passware Kit Forensic* manteniendo la ruta de instalación por defecto.
11) Instalar el lenguaje de programación de eventos AutoIt descargándolo de la web oficial[53].
12) Copiar la carpeta AutoItScripts del CD-ROM del proyecto a la máquina Windows.
13) Seguir las instrucciones del apartado 7.9 para compilar los scripts en el nuevo sistema.
14) Instalar la herramienta de red NetCat para poder ejecutar los scripts de manera remota tal y como se indica en el apartado 7.2.3.

---

[52] Lista de algunas placas base que permiten utilizar VGA-passthrough: http://wiki.xen.org/wiki/VTd_HowTo

[53] Web oficial de AutoIt: http://www.autoitscript.com/site/autoit/.

12.1 Utilización de la plataforma

Como ejemplo de utilización de la plataforma se propone el siguiente procedimiento:

1) Detectar el tipo de protección que presenta. Esto puede hacerse tanto con el motor de búsqueda ClamAV/ClamWIN como con la herramienta *Passware*.
2) Utilizar la herramienta de generación de diccionarios personalizados proporcionada por el grupo IUICP para generar el archivo de texto que se utilizará en los ataques.
3) Utilizando la misma herramienta se deberá dividir el diccionario en tantas partes como *clusters* o nodos vayan a participar en el descifrado.
4) Seleccionar la herramienta más adecuada para llevar a cabo el descifrado del archivo.
5) En caso de documentos de Office o documentos PDF se deberá utilizar la herramienta *Passware* lanzando el ataque de forma remota en los diferentes nodos utilizando para ello los *scripts* AutoIT desarrollados en el proyecto.
6) En caso de archivos comprimidos si no se dispone de la versión 0.3.0 de Hashkill se lanzará el ataque como en el caso anterior. Si se dispone de la versión 0.3.0 de Hashkill se deberá comparar previamente el rendimiento de los dos programas y elegir el más eficiente.

13

Conclusiones y Trabajo Futuro

Teniendo en cuenta los objetivos que se plantearon al inicio y los resultados obtenidos tras la realización del proyecto podemos obtener las siguientes conclusiones:

1) La seguridad actual de los archivos PDF, ZIP, RAR y Office es bastante robusta y por ello es necesario la utilización de tecnologías GPU y el uso de diccionarios personalizados para su descifrado.
2) El estado actual de las herramientas de libre distribución de descifrado de archivos no es lo suficientemente bueno como para poder sustituir a las herramientas comerciales. No obstante, se ha visto que herramientas como Hashkill están desarrollándose rápidamente y pueden ser utilizadas como alternativa para el desifrado de algunos tipos de archivo.
3) Es posible utilizar tecnologías de computación en la nube (Amazon-EC2) para descifrar archivos, sin embargo, las tarjetas GPU que se utilizan actualmente no son las más adecuadas para este tipo de tareas y se desaconseja su uso.
4) Se ha visto como utilizando el lenguaje de programación de eventos AutoIT es posible automatizar las herramientas de interfaz gráfica para su uso a través de la línea de órdenes. No obstante realizar la automatización completa de una herramienta de descifrado comercial conlleva demasiado tiempo y sólo puede realizarse de manera parcial.
5) Se ha visto que es posible utilizar las tarjetas gráficas GPU en máquinas virtuales utilizando el entorno de virtualización Xen. Sin embargo, esta característica no está todavía lo suficientemente madura y es necesario aplicar una serie de parches actualizados en el proyecto para que se puedan utilizar las tarjetas gráficas nVIDIA (que son las únicas utilizadas por la herramienta *Passware*).
6) Es posible utilizar la plataforma de virtualización Xen para crear un sistema en el que se incluyan las herramientas de descifrado desarrolladas para Windows y las desarrolladas para Linux con la capacidad de utilizar las tarjetas GPU en todas las herramientas.

Considerando todo esto se puede decir que tras la realización del proyecto estamos en disposición de desarrollar una plataforma de descifrado de archivos automatizada que unifique todas las herramientas de descifrado que se deseen utilizar.

Como futuro del proyecto se plantea la creación de la plataforma de descifrado desde el punto de vista *hardware.* Para ello será necesario por un lado utilizar una placa base adecuada que permita VGA-passthorugh y por otro lado se deberá realizar un estudio sobre las diferentes tarjetas gráficas disponibles en el mercado, teniendo en cuenta no

sólo el rendimiento de estas sino también el consumo de energía que realizan. En la red pueden encontrarse muchas comparativas de rendimiento de diferentes modelos de tarjetas GPU pero se recomienda dar prioridad a aquellas comparativas realizadas por herramientas de descifrado o empresas de seguridad. Se propone también realizar un estudio sobre diferentes métodos de refrigeración que servirían para evitar el sobrecalentamiento de las tarjetas GPU de la plataforma.

Otro de los trabajos futuros del proyecto será comparar las nuevas versiones de las herramientas de descifrado seleccionadas para saber en todo momento cuál es la mejor opción para atacar la seguridad de los archivos. Remarcar que en el caso de la herramienta Hashkill se tiene previsto a largo plazo incluir soporte para archivos PDF y archivos de Office lo que nos permitiría prescindir de la herramienta comercial *Passware Kit Forensic*.

También sería interesante comparar el rendimiento de la herramienta comercial *Passware* con otras herramientas comerciales existentes en la actualidad, aunque como ya se mencionó anteriormente el coste de las licencias de estos programas es bastante alto.

Finalmente indicar que una parte importante para el buen funcionamiento de la plataforma se basa en la creación de diccionarios personalizados por lo que como trabajo futuro se propone perfeccionar y optimizar el rendimiento de esta herramienta.

14

Pliego de Condiciones y Presupuesto

En este apartado se hace una estimación del coste que hubiera tenido para una empresa la realización de este proyecto. Al ser este un proyecto de investigación y desarrollo, hay costes que no se pueden calcular con exactitud, como por ejemplo el tiempo de formación en las tecnologías utilizadas y el coste de toda la bibliografía consultada. No obstante, la estimación de tiempos junto con el material utilizado es la parte más significativa.

Ejecución material

Mano de obra

Estudio y adaptación de las herramientas

- 9 meses * 20 días/mes * 5 horas/día = 900 horas
- 40 €/hora

TOTAL: 36.000 €

Redacción de la memoria

- 7 meses * 20 días/mes * 5 horas/día = 700 horas
- 40 €/hora

TOTAL: 28.000 €

Material

Equipos

- Estaciones de trabajo (3 utilizadas): 4500 €
- Tarjetas gráficas GPU (2x GTX 460): 400 €
- Impresora láser: 100€
- Toner: 70 €

TOTAL: 5.070 €

Software

La mayoría del software utilizado es gratuito o de código abierto.

- Sistemas operativos libres: 0 €
  - GNU/Linux Debian
  - GNU/Linux Ubuntu
- Entornos de virtualización: 0 €
  - Xen
  - KVM
  - VMware
  - VirtualBox.
- Herramientas de seguridad de libre distribución: 0 €
  - Búsqueda de archivos cifrados: ClamAV/ClamWin
  - Archivos Word y Excel: Free Word Excel Password
  - Archivos ZIP: PicoZip, FcrackZip, ZipCrack
  - Archivos RAR: UnrarExtract, Igrargpu, cRARk
  - Archivos Comrpimidos y hashes: Hashkill
  - Hashes: OCLHashcat-plus y OCLHashcat-Lite
- Herramienta de programación de eventos AutoIT: 0€
- Sistema operativo comercial *Windows 7 Professional*: 309€
- *Microsoft Office*: 89 €
- Herramienta de seguridad comercial *Passware Kit Forensic* (Licencia de uso personal): 769 €

TOTAL: 1.167 €

TOTAL: 70.237 €

Gastos Generales y Beneficio Industrial

Los gastos derivados de la utilización de las instalaciones de trabajo y el beneficio industrial se estiman en el 16% del coste de ejecución material.

TOTAL: 11.237,92€

Coste de ejecución por Contrata

- **Ejecución material:** 70.237 €
- **Gastos generales y beneficio industrial:** 11.237,92 €

TOTAL: 81.474,92 €

Importe Total

- **Coste de ejecución por contrata:** 81.474,92 €
- **I.V.A(18%):** 14.665,49 €

TOTAL: 96.140,41 €

El presupuesto total del proyecto asciende a la suma de:

Noventa y seis mil ciento cuarenta euros con cuarenta y un céntimos.

Anexo I

Computación en la nube

Una de las opciones que se debe contemplar para la realización de la plataforma de descifrado es la posibilidad de utilizar sistemas de procesado en la nube de forma alternativa o complementaria a la compra de tarjetas gráficas GPU.

Un ejemplo de este tipo de sistemas es el que ofrece la empresa Amazon a través de su servicio Amazon-EC2 que pone a disposición de los usuarios una serie de *clusters* de alto rendimiento basados en tarjetas GPU de nVIDIA. Las características de estos *clusters* son las siguientes:

- 22 GB de memoria.
- 33.5 EC2 Compute Units (una unidad de sistemas EC2 proporciona la capacidad de CPU equivalente de un procesador Opteron 2007 o Xeon 2007 de 1,0-1,2 GHz).
- 2 x nVIDIA Tesla “Fermi” M2050 GPUs.
- 1690 GB de capacidad de almacenamiento.
- Plataforma de 64 bits.
- Alto rendimiento de entrada/salida (10 Gigabit Ethernet).
- API name: cg1.4xlarge.
- Coste del alquiler: 2.10 dólares la hora

La tecnología GPGPU puede ser utilizada con diferentes finalidades y algunas tarjetas gráficas proporcionan un rendimiento mayor que otras dependiendo del ámbito de aplicación. En concreto el modelo de tarjetas gráficas que utiliza Amazon en sus *clusters* (nVIDIA Tesla “Fermi” M2050) no se encuentra entre las mejores opciones para descifrar archivos. Como se explicó en el apartado 3, para generar las claves de cifrado se realizan multitud de iteraciones de algoritmos hash, por lo que una buena referencia para saber si una tarjeta es buena atacando archivos cifrados es saber la velocidad de cálculo de dichos hashes.

En la página de Hashkill aparece una estimación del rendimiento que se obtendría con la mayoría de las tarjetas GPU disponibles en el mercado. La tarjeta Tesla m2050 posee 448 núcleos CUDA que son los mismos núcleos que posee el modelo Tesla c2050. La velocidad de esta tarjeta puede encontrarse en la tabla que proporciona *Hashkill* sobre la serie Fermi sm20 de nVIDIA:

NVidia Fermi architecture series (sm20)

GPU Model	SP Count	GPU Clock	Power, Wload	Pwr eff,MH/W	MD5 MH/s	SHA1 MH/s	SL3, max unlock time
GTX 465	352	1215	200	5	1010	307	37 days, 15 hours
GTX 470	448	1215	215	5	1285	391	29 days, 14 hours
GTX 480	480	1401	250	6	1588	483	23 days, 22 hours
GTX 570	480	1464	219	7	1659	505	22 days, 21 hours
GTX 580	512	1544	244	7	1866	568	20 days, 8 hours
GTX 590	1024	1215	365	8	2938	894	12 days, 22 hours
Tesla C2050	448	1150	238	5	1216	370	31 days, 6 hours
Tesla C2070	448	1150	247	4	1216	370	31 days, 6 hours
Tesla S2050	1792	1150	900	5	4866	1481	7 days, 19 hours

Como se puede ver, de todas las tarjetas de la misma serie solo hay una con un rendimiento peor que la tarjeta Tesla c2050.

Otra de las razones por las que se desaconseja la contratación del servicio Amazon EC2 para labores de descifrado se encuentra en el análisis realizado por la web de seguridad "Security By Default[54]" de este servicio.

En el artículo se comparan las instancias GPU de Amazon con un ordenador que utiliza una tarjeta gráfica GTX460 (la misma disponible para la realización del proyecto). Tras realizar la ruptura de los hashes de ejemplo proporcionados por la herramienta *OCLHashcat* (example.sh) el artículo concluye que "las **2 nVIDIA Tesla M2050** son **más lentas** que **una** tarjeta diseñada para **uso doméstico"**.

[54] www.securitybydefault.com

Anexo II

Problemas comunes en Xen

Xen es una plataforma de virtualización de software libre que trabaja para mejorar y desarrollar características tan interesantes como la de *VGA-Passthrough*, sin embargo, también es cierto que su instalación y configuración pueden ser algo tediosas debido a problemas de dependencias de paquetes o fallos que ocurren en la ejecución de los scripts que utiliza etc. En esta sección se van a exponer los principales problemas que se pueden experimentar con Xen basados en la experiencia obtenida al realizar este proyecto.

Errores de compilación:

Pueden ocurrir errores en la compilación por falta de paquetes o problemas de dependencias, la mejor forma para evitar estos errores es instalar todos los paquetes indicados en la sección 10.1 Passthrough.patch. Otra posibilidad en el caso de no necesitar parchear el código fuente sería instalar Xen desde los paquetes que te proporcione el distribuidor del sistema operativo utilizado.

La mayoría de los errores de compilación obtenidos identifican claramente el paquete perdido y por lo tanto tienen fácil solución, sin embargo, otros muestran un error que no aclara demasiado que dependencia genera el error.

Algunos ejemplos de esto pueden ser:

- Error debido a la ausencia del paquete python-dev:

```
*** OpenSSL headers missing
```

- Error debido al paquete xorg-x11-devel o xorg-x11-dev:

```
Checking check_x11_devel:
*** check_x11_devel FAILED: can't find X11 headers
```

Errores de compatibilidad con python:

Uno de los problemas más frecuentes con Xen se debe a que muchas de sus herramientas están basadas en python y se producen constantemente problemas de rutas erróneas, este problema está siendo subsanado actualmente en las nuevas versiones de Xen utilizando la nueva herramienta “xl” en vez de “xm” que está basada en python. En este apartado se explica cómo solucionar este fallo por si ocurriese tras la instalación.

El problema en cuestión se detecta cuando al iniciarse el demonio xend (necesario para el correcto funcionamiento de la herramienta xm) aparece algo similar a esto:

```
/etc/init.d/xend start
 * Caching service dependencies ...                    [ ok ]
```

```
 * Starting Xen control daemon ...
Traceback (most recent call last):
  File "/usr/sbin/xend", line 36, in <module>
    from xen.xend.server import SrvDaemon
ImportError: No module named xen.xend.server
Traceback (most recent call last):
  File "/usr/sbin/xend", line 36, in <module>
    from xen.xend.server import SrvDaemon
ImportError: No module named xen.xend.server      ...        [ !! ]
```

Como se puede ver se indica que no existe el modulo llamado xen.xend.server, pero podría aparecer cualquier otro modulo de Xen. Para verificar que el problema reside en la ruta de python y no en que realmente se haya perdido un módulo, se procede de la siguiente manera:

```
# python
# import xen
# from xen import xend
# from xend import server
```

Si alguna de las dos últimas operaciones falla el problema se debe a rutas erróneas, esto ocurre porque por defecto la ruta de python es /usr/lib/python2.6/ mientras que Xen busca sus *scripts* en /usr/local/lib/python2.6/dist-packages/ o en una ruta similar.

Hay dos soluciones posibles:

La primera de ellas y más elegante es ejecutar la compilación con el parámetro modificador apropiado:

```
    'make install-tools PYTHON_PREFIX_ARG='
```

La segunda consiste en modificar el archivo Config.mk y eliminar la ruta de python buscando la cadena de texto 'PYTHON_PREFIX_ARG'.

Errores debido a mezcla de versiones

Hay que tener en cuenta que Xen necesita que se haga una instalación limpia y coherente, es decir, las versiones de las herramientas que Xen utiliza (xen-tools) y la versión del software Xen deben coincidir. Si esto no es así puede ocurrir que algunas librerías estén desactualizadas y provoquen errores de lo más dispar.

Un ejemplo de esta situación se da cuando se ha instalado una versión de Xen sobre otra existente provocando que la librería de control “libxc” quede desactualizada y no sea capaz de cargar algunos módulos necesarios para su correcto funcionamiento:

Ejemplo de error al lanzar Xen:

```
Error: (111, 'Connection refused')
Not running and refusing to run: FATAL: Failed to open evtchn
device: No such file or directory
```

Este error no ocurrirá si se recompiló el kernel incluyendo de manera estática los módulos de Xen, sin embargo, si se diera el caso se podría solucionar esto cargando el módulo evtchn de forma manual:

```
$ ls -l /dev/xen
```

```
[...]
crw-r-----. 1 root root  10, 61 2011-05-04 16:50 eventchn
crw-rw----. 1 root root  10, 61 2011-05-04 16:49 evtchn
crw-------. 1 root root  10, 60 2011-06-04 11:17 gntdev
```

Ahora se puede cargar el módulo manualmente y ejecutar uno a uno los demonios de xen:

```
# modprobe xen_evtchn
# xenstored
# service xencommon start
# xend start
```

Forma general de proceder para detectar la raíz de un problema en Xen

A continuación se dan algunas pautas a seguir para aislar el origen de algún problema que ocurra en Xen.

1) El primer paso es asegurarse que el sistema de archivos de Xen está montado correctamente:

   ```
   # ls /proc/xen/
   capabilities  privcmd  xenbus  xsd_kva  xsd_port
   ```

   Si la salida de la orden anterior no mostrase nada, habría que montar el sistema de archivos de Xen manualmente. Para hacer esto basta con añadir al archivo /etc/fstab la línea siguiente y reiniciar el equipo:

   ```
   xenfs      /proc/xen     xenfs     defaults        0       0
   ```

2) El segundo paso es asegurarse que se está ejecutando el kernel Dom0, para ello hay que listar el contenido de /proc/xen/capabilities:

   ```
   # cat /proc/xen/capabilities
   control_d
   ```

   El valor 'control_d' en /proc/xen/capabilities significa que se está ejecutando el kernel apropiado.

3) Asegurarse de que hay archivos en /dev/xen. Al menos deben existir los siguientes nodos:

   ```
   # ls -la /dev/xen
   total 0
   drwxr-xr-x  2 root root     80 Jun 19 19:17 .
   drwxr-xr-x 20 root root   4580 Jun 19 19:33 ..
   crw-rw----  1 root root 10, 58 Jun 19 19:17 evtchn
   crw-rw----  1 root root 10, 57 Jun 19 19:17 gntdev
   ```

4) Finalmente se pueden ejecutar los componentes de Xen uno a uno para observar si alguno de ellos provoca el error:

```
# xenfs
# xenstored
# xenconsoled
# xend
```

Nota: Muchos de los problemas de Xen desaparecen si se utiliza la nueva herramienta de gestión llamada 'xl' en vez de utilizar la antigua herramienta 'xm' que es más propensa a errores y necesita del demonio Xend y de las librerías de python para funcionar.

15

Bibliografía

15.1 Seguridad de archivos

Formato de archivos Office:

http://msdn.microsoft.com/en-us/library/cc313153%28v=office.12%29.aspx

Cryptography Structure Specification:

http://msdn.microsoft.com/en-us/library/cc313071%28v=office.12%29.asp

Fallo de implementación:

Enlace: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.78.2125

Descarga:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.78.2125&rep=rep1&type=pdf

Debilidades en RC4:

http://www.quequero.org/uicwiki/images/Attacks_on_the_RC4_stream_cipher.pdf

Formato de archivos PDF:

http://www.adobe.com/devnet/pdf/pdf_reference_archive.html

Referencia versión 1.6:

http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_archives/PDFReference16.pdf

Referencia versión 1.7:

http://www.adobe.com/content/dam/Adobe/en/devnet/pdf/pdfs/pdf_reference_1-7.pdf

Formato de archivos Zip:

http://www.winzip.com/aes_info.htm

http://www.winzip.com/aes_tips.htm

Analysis of the Winzip Encryption Method:

http://eprint.iacr.org/2004/078.pdf

Seguridad en Winrar:

http://www.winrar.es/soporte/articulo/49

15.2 Alternativas de *software* libre

Herramientas probadas:

http://www.freewordexcelpassword.com/index.php?id=download

http://www.decryptum.com/

http://picozip.com/

http://oldhome.schmorp.de/marc/fcrackzip.html

http://sourceforge.net/projects/zipcrack/

http://www.gat3way.eu/hashkill/index.php

http://www.nvglabs.com/

http://www.golubev.com/rargpu.htm

http://golubev.com/files/igrargpu_v05.zip

http://www.gat3way.eu/forum/viewtopic.php?f=4&t=75

http://www.crark.net/cRARk.html

http://hashcat.net/wiki/oclhashcat_plus

Otra información:

http://hashcat.net/wiki/feature_requests

http://www.tmto.org/docs/HardwareComparisonGTXvsHD.pdf

ClamAV:

http://www.gossamer-threads.com/lists/clamav/users/53191

Wordlist de Kb y Mb:

http://bright-shadows.net/download/downloads.php

Script benchmark OCLHashcat:

http://thepasswordproject.com/oclhashcat-plus-0.07_benchmarking_script_for_linux

OCLHashcat web:

http://hashcat.net/oclhashcat-plus/

Ataques por diccionario con GPU:

http://www.accentsoft.com/helpdesk/ticket.php?track=7YZ-RA6-U999

http://www.gat3way.eu/hashkill/index.php?page=faq

http://hashcat.net/wiki/oclhashcat_plus

Hashkill 0.3.0:

http://www.gat3way.eu/forum/viewtopic.php?f=4&t=75

Oclhashcat no planea implementar office, rar, zip, pdf etc:

http://hashcat.net/wiki/feature_requests

Información sobre formato RAR y ZIP y GPU por Gat3way (autor de Hashkill):

http://hashcat.net/forum/thread-169-page-2.html

15.3 Bibliografía utilizada para Xen

Información variada sobre Xen:

http://old-list-archives.xen.org/archives/html/xen-devel/2009-08/msg01176.html

http://xen.1045712.n5.nabble.com/Patches-for-VGA-Passthrough-XEN-4-2-unstable-td4406265.html

http://www.linux-kvm.org/wiki/images/b/be/2011-forum-%24graphics-direct-assignment.pdf

http://www.mailinglistarchive.com/html/xen-devel@lists.xensource.com/2010-01/msg00787.html

http://old-list-archives.xen.org/archives/html/xen-devel/2010-03/msg01065.html

http://xen.org/files/xensummit_intel09/Graphics-Passthrough-with-VT-d.pdf

http://www.slideshare.net/xen_com_mgr/graphics-virtualization

http://xen.org/support/tutorial.html

http://en.wikipedia.org/wiki/PCI_configuration_space

http://article.gmane.org/gmane.comp.emulators.xen.devel/51194

http://www.xen.org/files/Marketing/HowDoesXenWork.pdf

http://article.gmane.org/gmane.comp.emulators.xen.devel/51194

http://es.scribd.com/doc/47729937/vga-passthrouth-xen

Extracción VGA BIOS:

http://latam.msi.com/forum/forum_posts.asp?TID=1562&PID=14182&title=inciar-dos-desde-un-usb-disk

http://www.sevenforums.com/tutorials/77347-nvidia-graphics-card-flash-bios.html

Wiki de Xen VGA-passthrough:

http://wiki.xen.org/xenwiki/XenVGAPassthrough

I/O de AMD:

http://developer.amd.com/documentation/articles/pages/892006101.aspx

Parches adaptados en este proyecto para la versión 4.2 de Xen, changeset-23350:

http://old-list-archives.xen.org/archives/html/xen-devel/2011-05/msg01426.html

Hilo de discusión sobre los parches:

http://xen.1045712.n5.nabble.com/Patches-for-VGA-Passthrough-XEN-4-2-unstable-td4406265.html

Mantenimiento posterior de los parches realizados en el proyecto:

http://www.davidgis.fr/blog/index.php?2011/12/07/860-xen-42unstable-patches-for-vga-pass-through

Blog David Techer (en francés):

http://www.davidgis.fr/blog/index.php?2011/08/28/818-xen-intel-vt-d-sur-core-5-2400-vga-passthrough-sur-carte-nvidia-msi-gt-440-partie-iv

http://www.davidgis.fr/blog/index.php?2011/10/17/837-xen-intel-vt-d-sur-core-5-2400-vga-passthrough-sur-carte-nvidia-evga-gtx-460-se-1024