ESCUELA DE DOCTORADO
INTERNACIONAL DE LA USC

Xosé
Fernández Fuentes

Tesis doctoral

Técnicas de análisis forense para la evaluación de la privacidad e integridad de la información: navegadores web y ataques de ransomware

Santiago de Compostela, 2023

Programa de doctorado en Investigación en Tecnologías de la Información

TESIS DE DOCTORADO

TÉCNICAS DE ANÁLISIS FORENSE PARA LA EVALUACIÓN DE LA PRIVACIDAD E INTEGRIDAD DE LA INFORMACIÓN: NAVEGADORES WEB Y ATAQUES DE RANSOMWARE

Xosé Fernández Fuentes

ESCUELA DE DOCTORADO INTERNACIONAL DE LA UNIVERSIDAD DE SANTIAGO DE COMPOSTELA

PROGRAMA DE DOCTORADO EN INVESTIGACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN

SANTIAGO DE COMPOSTELA
2023

Declaración del autor de la tesis

Técnicas de análisis forense para la evaluación de la privacidad e integridad de la información: navegadores web y ataques de ransomware

Don Xosé Fernández Fuentes

Presento mi tesis, siguiendo el procedimiento adecuado al Reglamento, y declaro que:

1. *La tesis abarca los resultados de la elaboración de mi trabajo.*
2. *De ser el caso, en la tesis se hace referencia a las colaboraciones que tuvo este trabajo.*
3. *Confirmo que la tesis no incurre en ningún tipo de plagio de otros autores ni de trabajos presentados por mí para la obtención de otros títulos.*
4. *La tesis es la versión definitiva presentada para su defensa y coincide la versión impresa con la presentada en formato electrónico.*

Y me comprometo a presentar el Compromiso Documental de Supervisión en caso de que el original no esté en la Escuela.

En Santiago de Compostela, 20 de mayo de 2023

Fdo. Xosé Fernández Fuentes

Autorización del Director/Tutor de la Tesis

Técnicas de análisis forense para la evaluación de la privacidad e integridad de la información: navegadores web y ataques de ransomware

Don José Carlos Cabaleiro Domínguez, Catedrático de Universidad del Área de Arquitectura y Tecnología de Computadores de la Universidade de Santiago de Compostela.

Don Tomás Fernández Pena, Catedrático de Universidad del Área de Arquitectura y Tecnología de Computadores de la Universidade de Santiago de Compostela.

INFORMAN:

Que la presente tesis, se corresponde con el trabajo realizado por ***Don Xosé Fernández Fuentes****, bajo nuestra dirección/tutorización, y autorizamos su presentación, considerando que reúne los requisitos exigidos en el Reglamento de Estudios de Doctorado de la USC, y que como directores/tutores de esta no incurre en las causas de abstención establecidas en la Ley 40/2015.*
De acuerdo con lo indicado en el Reglamento de Estudios de Doctorado, declara también que la presente tesis doctoral es idónea para ser defendida en base a la modalidad de Monográfica con reproducción de publicaciones, en los que la participación del doctorando fue decisiva para su elaboración y las publicaciones se ajustan al Plan de Investigación.

En Santiago de Compostela, 20 de mayo de 2023

Fdo. José Carlos Cabaleiro Domínguez
Director tesis

Fdo. Tomás Fernández Pena
Director tesis

A J y MJ

Llegar tan lejos no hubiera sido posible sin todo su apoyo.

First, you have to finish.

Michael Schumacher

Agradecimientos

Gracias a mis directores, Tomás y Caba, por darme la oportunidad de investigar en un campo de la informática que realmente me apasiona. Su asesoramiento y dedicación me han ayudado a hacer este sueño realidad.

20 de mayo de 2023

Agradecimientos adicionales

Este trabajo ha recibido apoyo financiero de la Consellería de Cultura, Educación y Universidad de la Xunta de Galicia (acreditación 2019-2022 ED431G-2019/04, referencia del grupo competitivo 2022-2024, ED431C 2022/16) y el Fondo Europeo de Desarrollo Regional (FEDER), que reconoce al Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS) de la Universidad de Santiago de Compostela como un Centro de Investigación del Sistema Universitaria Gallego. Este trabajo también ha recibido el apoyo del Ministerio de Economía y Competitividad del Gobierno de España (subvención n.º PID2019-104834 GB-I00).

Índice general

Resumen

Año tras año la tecnología invade más áreas de nuestra vida. Hace tan solo un par de décadas, el ciudadano medio no tenía que prestar especial atención a la privacidad en su día a día. Simplemente no era tecnológicamente posible que su nevera enviara al fabricante qué comida había dentro ni que su aspiradora enviara un plano de su vivienda a una empresa en el extranjero. Desde el punto de vista de la seguridad, el ciudadano medio "solo" tenía que protegerse de posibles amenazas directas y cercanas, como robos. No era necesario instalar una puerta blindada para evitar que una agencia de inteligencia instalara micrófonos en su casa para escuchar las conversaciones.

A día de hoy la situación es totalmente distinta. Una gran cantidad de dispositivos invaden nuestra privacidad recopilando información sobre nuestros movimientos, nuestra salud, nuestras amistades, nuestros viajes, nuestras conversaciones... Y lo peor es que esta información no está bajo nuestro control, sino que es almacenada y procesada por las empresas que fabrican dichos dispositivos. Y ya no solo hay que preocuparse por quien recopila nuestros datos, sino que también tenemos que prestar atención a proteger debidamente los datos que realmente nos interesan para evitar que caigan en malos manos, sean filtrados o que los perdamos para siempre. Ahora no es suficiente con protegerse de que un desconocido descubra el código de la alarma de nuestra casa, necesitamos tener una visión un poco más amplia y tener en cuenta nuevos frentes, ya que el ataque puede provenir desde cualquier punto del mundo.

Una de las herramientas básicas de nuestra vida cotidiana es el uso de un navegador web. Gracias a la cantidad de servicios en línea, actualmente es posible hacer prácticamente cualquier tarea, ya sea del ámbito personal o profesional, con solo un navegador web. Con él podemos comprobar el correo electrónico, hacer llamadas y videollamadas, leer las noticias, jugar a videojuegos, organizar álbumes de fotos, escu-

char *podcasts* y un largo etcétera. Es una herramienta que empleamos para manejar una gran cantidad de información personal o sensible. Debido a esto, los desarrolladores fueron añadiendo nuevas medidas de seguridad y privacidad al código de los navegadores. Una de estas medidas fue la introducción del modo privado. Cuando este modo está activado, datos como el historial de navegación, las contraseñas, las *cookies* o la caché no son almacenados en el dispositivo. Es una característica muy útil a la hora de aislar determinadas actividades en línea y especialmente conveniente cuando empleamos un ordenador compartido. Con el fin de verificar que este modo se comporta tal y como promete, en la primera parte de este trabajo se presenta una metodología con la que poner a prueba los navegadores en un amplio rango de situaciones, haciendo uso de técnicas de análisis forense. Esta metodología es completamente independiente del sistema operativo y de los navegadores utilizados y tiene por objetivo descubrir posible fugas de información que expongan las actividades realizadas con este modo de navegación.

La primera aplicación de la metodología se realizó con los conocidos navegadores Mozilla Firefox y Google Chrome ejecutándose en modo privado sobre cuatro entornos Linux diferentes. Tras analizar la información escrita en disco así como la información disponible en memoria, se observó que ninguno de los dos navegadores almacenaron información relacionada con la navegación en el disco duro. Sin embargo, el análisis de la memoria reveló que era posible recuperar una gran cantidad de información dependiendo del entorno utilizado. Por ejemplo, en el caso en que los navegadores se ejecutaron en una máquina virtual de VMware, fue posible recuperar la mayoría de las acciones realizadas, desde las palabras clave introducidas en un campo de búsqueda hasta el nombre de usuario y la contraseña introducidos para iniciar sesión en un sitio web, incluso después de reiniciar el ordenador. En cambio, cuando se ejecutó Mozilla Firefox en un Linux no virtualizado y ligeramente fortificado, no fue posible recuperar ningún artefacto relacionado con la navegación después de cerrar el navegador.

A pesar de que los sistemas de escritorio siguen representando una gran cantidad del tráfico de Internet generado, ya no son los mayoritarios. A día de hoy, más del 50 % del tráfico es generado por dispositivos móviles. Estos se han convertido en dispositivos esenciales en nuestra vida cotidiana. No se separan de nosotros en todo el día y llevamos en ellos nuestros datos más preciados e importantes, desde los vídeos familiares de los últimos años hasta la información de la tarjeta de crédito para poder

pagar con el teléfono. Al ser uno de los dispositivos con los que más tiempo pasamos a lo largo del día, no es de extrañar que cada vez exijamos un mayor nivel de privacidad a cualquier aplicación o dispositivo con el que gestionamos nuestra información más privada. Al igual que en los sistemas de escritorio, las versiones de los navegadores para dispositivos móviles también incorporan el modo privado. El problema es que confiamos en que se van a comportar de forma idéntica a como funcionan las versiones de escritorio sin realmente comprobarlo. Y lo peor es que los estudios publicados analizando su comportamiento son realmente pocos y, además, la mayoría de ellos realizan las pruebas utilizando diversos emuladores o máquinas virtuales, lo que no representa un entorno de pruebas real. Por ello, decidimos que el segundo caso de uso para la metodología presentada sería con varios navegadores, concretamente Google Chrome, Brave, Mozilla Firefox y Tor Browser, ejecutándose en una tableta con Android 13 y en dos dispositivos virtuales creados con Android Emulator. Los resultados confirman que estos navegadores no escribieron ninguna información sobre la navegación realizada en modo privado en el sistema de archivos. Sin embargo, el análisis de la memoria volátil permitió recuperar el nombre de usuario y la contraseña utilizados para iniciar sesión en un sitio web o las palabras clave introducidas en un motor de búsqueda, incluso después de reiniciar los dispositivos.

El modo privado nos permite reducir la cantidad de información que queda guardada en un dispositivo tras haber navegado por la web. El problema es que, a medida que pasan los años, somos más y más dependientes de una conexión a Internet, lo que hace que estemos expuestos a un mayor número de amenazas. Muchas de estas amenazas no existían hace unos pocos años y, por lo tanto, no estamos preparados para defendernos adecuadamente.

Como es obvio, no utilizamos los ordenadores y los dispositivos móviles exclusivamente para navegar, sino que también los usamos para gestionar una gran cantidad de documentos, tanto de nuestra vida privada como laboral, durante nuestro día a día. Los ciberdelincuentes se han dado cuenta del valor de la información que almacenamos en los dispositivos y están decididos a ir a por ella. A muchos delincuentes no les importará el contenido de los documentos, pero saben que nos pueden hacer daño si nos impiden el acceso a ellos o, aún peor, nos amenazan con perderlos para siempre. Esta amenaza se ha convertido en la peor pesadilla de los últimos años, tanto para grandes empresas como para particulares, y es conocida como *ransomware*. El

ransomware es un tipo de *malware* que bloquea el acceso a los archivos de un dispositivo, normalmente cifrándolos con una clave que solo conoce el atacante, hasta que la víctima pague un rescate. Cuando se paga el rescate, el atacante entrega la clave para recuperar el acceso a los archivos, siendo necesario confiar en que realmente el atacante está en posesión de dicha clave, ya que no hay forma de estar completamente seguro de que el atacante sea capaz de descifrar todos los archivos. El problema es que hay ocasiones en las que el atacante no tiene la clave de descifrado, por lo que pagar el rescate es totalmente inútil. Además, esta amenaza continúa evolucionando, siendo capaz a día de hoy de expandirse rápidamente por la red e infectar otros equipos. Su método de infección inicial favorito son el correo electrónico y los sitios web de descarga de programas. Sin embargo, ya existen *ransomware* más avanzados que no requieren de la intervención del usuario y se aprovechan de diferentes vulnerabilidades para conseguir propagarse. Durante el transcurso de esta investigación se encontraron algunas de estas situaciones, que se mostrarán y explicarán en detalle.

Como este tipo de *malware* se ha hecho tan popular, ahora es posible encontrar en Internet repositorios de código fuente con *ransomware* preparados para usar. Muchos de ellos se crean con fines educativos, con la idea de que otros investigadores ayuden a crear nuevas medidas contra estas amenazas. El problema es que, al tratarse de un *malware* tan lucrativo, atrae el interés de todo tipo de malos actores, desde grupos altamente cualificados hasta simples *script-kiddies*. Por lo tanto, el objetivo de la segunda parte de este trabajo es ayudar a recuperarse de un ataque de *ransomware* utilizando técnicas de análisis forense. Para ello, se analizaron varios *ransomware* populares disponibles en el repositorio de código GitHub con el fin de mostrar cómo es posible, en muchas ocasiones, recuperar la clave de cifrado sin pagar el rescate. En concreto, los *ransomware* seleccionados están escritos en Python y pueden ser fácilmente utilizados para realizar ataques multiplataforma.

Lo más preocupante es que estos *ransomware* disponibles en plataformas como GitHub no son detectados por la mayoría de las soluciones antivirus. De los *ransomware* analizados, el *ransomware* RAASNet fue el que más positivos dio en VirusTotal, siendo la muestra para Windows detectada por el 26 % de los antivirus y, la muestra para Linux, por el 8 %. Y solo fue el más detectado por el hecho de que parte de su código fue utilizado en una campaña real de *ransomware* en el año 2019 y en el año 2021.

En el transcurso de esta investigación se desarrollaron dos herramientas que permiten recuperar rápidamente las claves de cifrado utilizadas por varios *ransomware* a partir de un volcado de memoria, independientemente del sistema operativo infectado. Estas dos herramientas pueden ser útiles contra otros *ransomware* similares, ya que muchos de ellos utilizan las mismas librerías para realizar el cifrado de archivos y para gestionar las claves.

Introducción

Toda persona y toda organización tiene información que necesita mantener a salvo. Esta información pueden ser historiales médicos, información bancaria o diseños para futuros productos. El objetivo es impedir que la información caiga en las manos equivocadas, ya que si eso ocurriera, las consecuencias podrían ser desastrosas. Antes de la era digital, toda la información era almacenada de forma física. Esto hacía que asegurar esa información fuera relativamente sencillo. "Todo" lo que había que hacer era controlar quien tenía acceso físico al lugar donde se almacenaba dicha información. No era necesario preocuparse de que alguien en la otra punta del mundo pudiera acceder de forma remota. Simplemente no era posible. A día de hoy, una gran cantidad de información se almacena digitalmente, por lo que el número de frentes que hay que proteger es muy distinto y mucho mayor. Esto hace que haya una creciente necesidad de mejorar la seguridad informática y, en general, la seguridad de la información, ya que es imprescindible disponer de mecanismos y políticas con las que fortificar los múltiples dispositivos en los que esta se almacena.

Todo comenzó en la década de los 60. Las empresas empezaban a utilizar *mainframes* para el procesamiento de información y querían evitar que cualquier persona, con unos mínimos conocimientos de cómo funcionaba un ordenador, pudiera acceder a la información almacenada [141]. Para ello, la principal medida de seguridad implementada era el control físico, estableciendo medidas para impedir el acceso a las salas donde se encontraban los *mainframes*. También se empezó a implantar el uso de contraseñas para evitar accesos no autorizados. Por supuesto, a principios de los 60 no era necesario preocuparse por ataques provenientes de la red.

El primer sistema operativo en integrar funciones de seguridad fue MULTICS (Multiplexed Information and Computing Service) [242], un sistema destinado a *mainfra-*

mes que fue desarrollado a mediados de los años 60 por el científico Fernando J. Corbató del MIT junto con las empresas Bell Labs y General Electric. Aunque este sistema operativo está totalmente obsoleto a día de hoy, fue en él donde empezó la investigación en el campo de la seguridad informática.

A mediados de la década de los 60 apareció la necesidad de abordar tareas más complejas y sofisticadas, lo que implicó buscar nuevas formas en la que los *mainframes* pudieran intercambiar información [242]. Definitivamente el método utilizado hasta el momento no estaba a la altura: el envío por correo de cintas magnéticas de un centro de computación a otro. Afortunadamente, en 1968, el Dr. Larry Roberts desarrolló la solución al problema: ARPANET, un sistema de comunicación en red redundante que permitía el intercambio de información entre diferentes instituciones.

En 1969, cuando parecía evidente la necesidad de incorporar medidas de seguridad en los sistemas operativos, nació UNIX sin incluir absolutamente ninguna [242]. UNIX fue creado por varios desarrolladores de MULTICS, los cuales consideraron que, como la principal tarea de UNIX era el procesamiento de texto, no era necesario tener en cuenta la seguridad. Es más, UNIX no incorporó la posibilidad de utilizar contraseñas hasta varios años después.

El 29 de octubre de 1969, el profesor de la UCLA (University of California, Los Angeles) Leonard Kleinrock junto con su estudiante Charley Kline enviaron el primer mensaje a través de ARPANET, utilizando un ordenador SDS Sigma 7 [96]. El destinatario del mensaje era Bill Duvall, un programador de Stanford Research Institute. Curiosamente, el mensaje que tenían intención de enviar desde la UCLA era la palabra `login`, pero el equipo se quedó totalmente bloqueado tras pulsar la tecla 'o'. Por lo tanto, el primer mensaje enviado por "Internet" fue `lo`. Desde entonces, existe la creencia de que el mensaje que intentaban enviar era "lo and behold", que en español podría traducirse por "he aquí", "oh sorpresa" o "mira por dónde".

A principios de la década de los 70 apareció el primer virus informático [96]. Robert Thomas, un investigador en la empresa BBN Technologies en Cambridge (Massachusetts), creó un programa llamado Creeper que se movía entre terminales Tenex utilizando ARPANET. A su paso dejaba un pequeño rastro e imprimía el mensaje "I'M THE CREEPER: CATCH ME IF YOU CAN". A Ray Tomlinson, un programador que también trabajaba en BBN Technologies y que fue el inventor del correo electrónico, le gustó la idea de Robert Thomas y decidió crear el "antídoto". Creó el

primer antivirus, al que llamó Reaper, el cual se dedicaba a buscar copias de Creeper y eliminarlas de los equipos.

A pesar de que Creeper y Reaper fueron creados por pura curiosidad y sin ninguna intención maliciosa, estos revelaron una serie de fallos de seguridad en ARPANET, lo que preocupó a muchas organizaciones y gobiernos que estaban conectando sus ordenadores a la red. Al mismo tiempo, empezaron a surgir grupos de personas que se interesaban en buscar formas de infiltrarse en dichas redes y robar información [141].

En 1970 se publicó el que ahora es considerado el trabajo que inició el campo de la seguridad informática [242]. Este trabajo, identificado como RAND Report R-609, estuvo clasificado durante nueve años y fue el primero en destacar que la seguridad iba más allá de la seguridad física de los dispositivos. RAND Report R-609 remarcó la importancia de proteger los datos, de implantar políticas de acceso y de realizar una gestión de roles apropiada.

También a principios de los 70, Robert M. Metcalfe (uno de los creadores de Ethernet) destacó los innumerables problemas fundamentales de seguridad que tenía ARPANET [242]. Por ejemplo, carecía de medidas que protegieran la información de usuarios no autorizados, faltaban procedimientos de seguridad para el establecimiento de las conexiones telefónicas y no existía ningún tipo de identificación de usuarios.

En 1973 nació el término de Internet tras la conexión de la University College of London (Inglaterra) y la Royal Radar Establishment (Noruega) a ARPANET [70]. Al año siguiente (1974) nació el primer proveedor de servicios de Internet (ISP por sus siglas en inglés) con el nombre de Telenet, el cual era una versión comercial de ARPANET. También en el año 1974, Vinton Cerf y Bob Kahn (considerados por muchos los padres de Internet) publicaron los detalles de diseño del protocolo TCP. Y, en el año 1976, la reina Isabel II pulsó el botón de "enviar" en su primer correo electrónico.

A finales de los 70, llegaron los primeros microprocesadores y con ellos una revolución que poca gente se podía imaginar [242]. El uso de los *mainframes* quedó superado por la expansión de los ordenadores personales, llevando a una descentralización de los datos. Esto hizo que fuera necesario expandir las redes de comunicaciones, lo que condujo al desarrollo de nuevos protocolos.

A principios de los 80, se implantó la suite de protocolos TCP/IP como la base de ARPANET y nació el protocolo DNS [242]. También por esta época, la necesidad

de preocuparse por la seguridad de los sistemas empezaba a ser más que evidente en algunos ámbitos. Por ejemplo, el gobierno de los Estados Unidos formalizó el reconocimiento de la seguridad informática como una cuestión crítica para los sistemas de información federales.

En 1988, Robert Morris creó el primer gusano famoso de la historia. Conocido posteriormente como gusano Morris, este programa tenía la capacidad de extenderse por la red, copiándose a sí mismo en los equipos por los que pasaba [141]. Para ello, explotaba varias vulnerabilidades en los programas sendmail, finger y rsh/rexec. Además, también incluía una lista de 900 contraseñas así como la capacidad de hacer fuerza bruta utilizando de referencia los nombres de usuario para conseguir acceder [137]. Cuando el gusano llegaba a una nueva máquina, verificaba si ya estaba infectada y, si lo estaba, la reinfectaba el 14 % de las veces. Esto hizo que adquiriera el comportamiento de una bomba *fork*. A medida que las máquinas eran infectadas más y más veces por el gusano, se ralentizaban hasta el punto de quedar totalmente bloqueadas. Esto hizo que el experimento de Morris pasara de ser un recto intelectual a un auténtico ataque de denegación de servicio. Es evidente que el nivel de replicación fue ligeramente mayor de lo esperado y llegó a infectar unas 6 000 máquinas UNIX, lo que representaba el 10 % de todo Internet [137]. La estimación de los daños causados osciló entre 100 000 y 10 000 000 de dólares, lo que llevó a que Robert Morris fuera el primero en ser acusado bajo la ley contra el fraude y el abuso informático (*Computer Fraud and Abuse Act* en inglés).

Este suceso marcó un punto de inflexión en la historia de la seguridad de la información. Aunque Robert Morris no tenía ninguna intención maliciosa, fue el primero en demostrar al mundo que no se debe asumir que todo el mundo vaya a actuar con buenas intenciones [137]. Es necesario disponer, como mínimo, de mecanismos robustos que controlen el acceso a los equipos. Este acontecimiento creó la base para la idea de Confianza Cero (o *Zero Trust* en inglés), donde todo usuario, ya sea interno o externo, tiene que ser autenticado, autorizado y validado para todo acceso que realice en una aplicación [165].

En 1989, Internet fue puesto a disposición del público después de décadas de ser exclusivo para uso gubernamental, académico y profesional [141, 242]. Internet se convirtió en una interconexión de millones de redes repartidas por prácticamente todo el mundo, naciendo innumerables nuevos usos para esta creación sin precedentes. Solo

había un pequeño problema: la falta de un organismo de estandarización. Esto hizo que, inicialmente, todas las conexiones se basaran en estándares de facto, las cuales no le daban prioridad ninguna a la seguridad. De esta época provienen muchas de las carencias en seguridad del correo electrónico, como la falta de autenticación de los servidores de correo o la falta de cifrado de los mensajes. La mentalidad de aquel momento seguía siendo mayoritariamente la de seguridad física. Afortunadamente, esto fue cambiando a medida que Internet siguió creciendo y, por supuesto, al nacimiento de nuevas amenazas que demostraron la necesidad de tomar en serio todos los aspectos relacionados con la seguridad. A principios de los 2000 nació la seguridad de la información como una disciplina propia y muchas empresas empezaron a integrar soluciones de seguridad, como los antivirus, cuya popularidad explotó en los años siguientes.

Año tras año, el tamaño de la "red de redes" sigue en aumento, conectando miles de millones de dispositivos entre ellos. Desde los innumerables dispositivos móviles vendidos en los últimos años, como teléfonos inteligentes y tabletas, hasta la multitud de dispositivos del Internet de las cosas (IoT por sus siglas en inglés) [242] que nacieron en los últimos años. Todo esto hace que la variedad de dispositivos que están conectados a la red sea enorme. Cada familia de dispositivos presenta una serie de amenazas totalmente distinta y no solo eso, sino que algunos de ellos carecen de cualquier protección frente a ataques, especialmente los de la familia IoT. Tampoco hay que olvidar la cantidad de ellos que ejecutan sistemas obsoletos o sistemas desactualizados, lo que hace que la situación actual sea aún peor.

El nivel de los ataques realizados por los ciberdelincuentes, al igual que el tamaño de Internet, se incrementa con cada año que pasa [242]. A principios del siglo XXI, era común ver los ataques conocidos como *defacements*. Estos consistían en cambiar el contenido de los sitios web y eran considerados como una forma de vandalismo digital, algunas veces motivados únicamente por diversión. A día de hoy, los ciberdelincuentes son auténticos profesionales muy bien preparados que buscan maximizar sus ganancias mediante el robo y la extorsión. Por ejemplo, en los años 2013 y 2014, un grupo de ciberdelincuentes consiguió entrar en Yahoo y tener acceso a toda la información personal de los tres mil millones de usuarios [141]. La empresa no informó del suceso a tiempo y pagó las consecuencias: una multa de 35 millones de dólares, un descenso de su valor en bolsa de 350 millones de dólares y la pérdida de la confianza de innumerables

usuarios. En 2015, los investigadores de seguridad Charlie Miller y Chris Valasek demostraron su capacidad para controlar por completo, y de forma remota, un Jeep Cherokee. Esto incluía deshabilitar los frenos, tocar la bocina, apretar los cinturones de seguridad o controlar la dirección del coche [85]. Otro ejemplo, en diciembre de 2019, la empresa británica The Heritage Company se vio obligada a cerrar, dejando sin empleo a 300 personas. El motivo: un ataque de *ransomware* [93]. Este tipo de sucesos hace que cada vez más individuos y más empresas se preocupen e inviertan en seguridad, ya que, en el mundo digitalizado en el que vivimos, los ataques pueden llegar a ser devastadores.

Teniendo en cuenta la importancia de la seguridad de la información, en este trabajo nos centramos en dos áreas concretas. La primera es en el análisis del nivel de privacidad ofrecido por la herramienta básica e imprescindible que utilizamos para navegar por Internet desde su invención: el navegador web. Y la segunda es el estudio del ataque más devastador de los últimos años: el *ransomware*. Ambos frentes estudiados desde la perspectiva del análisis forense digital.

Análisis forense digital

El análisis forense digital es una rama de la ciencia forense que se centra en la identificación, adquisición, procesamiento, análisis y elaboración de informes sobre datos almacenados electrónicamente [65].

Una de sus principales aplicaciones es intentar determinar cuál fue la cadena de acontecimientos que llevaron a una situación particular [110]. Por ejemplo, si una empresa detecta que uno de sus servidores ha sido comprometido o desconfía de que uno de sus empleados pueda estar filtrando información, puede recurrir a la investigación forense digital para intentar verificar si las sospechas son ciertas y, al mismo tiempo, identificar el origen del problema. Esto convierte a esta disciplina en uno de los componentes críticos del proceso de respuesta ante incidentes.

En este trabajo, se aplicaron diferentes técnicas de análisis forense digital con dos fines: 1) detectar fugas de información en el modo privado de los navegadores y 2) ayudar a recuperarse de un ataque de *ransomware*.

Modo de navegación privada

El número de actividades que realizamos utilizando Internet va en aumento día tras día. Solo hay que prestar atención a la cantidad de solicitudes, procedimientos, reclamaciones, compras, pagos, gestiones, etc. que antes eran realizadas de forma presencial o por vía telefónica y que ahora son hechas de forma totalmente telemática. En la gran mayoría de los casos, el elemento común que tienen todas estas actividades es que requieren del uso de un navegador web para poder realizarlas en línea.

A medida que la tecnología llega a más ámbitos de nuestras vidas, los usuarios exigimos un mayor control sobre los datos que exponemos en las distintas aplicaciones. Esto hace que diferentes servicios y productos incorporen nuevas funcionalidades destinadas a proteger la privacidad. Una de estas funcionalidades es el modo privado introducido en la mayoría de los navegadores web actuales. En el caso de Google Chrome, este modo se introdujo en el año 2008 con el nombre de "Modo incógnito". Al año siguiente, con el lanzamiento de la versión 3.5, Mozilla Firefox también incorporó esta funcionalidad con el nombre de "Navegación privada". Este nuevo modo de navegación tiene como objetivo evitar que cualquier información relacionada con la navegación se guarde en el dispositivo que se esté utilizando. Además, algunos navegadores activan funciones adicionales cuando se utiliza este modo, como el bloqueo por defecto de rastreadores conocidos o la activación de la cabecera "No rastrear" (*Do Not Track* en inglés).

Esta característica parece haber sido bien recibida por los usuarios, ya que según un estudio publicado en 2018 [90], cada vez son más los usuarios que hacen uso del modo privado, y no solo por motivos de privacidad, sino también de seguridad y prácticos. El problema es que a menudo se sobrevalora la protección ofrecida por este modo de navegación.

Ransomware

El *ransomware* es un tipo de *malware* que impide el acceso a los archivos del dispositivo, normalmente cifrándolos con una clave que solo conoce el atacante, hasta que la víctima paga un rescate. Este tipo de ataques causa grandes pérdidas económicas a las víctimas, que pueden ser tanto particulares como grandes empresas. En los últimos

años, el número de ataques ha crecido exponencialmente, surgiendo un gran número de familias de *ransomware* diferentes.

De forma genérica, los pasos que efectúa un *ransomware cifrador* cuando se ejecuta son los siguientes: genera una clave con la que cifrará los archivos, se comunica con el servidor de Mando y Control (en inglés, *Command & Control* o, abreviadamente, C&C), escanea el sistema de ficheros en busca de archivos que tengan una extensión en particular y que no se encuentren dentro de ninguna de las carpetas que no quiere alterar, cifra los archivos (ya sea a medida que los encuentra o después de haber escaneado todo el sistema de ficheros) y, por último, muestra una nota de rescate (a menudo junto con un temporizador).

Debido a su popularidad, rentabilidad y a que no son particularmente complicados de programar es posible encontrar en Internet repositorios de código fuente de *ransomware* listos para usar. Aunque muchos de ellos se crean con fines educativos, no todos acaban siendo utilizados con dicho propósito y algunos terminan siendo empleados en campañas reales.

Objetivos

El objetivo principal de esta tesis es proponer nuevas metodologías y soluciones a problemas existentes en el mundo de la seguridad informática mediante el uso de técnicas de análisis forense digital. Concretamente, los esfuerzos están centrados en el ámbito de la privacidad digital y en el ámbito de la criptovirología.

Los objetivos específicos a alcanzar son los siguientes:

- Diseñar una metodología para verificar el correcto funcionamiento del modo de navegación privada incluido en los navegadores web actuales. Para ello, la metodología tiene que detectar cualquier fuga de información sensible, teniendo en consideración los múltiples entornos y condiciones en los que se puede ejecutar un navegador web. Además, la metodología tiene que ser independiente del sistema operativo y del navegador empleado, permitiendo aplicarla tanto a sistemas de escritorio como a dispositivos móviles.

- Desarrollar nuevas herramientas que permitan revertir los daños producidos por un ataque de *ransomware*. Estas harán uso de técnicas de análisis forense con

el fin de recuperar la clave de cifrado empleada para bloquear el acceso a los ficheros.

Esquema de la tesis

El resto del documento está estructurado del siguiente modo:

- En el Capítulo 1 se introduce brevemente el análisis forense digital.
- En el Capítulo 2 se describe la metodología diseñada para probar el modo privado de los navegadores.
- En el Capítulo 3 se muestra la aplicación de la metodología a sistemas operativos Linux y Android.
- En el Capítulo 4 se relata la historia del *malware* de rescate desde su nacimiento hasta el día actual.
- En el Capítulo 5 se presentan las dos herramientas desarrolladas para revertir un ataque de *ransomware*.
- En el Capítulo 6 se presentan las conclusiones así como el posible trabajo futuro.

Capítulo 1

Introducción al análisis forense digital

El análisis forense digital es una disciplina relativamente nueva y se aplica tanto para la investigación de delitos "tradicionales" como para los propiamente relacionados con las tecnologías de la información y la comunicación [64].

En las siguientes secciones se describe el nacimiento de esta disciplina, las fases que la forman y algunas de las opciones hardware y software disponibles actualmente para la realización de un análisis forense.

1.1. Nacimiento

Las leyes y las técnicas policiales existentes hasta mediados de los ochenta no contemplaban ningún delito informático, por lo que las fuerzas del orden no perseguían a este tipo de delincuentes. Sin embargo, a medida que se expandió el uso de la informática, las fuerzas de seguridad empezaron a prestarle más atención. Por ejemplo, organismos como el FBI empezaron a incorporar unidades dedicadas a las investigaciones digitales y forenses [110].

Hubo dos acontecimientos en particular que permitieron dar a conocer la necesidad de investigaciones forenses. El primero fue en el Laboratorio Nacional Lawrence Berkeley (LBL). Clifford Stoll, un astrónomo convertido a administrador de sistemas en LBL, detectó una intrusión en sus sistemas. Tras muchos esfuerzos y mucho tiem-

po, consiguió elaborar un plan para mantener al ciberdelincuente conectado el tiempo suficiente para rastrear el origen de la conexión. Gracias al apoyo de las autoridades, consiguieron detenerlo y procesarlo por espionaje [110]. Este ciberdelincuente era Markus Hess, un alemán que se dedicaba a entrar en redes militares e industriales de prácticamente todo el mundo para luego vender la información a la KGB, la agencia de inteligencia de la Unión Soviética. Clifford Stoll recogió toda la historia en su libro titulado "The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage" [201].

El segundo acontecimiento fue el daño ocasionado por el Gusano Morris en el año 1988 mencionado en la introducción. Este gusano dejó inoperativos un buen porcentaje de los equipos conectados a la red por aquel entonces, causando daños por valor de más de 100 000 dólares. Este incidente llevó a la creación del Centro de Coordinación de Equipos de Incidentes Informáticos (Computer Emergency Response Team Coordination Center o CERT/CC, en inglés) de Carnegie Mellon [110].

1.2. Descripción del proceso

La definición proporcionada por EC-Council establece que "el análisis forense digital es una rama de la ciencia forense que se centra en la recuperación e investigación de material encontrado en dispositivos digitales relacionados con el delito cibernético" [237]. Toda información transmitida o almacenada en forma binaria consiste en una prueba digital [97]. Puede ser prácticamente cualquier cosa, desde el historial de navegación hasta los documentos almacenados en un disco externo o las imágenes guardadas en un teléfono móvil.

Existen varios esquemas que definen cómo se debe realizar un análisis forense digital y, en su mayor parte, suelen seguir unas pautas muy similares. Uno de los más utilizados es el marco de investigación digital del Digital Forensics Research Workshop (DFRWS), el cual está formado por seis etapas: identificación, preservación, recopilación, inspección, análisis y documentación. En las siguientes subsecciones se incluye una descripción del propósito de cada una de las fases mencionadas así como las tareas a realizar.

1.2.1. Identificación

La primera fase consiste en examinar el entorno, tanto físico como digital, en busca de posibles fuentes de pruebas. Los técnicos forenses registran cualquier dispositivo que pueda ser relevante para la investigación. A veces esto puede ser un reto, ya que puede haber dispositivos digitales que pasen totalmente desapercibidos. Por ejemplo, puede parecer que una consola de juegos, como una Xbox, es totalmente irrelevante para el caso en cuestión. Sin embargo, la consola pudo haber sido modificada para ejecutar un sistema operativo completamente diferente y utilizada cualquier otra actividad diferente de jugar [38]. También es realmente importante prestar atención a posibles objetos aparentemente normales, ya que estos pueden haber sido también alterados. Por ejemplo, un objeto con la apariencia de una moneda puede haber sido hecho a medida para alojar en su interior una tarjeta de memoria.

Un principio que es de gran relevancia en esta fase es el principio de intercambio de Locard [110]. Este principio básico, formulado por el Dr. Edmond Locard, un pionero de la ciencia forense, consiste en lo siguiente: "Todo contacto deja un rastro". Aplicando este principio al mundo digital, esto quiere decir que, cada vez que dos sistemas se comunican entre ellos, quedan registros en ambos sistemas sobre la conexión establecida. Por ejemplo, si un equipo se conecta a un servidor para descargar una página web, es muy probable que queden registros en ambos extremos. En el ordenador es posible que quede almacenada una *cookie*, una entrada en el historial de navegación o elementos en la caché del sitio visitado. En el servidor existe la posibilidad de que se cree una entrada en el fichero de *log* con los detalles sobre la conexión, incluyendo la IP desde donde se inició la conexión, el momento en el que se produjo y las páginas descargadas. Por supuesto, esta información puede ser temporal y dependerá de las habilidades del analista, así como de las herramientas disponibles, para conseguir recuperar dicha información. También es importante destacar que toda evidencia encontrada puede haber sido manipulada, por lo que se debe evitar confiar en una única prueba si no hay otras evidencias que la respalden.

1.2.2. Preservación

Una vez identificados los dispositivos de posible relevancia es imprescindible mantener la integridad de los datos almacenados en ellos, evitando que se produzca cualquier

tipo de modificación o eliminación [110]. En el caso de equipos, como ordenadores de sobremesa, será necesario aislarlos de la red, ya sea de forma física o mediante el establecimiento de controles de acceso a la red, y evitar que cualquier persona tenga acceso a ellos. En el caso de máquinas virtuales, se deberá crear una instantánea (*snapshot* en inglés) y almacenarla en un dispositivo no volátil.

1.2.3. Recopilación

La tercera fase consiste en adquirir las pruebas digitales teniendo en cuenta la naturaleza volátil de algunas de las mismas. Por ejemplo, si se apaga un ordenador se pueden perder todos los datos que estaban almacenados en memoria, lo que puede incluir información realmente valiosa, como conexiones activas y procesos en ejecución. Por consiguiente, es imprescindible tener en cuenta esta volatilidad en el momento de empezar el proceso de recogida de las pruebas. A modo de guía, el IETF (Internet Engineering Task Force) desarrolló un documento [34] que aborda el orden de volatilidad de las evidencias digitales. A continuación se incluye un ejemplo del orden de volatilidad para un sistema típico, desde el más volátil al menos volátil:

1. Registros y caché.
2. Tabla de procesos, caché ARP, tabla de enrutamiento y memoria.
3. Archivos temporales.
4. Disco duro.
5. Registros almacenados remotamente que puedan tener información sobre el sistema en cuestión.
6. Configuración física y topología de la red.
7. Información archivada.

A la hora de manipular toda evidencia es imprescindible hacerlo de la manera correcta, ya que una adquisición incorrecta lleva a que dicha prueba sea totalmente inválida, por muy relevante que pueda ser [110].

El otro punto de gran relevancia es documentar toda acción llevada a cabo, ya sea tomando notas, creando diagramas o sacando fotografías. De esta forma es posible

justificar la integridad de las pruebas si llega a ser necesario. A modo de curiosidad, una frase que suelen utilizar los agentes de las fuerzas del orden es "si no lo escribiste, no sucedió".

1.2.4. Inspección

La cuarta fase detalla las herramientas específicas y las técnicas forenses que van a ser utilizadas para procesar las evidencias. Deben utilizarse programas que permitan extraer información, tanto de manera automatizada como manual, de las pruebas incautadas con el objetivo de reconstruir los sucesos anteriores [97]. Por ejemplo, si se deben procesar las evidencias obtenidas de un equipo que se sospecha que fue infectado por *malware*, en esta fase habrá que procesar el volcado de memoria con el fin de extraer los procesos que estaban en ejecución así como las conexiones que había establecidas. Asimismo, habrá que revisar los registros del sistema en busca de pistas así como examinar el disco duro para determinar si hay algún fichero sospechoso. Por supuesto, también se deberán intentar recuperar archivos borrados, restaurar ficheros que hayan sido manipulados y revisar el contenido de ficheros ocultos.

Al igual que en la fase anterior, será necesario prestar atención a la preservación adecuada de las pruebas, evitando contaminar las evidencias originales durante el proceso de inspección.

1.2.5. Análisis y documentación

La fase de análisis trata sobre el estudio de toda la información extraída anteriormente para determinar su pertinencia a la investigación [97]. El objetivo de esta fase es reconstruir los sucesos que llevaron a esa situación y validar las posibles hipótesis planteadas en el inicio de la investigación. Por ejemplo, volviendo al ejemplo del ordenador infectado por *malware*, el analista puede encontrar en el análisis del volcado de la memoria una conexión abierta a una IP sospechosa. Entonces, debe validar la existencia de esa conexión con la información obtenida de una captura de red. Esto le permitirá corroborar la información obtenida del volcado de memoria así como una idea de qué datos fueron transmitidos a dicha IP. Luego deberá intentar determinar si esa IP es conocida por ser utilizada por algún software malicioso. Esto le ayudará a

determinar si realmente había algún proceso extraño corriendo en el equipo y validar la hipótesis de si el ordenador estaba infectado por *malware*.

La última fase, documentación, consiste en elaborar un informe detallado que describa todas las acciones realizadas durante la investigación así como toda la información extraída [110]. Este informe debe ser exhaustivo y estar libre de opiniones.

1.3. Hardware

Durante el desarrollo de una investigación forense es necesario prestar mucha atención a la posible contaminación de las evidencias obtenidas. Esto empieza por una correcta configuración de los equipos a utilizar. Se debe disponer de al menos dos equipos. El primero será utilizado de manera exclusiva para la realización de todas las tareas de análisis y extracción de información. Además, carecerá de conexión a Internet con el fin de reducir las posibilidades de que sea manipulado, intentando así asegurar la integridad de las pruebas durante el análisis. El segundo equipo sí dispondrá de conexión a Internet y será el utilizado para todas las tareas restantes, como investigaciones adicionales en línea o redacción de informes [110].

Todo disco duro que contenga evidencias no puede ser conectado directamente a ningún equipo. Si se hiciera, se estarían alterando los datos, lo que invalidaría las pruebas. Por ejemplo, con el simple hecho de conectarlo, el sistema operativo podría intentar corregir el sistema de ficheros si encontrara alguna inconsistencia, o la simple apertura de un fichero modificaría la fecha y hora de último acceso. Por lo tanto, es imprescindible disponer de un bloqueador físico de escritura (*physical write blocker* en inglés). Este dispositivo se conecta entre el ordenador y el disco duro a examinar y evita que se produzca cualquier escritura en él. De esta forma, las evidencias pueden ser analizadas sin necesidad de preocuparse por dañar la integridad de las pruebas. En la Figura 1.1 se puede ver un bloqueador físico de escritura para discos SATA.

1.4. Software

Hay un grupo de programas básicos que todo laboratorio forense debe tener. Este grupo debe incluir herramientas para crear imágenes de discos, examinar las imágenes creadas, analizar volcados de memoria y generar informes [110].

Figura 1.1: Bloqueador físico de escritura. Este dispositivo impide cualquier intento de modificación en el disco duro conectado. Fuente: Wikimedia Commons

A día de hoy existen aplicaciones comerciales diseñadas exclusivamente para la ejecución de tareas forenses. Suelen ser muy utilizadas por las fuerzas del orden así como por la industria privada. Probablemente las tres más populares sean OpenText EnCase Forensic [145] (de las más completas, asistiendo en la realización de prácticamente todas las tareas a realizar durante el análisis), FTK Forensic Toolkit [77] (igual de completa que EnCase Forensic y también muy utilizada) y X-Ways [248] (carece de algunas funcionalidades, pero su licencia de uso es menos costosa).

La principal alternativa de código abierto es Autopsy [18]. Esta plataforma se caracteriza por su facilidad de uso y por incluir, entre otras, las herramientas de The Sleuth Kit (TSK) [211]. Debido a que ofrece funcionalidades que solo suelen estar disponibles en plataformas comerciales, es muy utilizada tanto por las fuerzas del orden como por empresas privadas e investigadores. Algunas de las funcionalidades destaca-

das son el análisis del registro de Windows, la recuperación de correos electrónicos, la detección de archivos maliciosos conocidos, la extracción de artefactos web, el análisis de líneas temporales, la obtención de una lista de todos los dispositivos USB conectados alguna vez al sistema, la recuperación de archivos eliminados y la búsqueda por palabras clave.

También existen diversas distribuciones Linux creadas con el fin de ofrecer un entorno listo para usar con las diferentes herramientas que pueden ser necesarias durante un análisis forense. Las más populares e importantes son PALADIN EDGE [147], SIFT Workstation [193] y CAINE [39].

PALADIN EDGE no es una distribución pensada para ser instalada en un equipo, sino que ha sido creada para ser ejecutada directamente desde una unidad de almacenamiento extraíble. Está basada en Ubuntu y su objetivo es asistir en diferentes tareas forenses relacionadas con la adquisición de evidencias. Por ejemplo, incluye herramientas para buscar y previsualizar ficheros (tanto por palabras clave como por tipos MIME), generar imágenes en varios formatos, clonar dispositivos y crear una imagen tanto del espacio no asignado como del espacio libre de un disco duro.

SIFT Workstation, a diferencia de PALADIN EDGE, es una distribución que sí tiene que ser instalada en un equipo antes de su uso. También se basa en Ubuntu e incluye herramientas de código abierto orientadas en ayudar en las tareas de respuestas ante incidentes así como en las tareas forenses. Algunas de sus capacidades son la creación imágenes, el análisis de memoria y la generación de líneas temporales.

La distribución CAINE está pensada tanto para ser ejecutada directamente desde una unidad de almacenamiento extraíble como para ser instalada en un ordenador. También está basada en Ubuntu y proporciona herramientas que facilitan las tareas a lo largo de todo el análisis forense. Por ejemplo, para ayudar en la inspección de las evidencias, su gestor de ficheros incluye diferentes *scripts* para previsualizar rápidamente muchas bases de datos, historiales de navegación, registros de Windows, archivos borrados así como extraer metadatos de diversos tipos de ficheros. Además, con el fin de mantener la integridad de las pruebas, CAINE automáticamente monta cualquier unidad de almacenamiento en modo solo lectura, impidiendo así cualquier escritura accidental.

Capítulo 2

Navegación privada

En los últimos años han ocurrido diferentes acontecimientos que han despertado mucha atención sobre la importancia de la privacidad. Revelaciones como la de Edward Snowden [87, 86], el escándalo de Cambridge Analytica [214] o la gran filtración de datos de Equifax [26] han dado un gran impulso para que el mundo sea consciente del nivel al que llega la recopilación de datos personales. Los datos personales son recogidos, compartidos y vendidos por un gran número de servicios en línea. Servicios que, la mayoría de las veces, no necesitan manejar tal cantidad de datos personales para realizar su función. Sin embargo, siguen recopilándolos porque es muy fácil y muy barato almacenarlos con la excusa de que podrían ser útiles en el futuro. Y lo que es aún peor, las empresas que recopilan estos datos no invierten lo suficiente en proteger adecuadamente la información, lo que da lugar a filtraciones. Filtraciones en las que las mayores víctimas son los usuarios, cuyos datos personales quedan expuestos a cualquiera con acceso a Internet.

Con el fin de garantizar un mayor control sobre los datos recopilados por las empresas, el Parlamento Europeo introdujo el Reglamento General de Protección de Datos (RGPD) en 2016. El RGPD restringe y determina como deben gestionarse los datos personales. Por ejemplo, ahora las empresas deben informar a los usuarios de qué información están recopilando, deben eliminar los datos una vez que ya no sean necesarios, no deben recopilar más información de la estrictamente necesaria y deben garantizar que los datos están protegidos con las medidas de seguridad adecuadas [9]. Desde la introducción del RGPD, las empresas han reducido el uso de cookies, los

internautas han notado un aumento en el número de consentimientos y los sitios web muestran a los usuarios qué información recopilan y con qué fin [123, 9]. También se han publicado nuevos estudios que pretenden ayudar a las empresas a adaptarse al nuevo reglamento. Por ejemplo, en [40] se presenta una metodología para detectar posibles violaciones de privacidad al realizar operaciones con grandes cantidades de datos, como la integración de datos o la vinculación de registros.

Para ayudar a los usuarios a ser conscientes de como se comparten sus datos personales entre distintos servicios web, existen herramientas que muestran gráficamente como fluye la información en la red durante una sesión de navegación. Por ejemplo, herramientas como CHRAVAT [48] o VIPAT [33] permiten obtener gráficos en tiempo real de los distintos proveedores con los que interactúa el usuario mientras navega por la web.

Cuanto más conscientes sean los usuarios de la importancia de la privacidad, más demandarán herramientas que les ayuden a preservarla. Algunas de las mejoras ya están entre nosotros, como el cifrado por defecto del sistema de ficheros en los teléfonos inteligentes [125], la incorporación de cifrado de extremo a extremo en las aplicaciones de mensajería más populares [4, 71, 111], un control más granular a la hora de conceder permisos a las aplicaciones en los sistemas Android e iOS [8, 117, 243], un aumento del 55 % de las páginas web que utilizan HTTPS en los últimos 10 años [127] y un largo etcétera. Todo ello gracias a que los usuarios poco a poco vamos tomando conciencia del problema y exigiendo un mayor nivel de seguridad y privacidad. Porque no debemos olvidar que, como dijo Glenn Greenwald [135], "Todos necesitamos lugares donde podamos ir a explorar sin que los ojos juiciosos de otras personas se posen sobre nosotros [...] Es realmente en el ámbito privado donde residen la disidencia, la creatividad y la exploración personal".

Es curioso, pero la forma en la que exploramos el mundo ha dado un cambio radical en los últimos 30 años. Antes, la "herramienta" principal para ampliar nuestros conocimientos eran los libros, las enciclopedias, los manuales, las bibliografías... Ahora, nuestra herramienta preferida es el navegador web. Desde la creación de la *World Wide Web* en el año 1990 por parte Tim Berners-Lee [126], los navegadores web han evolucionado enormemente. Gracias a ellos tenemos acceso a un mundo virtual inmenso con el que ampliar nuestro conocimiento y, al mismo tiempo, explorar el mundo físico. Desde acceder a la enciclopedia más grande del mundo hasta "pasear" por las calles

de cualquier ciudad que nos apetezca. Y todo con un "sencillo" navegador.

En el mundo físico, cuando vamos a una biblioteca en búsqueda de nueva información, no nos gusta que nadie nos juzgue o nos vigile cada vez que abrimos un libro en concreto. Nos gusta poder explorarlo libremente, expandir nuestro conocimiento y disfrutar del libro sin preocupaciones. Esta sensación de libertad la queremos seguir manteniendo en el mundo virtual. No queremos que nos pongan freno a nuestra curiosidad ni que almacenen en un servidor remoto de forma permanente qué "libro" hemos decidido ojear. Por eso es tan importante que nuestra herramienta actual, el navegador web, incorpore las medidas necesarias para proteger nuestra privacidad y, con ella, nuestro deseo de curiosidad.

Afortunadamente, cada vez hay más servicios en línea y más herramientas que incluyen funcionalidades pensando en la privacidad de los usuarios. Una de estas medidas es el modo de navegación privada incluido en la mayoría de los navegadores web modernos. Este modo está diseñado para evitar que cualquier información relacionada con la navegación se almacene en el dispositivo que se está utilizando. Por supuesto, no es perfecto. Pero tenemos que seguir exigiendo que las herramientas incluyan cada vez más medidas y que sean cada vez más robustas. Dada la importancia de los navegadores web a día de hoy, y de la relevancia de disfrutar de un buen nivel de privacidad en línea, en este capítulo presentamos una metodología con la que poner a prueba este modo de navegación y poder revelar así sus posibles puntos débiles.

Es importante remarcar que nuestro objetivo con esta metodología no es mostrar cómo capturar el dispositivo a estudiar, o cómo descubrir el código de bloqueo de un equipo, o revelar una vulnerabilidad que permita saltarse el bloqueo de un dispositivo. Nuestro objetivo es dar a conocer qué información podría recuperar un analista, o un atacante, de una sesión de navegación privada una vez que el dispositivo cae en sus manos. Con esta metodología buscamos evaluar la efectividad del modo privado y responder a preguntas como las siguientes: ¿Se puede recuperar información sobre la navegación realizada en modo privado si se ha cerrado por completo el navegador? ¿Quedó algún artefacto en el dispositivo sobre la navegación realizada si he reiniciado el dispositivo?

En las próximas secciones se hace una revisión de los trabajos ya existentes en este ámbito y se presenta en detalle la metodología creada.

2.1. Estudios relacionados

En la literatura se pueden encontrar trabajos previos que estudian posibles filtraciones de información del modo privado incluido en los navegadores. En esta sección se hace un repaso a los trabajos más relevantes y exhaustivos de los últimos años. Para facilitar la lectura, esta sección está dividida en dos subsecciones. La primera está dedicada a revisar los estudios realizados en sistemas operativos de escritorios y, la segunda, en sistemas operativos móviles. Ambas subsecciones van revelando los trabajos por orden cronológico, del más antiguo al más reciente.

2.1.1. Sistemas de escritorio

Uno de los primeros trabajos más completos publicados sobre el modo de navegación privada fue el Gaurav Aggarwal *et al.* [5] en el año 2010. Analizaron el distinto comportamiento de este modo de navegación en los siguientes navegadores: Internet Explorer 8, Mozilla Firefox 3.5, Safari 4, y Google Chrome 5. Detectaron varias inconsistencias como, por ejemplo, que Safari utilizaba las *cookies* creadas en el modo normal dentro del modo privado o que Mozilla Firefox permitía utilizar los certificados de cliente SSL desde el modo privado. También destacaron aspectos que están fuera del control del navegador y que podrían dificultar el correcto funcionamiento del modo privado. Por ejemplo, el sistema operativo podría guardar en caché las peticiones DNS realizadas o podría mover a la memoria de intercambio (*swap* en inglés) algunas páginas de memoria que contuvieran información sensible. Este último punto lo demostraron con Mozilla Firefox, donde fueron capaces de recuperar algunas de las URL visitadas de la *swap*. Otro aspecto que estudiaron fue para qué fines utilizaban el modo privado los usuarios. Para ello, utilizaron redes publicitarias para ejecutar un código JavaScript que permitía detectar si el usuario estaba navegando en modo privado. Luego, centraron sus esfuerzos en el navegador Mozilla Firefox. Llevaron a cabo una revisión de parte del código fuente así como ejecutaron diversos tests unitarios, lo que les permitió detectar funcionalidades del navegador que no tenían en cuenta si el modo privado estaba activado o no, lo que anulaba por completo las ventajas de este modo de navegación. Y no se detuvieron ahí, sino que también analizaron un total de 32 extensiones, de las cuales 16 filtraban información sensible al disco. Por último, presentaron una extensión que, antes de abrir el modo privado, deshabilitaba

las extensiones que podrían filtrar la navegación realizada y no las volvía a habilitar hasta que se cerraba el modo privado.

Los años siguientes no vieron mucha actividad hasta que, en el año 2014, Calum Findlay y Petra Leimich [74] publicaron su estudio sobre el comportamiento de Mozilla Firefox en cuatro situaciones diferentes: modo normal y privado con Mozilla Firefox instalado en el sistema y modo normal y privado utilizando la versión portable de Mozilla Firefox. Su objetivo era establecer cuáles eran las condiciones que reducían la cantidad de datos filtrados, maximizando así la privacidad del usuario. Observaron que no se escribió ningún dato en el almacenamiento permanente durante las sesiones de navegación privada, ni con la versión instalada ni con la versión portátil. La única excepción en la que existía la posibilidad de escritura en disco era si el sistema operativo decidía mover una página de memoria que contuviera información sensible a la memoria de intercambio. Como buena práctica, recomiendan reiniciar el ordenador tras finalizar una sesión de navegación para evitar la recuperación de información de la RAM.

Al año siguiente, Reza Montasari y Pekka Peltola [136] estudiaron qué información se podía extraer al realizar un análisis forense de un equipo tras haber navegado en modo privado. Para ello pusieron a prueba los siguientes navegadores: Chrome 26, Firefox 20, Internet Explorer 9 y Safari 5. Todos ellos fueron ejecutados en una máquina virtual de VirtualBox con Windows. Con cada uno de los navegadores realizaron varias actividades, como reproducir un vídeo en YouTube, buscar un producto en Amazon o previsualizar un archivo PDF. Para realizar el análisis, decidieron volcar la RAM justo antes de cerrar el navegador y crear una imagen del disco justo después de cerrar el navegador. Las herramientas utilizadas, tanto para la captura de evidencias como para su procesamiento, fueron FTK Imager, Autopsy, FTK y WinHex. Los resultados del análisis de las imágenes de disco revelaron que Google Chrome fue el único que no dejó ningún artefacto en disco. Sin embargo, el análisis de la RAM sí reveló casi todas las actividades realizadas en modo privado, independientemente del navegador utilizado.

Los estudios continúan en el 2016, donde P. Anuradha *et al.* [12] estudiaron qué información se podía recuperar de una imagen de disco creada después de haber borrado los artefactos de navegación generados. El navegador elegido para las pruebas fue Google Chrome ejecutándose en Ubuntu 14.04. La sesión de navegación consistió

en ver vídeos en YouTube, buscar imágenes en Google Imágenes, buscar artículos en Amazon y acceder a Gmail. Una vez finalizada la sesión de navegación usando Google Chrome en modo normal (no incógnito), borraron manualmente todos los artefactos de navegación. A continuación, crearon una imagen del disco duro y realizaron una serie de búsquedas con la herramienta AccessData Forensic Tool Kit. Los resultados muestran que pudieron recuperar gran parte de la información generada, incluidas algunas de las imágenes visualizadas. Sin embargo, no consiguieron recuperar las contraseñas utilizadas ni los vídeos reproducidos.

En el año 2017, Nikolaos Tsalis *et al.* [219] estudiaron el modo privado de Google Chrome 47, Mozilla Firefox 43, Internet Explorer 11 y Opera 34. Realizaron el análisis desde el punto de vista de un atacante que tuviera acceso físico temporal al ordenador después de que un usuario hubiera navegado utilizando el modo privado y hubiera dejado el ordenador encendido. La herramienta de monitorización escogida en este caso fue Process Monitor [182]. Tras realizar los experimentos en una máquina virtual con Windows 7, descubrieron situaciones en las que se producían violaciones de privacidad que no deberían haberse producido en modo privado según la documentación de los navegadores. En una de las pruebas descubrieron que, al guardar un marcador en Mozilla Firefox o en Google Chrome, automáticamente se almacenaba información adicional que indicaba si se había creado desde el modo privado. En otra prueba descubrieron que al cambiar los permisos específicos de un sitio web, como bloquear las cookies de un sitio concreto, estos ajustes se almacenaban permanentemente en las preferencias del navegador. En el caso de Mozilla Firefox, también descubrieron que las respuestas del Protocolo de Verificación de Certificados en Línea (en inglés, *Online Certificate Status Protocol* o, abreviadamente, OCSP) se almacenaban en la carpeta de caché del navegador, filtrando los sitios web a los que se había accedido. Como solución para evitar este tipo de filtraciones, los autores sugieren almacenar el perfil del navegador en un sistema de archivos virtual alojado en un medio volátil (como la memoria RAM).

Graeme Horsman [98] realizó un análisis a nivel de proceso de Google Chrome durante una sesión de navegación privada. La versión de Google Chrome analizada fue la 55 ejecutándose en Windows 7. Para mostrar la interacción con el sistema operativo, utilizó las herramientas disponibles en la suite Sysinternals de Microsoft. En concreto, uno de los experimentos realizados consistió en comparar el número de eventos del

sistema generados por una sesión normal con una sesión de incógnito. El resultado fue una disminución significativa de eventos en la sesión privada. Tras el análisis, y a pesar de que no pudo verificar el contenido de algunos de los archivos temporales creados por el navegador, el autor concluye que la mejor forma de recuperar la navegación realizada es analizando el contenido de la memoria RAM.

En el año 2018 se publicaron estudios que exploraron el modo privado de navegadores menos populares. Por ejemplo, Szu-Yuan Teng *et al.* [208] ejecutaron en un Windows 10 virtualizado los siguientes seis navegadores que ofrecen funciones de incógnito: Epic Privacy Browser, Secure Browser, Comodo Dragon, SRWare Iron, Dooble y Maxthon. Las pruebas consistieron en analizar el tráfico de red generado al abrir cada uno de los navegadores así como analizar un volcado de la memoria creado después de haber utilizado cada uno de ellos. De este volcado de memoria consiguieron obtener el nombre de usuario y la contraseña que habían utilizado para iniciar sesión en un sitio web desde el modo privado. Su conclusión fue que el modo privado supone un reto para el análisis forense, siendo solo posible recuperar información valiosa cuando se accede al contenido de la memoria asignada al navegador.

Llega el año 2019 y con él dos de las publicaciones más interesantes hasta el momento. La primera es de Abid Khan Jadoon *et al.* [107]. En su trabajo diseñaron e implementaron una serie de escenarios para poner a prueba la privacidad y el anonimato que ofrece el navegador Tor Browser. Como entorno de pruebas utilizaron un Windows 8.1 virtualizado con VMware Workstation. La adquisición de datos la realizaron en tres momentos distintos: 1) con el navegador recién abierto y sin haber realizado ninguna actividad, 2) con el navegador aún en ejecución y tras haber completado las actividades predefinidas y 3) después de cerrar el navegador tras haber realizado las actividades predefinidas. Realizaron un análisis exhaustivo del registro, la RAM y el disco duro que les llevó a la conclusión de que el navegador Tor Browser deja un gran número de artefactos, especialmente en memoria, lo que permite recuperar muchas de las actividades realizadas en línea.

La segunda publicación a destacar del 2019 es la de Graeme Horsman *et al.* [99]. Probaron el modo de navegación privada de 30 navegadores diferentes. Cada uno de ellos se ejecutó en una máquina virtual independiente de Windows 10 en VirtualBox. La prueba consistió en visitar cinco URL con cada uno de ellos y, a continuación, utilizar un término de búsqueda para cada URL con el fin de determinar si alguno

de los navegadores había escrito en disco algún dato relacionado con la navegación. Los resultados revelaron que solo 5 de los 22 navegadores que ofrecían modo privado filtraron información de la sesión de navegación al disco. Algunas de las limitaciones destacadas fueron el uso de una plataforma de virtualización o la reducida longitud de la sesión de navegación.

Hasta donde sabemos, la publicación más reciente en este ámbito es la Rebecca Nelson *et al.* [142] en el año 2020. Su trabajo se centró en realizar un análisis exhaustivo de los siguientes navegadores: Mozilla Firefox 55, Google Chrome 61 y Tor Browser 7. La principal herramienta de análisis forense utilizada fue FTK y todos los navegadores fueron probados en máquinas virtuales con Windows 7. Los resultados muestran que, cuando utilizaron Firefox y Chrome en modo normal, pudieron recuperar prácticamente toda la sesión de navegación. En el momento en el que activaron el modo privado, la información que pudieron recuperar se redujo drásticamente. Además, cuando pusieron a prueba Tor Browser, descubrieron que no pudieron obtener prácticamente nada relacionado con la navegación realizada. Cabe destacar que todo el análisis se centró exclusivamente en la información que se puede recuperar de una imagen de disco, por lo que no incluye el análisis de la memoria volátil.

2.1.2. Sistemas móviles

En la literatura se puede encontrar una gran variedad de estudios que realizaron un análisis forense digital a diferentes aplicaciones para Android. Por ejemplo, hay un elevado número de estudios que analizaron diversas aplicaciones de mensajería instantánea como IMO [1, 216, 202], WeChat [253, 245, 246, 173], Telegram [185, 173], WhatsApp [192, 173], Twitter [246] o Viber [173]. También hay trabajos que estudiaron el correo electrónico, como [222] y [44]. U otras investigaciones, como [13], donde Dimitris Apostolopoulos *et al.* exploraron si era posible recuperar las credenciales de autenticación de un total de 30 aplicaciones mediante el volcado de la memoria asignada a dichos procesos.

En lo que respecta al análisis forense de navegadores web, el número de trabajos centrados en plataformas móviles, concretamente Android, es bastante reducido, especialmente si se compara con el número de trabajos centrados en plataformas de escritorio. En los párrafos siguientes se incluyen, por orden cronológico, algunos de los estudios más relevantes y recientes que realizan un análisis forense a distintos

navegadores ejecutándose en Android.

En 2013, Nedaa Al Barghouthy *et al.* [7] pusieron a prueba el navegador Orweb corriendo en un teléfono Samsung Galaxy S2 con Android 2.3.3. Su objetivo era determinar qué información podía recuperarse del sistema de ficheros tras utilizar el navegador. En concreto, las actividades realizadas con el navegador consistieron en visitar Facebook, iniciar sesión, iniciar una conversación con un amigo y enviarle una foto utilizando Facebook Message. Repitieron las pruebas con el dispositivo totalmente de serie y con el dispositivo *rooteado*. Para capturar las evidencias utilizaron las aplicaciones Titanium Backup y Android File Explorer. Su análisis reveló que, con el dispositivo de serie, no fue posible recuperar ningún artefacto relacionado con la navegación debido a la falta de privilegios para acceder a carpetas sensibles, como la carpeta que almacena el perfil del navegador. Sin embargo, con el dispositivo *rooteado*, pudieron acceder a todo el sistema de archivos, lo que les permitió recuperar algunas de las actividades realizadas con Orweb de la carpeta que contiene el perfil del navegador.

En 2014, estos investigadores [6] repitieron su experimento anterior pero, esta vez, utilizando las ideas presentadas por Timothy Vidas *et al.* [225] en 2011. El problema de muchas de las herramientas comerciales que permiten la adquisición física de un dispositivo es que requieren privilegios de superusuario. Al *rootear* un dispositivo se están modificando las particiones del sistema y del usuario, lo que podría estar destruyendo pruebas importantes. Por ello, el enfoque de [225] fue sustituir la partición de recuperación por las herramientas necesarias para obtener las evidencias del dispositivo, evitando así modificar el resto de particiones. Con este enfoque, Nedaa Al Barghouthy *et al.* prepararon dos escenarios diferentes en los que probar el navegador Orweb. En el primer escenario, utilizaron un Samsung Galaxy S2 de serie pero instalando en la partición de recuperación una imagen de ClockWorkMod. ClockWorkMod es una herramienta de recuperación que ofrece una serie de funcionalidades avanzadas como la instalación de una nueva imagen del sistema, la realización de tareas de mantenimiento o la creación y restauración de copias de seguridad. Gracias a esta herramienta pudieron ejecutar Nandroid para obtener una imagen del sistema de ficheros completo. En el segundo escenario, *rootearon* el dispositivo, modificando así las particiones del sistema y del usuario. A continuación, utilizaron directamente Nandroid para obtener una imagen del sistema de archivos. Los resultados del análisis de ambas

imágenes revelaron la misma cantidad de información sobre la navegación realizada con Orweb. Estos resultados apoyaron la idea presentada por Timothy Vidas *et al.*: no es necesario ser superusuario para poder obtener pruebas de un dispositivo Android. La conclusión de este trabajo fue que, tras unos resultados casi idénticos, es preferible no *rootear* el dispositivo. Sin embargo, también destacaron que su análisis no incluyó la adquisición de la memoria del dispositivo, lo que habría requerido privilegios de *root*.

En 2021, L. B. Younis *et al.* [252] probaron Google Chrome, Mozilla Firefox, Dolphin y Opera en un emulador de Android llamado Nox Player. Cada uno de los navegadores se probó en modo normal y privado. La sesión de navegación realizada consistió en visitar varios sitios web y acceder a Gmail para enviar y recibir varios correos electrónicos. La obtención de evidencias se limitó a la creación de un volcado de memoria inmediatamente después de finalizar la sesión de navegación. El análisis de las pruebas con la herramienta Autopsy reveló que Google Chrome proporcionó el nivel más bajo de privacidad, ya que pudieron recuperar las URL visitadas así como el contenido de los correos electrónicos en ambos modos de navegación. El navegador que ofreció el mayor nivel de privacidad de los cuatro fue Mozilla Firefox. Cuando se ejecutó en modo normal, consiguieron recuperar el historial de navegación y las direcciones de correo electrónico, pero no el contenido de los correos. Y, cuando se ejecutó en modo privado, solo pudieron recuperar las direcciones de correo electrónico. Hay que señalar que, al haber realizado todas las pruebas en un emulador, la extrapolación de los resultados a un dispositivo real puede no ser directa.

En 2022, Warren Thompson [213] se centró en determinar qué artefactos podían recuperarse del sistema de archivos tras utilizar el modo privado de diferentes navegadores en Android. En concreto, los navegadores analizados fueron Google Chrome, Mozilla Firefox, DuckDuckGo y Tor Browser ejecutándose en una máquina virtual de VirtualBox con Android x86 versión 9.0-r2. Las pruebas consistieron en acceder a varias páginas web (añadiendo algunas a favoritos), iniciar sesión en varios sitios web (guardando las credenciales en los navegadores que lo permitían), enviar y recibir correos electrónicos entre dos cuentas de correo y realizar varias búsquedas en diferentes buscadores en línea. Una vez finalizada la navegación, usó FTK Imager para crear una imagen del disco duro de la máquina virtual. El análisis de las distintas imágenes reveló que DuckDuckGo fue el que más artefactos creaba, permitiendo incluso la

recuperación de las imágenes cargadas durante la navegación. El mejor de los cuatro fue Mozilla Firefox, ya que generó el menor número de artefactos en disco. El autor concluye que varios de los artefactos se recuperaron del espacio no asignado. Por lo tanto, repetir este tipo de análisis en un dispositivo real podría ser complicado, ya que obtener una imagen completa del disco no es sencillo, ni siquiera con privilegios de superusuario. Además, hay que tener en cuenta que, desde hace ya varias versiones de Android, el disco está totalmente cifrado, lo que dificulta aún más la obtención de información del espacio no asignado.

En la literatura también existen diferentes kits de herramientas (o *toolkits*) para Android que permiten adquirir artefactos forenses de diferentes aplicaciones de forma automatizada. Por ejemplo, en el caso de los navegadores web se puede destacar AndroKit. AndroKit extrae del sistema de archivos del dispositivo diferentes artefactos de los navegadores Google Chrome, Opera, Mozilla Firefox y Dolphin. Algunos de los artefactos que es capaz de obtener son los marcadores, los archivos descargados, el historial web, las credenciales de usuario o las sesiones almacenadas. Además, AndroKit puede *rootear* el dispositivo para acceder con éxito a las carpetas donde los navegadores almacenan sus perfiles. En [17], los autores demostraron el funcionamiento de AndroKit ejecutándolo tanto en un emulador de Android como en dos teléfonos Samsung.

Es importante señalar que este tipo de kits de herramientas no se centran en probar el modo privado ni en analizar el contenido de la memoria. Por lo tanto, la cantidad de información que pueden recuperar de una sesión de navegación privada es increíblemente reducida. Solo podrían recuperar información sobre la navegación realizada en modo privado si el navegador dejó algún artefacto en el sistema de archivos.

2.2. Metodología para el análisis del modo privado

Como se puede ver en la Sección 2.1, en la literatura se pueden encontrar diferentes trabajos que exploran la eficacia del modo privado de diferentes navegadores. Sin embargo, en estos trabajos los navegadores suelen probarse de forma superficial y poco estructurada, lo que dificulta la comprobación del correcto funcionamiento del modo privado en diferentes entornos o la comparación del nivel de privacidad ofrecida por los distintos navegadores. Por ello, el objetivo de la metodología que presentamos

en este trabajo es proporcionar una forma coherente y exhaustiva de probar el modo privado de los diferentes navegadores web, ayudando a determinar el nivel de efectividad de esta característica de privacidad, ya que la metodología permitirá conocer qué información puede ser recuperada después de haber navegado en modo privado. Además, todo el proceso está diseñado de forma que sea completamente independiente del navegador y del sistema operativo utilizados, por lo que puede aplicarse a un gran número de entornos.

Comparando la metodología que se presenta en este capítulo, así como los casos de uso donde se puso a prueba, con trabajos anteriores, las principales aportaciones pueden resumirse en los siguientes puntos:

1. La RAM se captura en más situaciones. Los trabajos que se pueden encontrar en la literatura solo vuelcan la memoria cuando el navegador está en ejecución o justo después de cerrarlo. La metodología presentada permite obtener una perspectiva más amplia de la información que se puede recuperar en distintas circunstancias.

2. Ningún trabajo anterior, hasta donde sabemos, ha estudiado si era posible recuperar el llavero de contraseñas de un navegador a partir de un volcado de memoria. Como se verá en el siguiente capítulo, los casos de uso donde se probó la metodología demuestran que, en algunas situaciones, se pueden obtener todas las contraseñas guardadas en el navegador con solo un volcado de memoria.

3. Los trabajos previos mencionados anteriormente utilizaron entornos virtualizados o entornos *bare metal*, pero no ambos. Los casos de uso presentados para la metodología incluyen ejecuciones en ambos entornos. Además, también se probaron distintas opciones de fortificación del *kernel* en un entorno adicional con el fin de estudiar como afectan al comportamiento de los navegadores.

4. Se diseñó una sesión de navegación que incluye todas las actividades utilizadas en trabajos anteriores así como otras adicionales no contempladas hasta la fecha. Las fases de despliegue, ejecución y captura de evidencias se han diseñado y ejecutado para cubrir un amplio número de escenarios, lo que ha permitido estudiar a fondo el comportamiento de la navegación privada.

La metodología diseñada en este trabajo está formada por las siguientes fases:

1. Configuración del entorno y del navegador a estudiar.
2. Monitorización de los cambios.
3. Sesión de navegación.
4. Adquisición de las evidencias.
5. Análisis.

A continuación, se describe en detalle cada una de estas cinco fases.

2.2.1. Configuración del entorno

El objetivo de esta primera fase es diseñar y desplegar el entorno (o entornos) donde el navegador seleccionado va a ser puesto a prueba. Es esencial considerar el uso de más de un entorno, ya que permite obtener una visión más amplia de como se comporta el navegador en diversas situaciones. En este caso, cuando se usan múltiples entornos, es importante tener en consideración las siguientes características:

- Diferentes sistemas operativos o diferentes opciones de bajo nivel del mismo sistema operativo.
- Un entorno sin virtualizar y otro virtualizado.

La inclusión de un entorno que no sea virtualizado es de vital importancia, ya que no se debe asumir que la información que se puede recuperar de un entorno virtualizado es la misma que en un entorno sin virtualizar. Hay que tener en cuenta que la gestión de la memoria en un entorno virtualizado, por ejemplo, es totalmente diferente debido a la incorporación de la capa del hipervisor.

Cuando se configura un equipo para realizar las pruebas se debe prestar especial atención a los siguientes puntos:

- Dedicar un dispositivo para las pruebas con una instalación del sistema operativo limpia. De esta forma, cualquier artefacto encontrado tiene que haber sido creado por el navegador. También es aconsejable utilizar otro ordenador para todas las tareas posteriores de análisis y de procesamiento, evitando así la contaminación del equipo destinado a los experimentos.

- Deshabilitar las actualizaciones automáticas. Con el fin de prevenir que la versión del navegador cambie entre distintas ejecuciones del mismo experimento es necesario deshabilitar las actualizaciones automáticas. Para ello, hay dos opciones: 1) configurar el sistema para que no actualice el navegador web, pero sí el resto de los programas o 2) configurar el sistema para que no realice ninguna actualización. El problema con la primera opción es que se introducen más variables cuando se realizan los experimentos. Por ejemplo, si al repetir un experimento se obtiene un resultado diferente, va a ser más difícil determinar si el problema está en el propio experimento o en un cambio realizado por alguna de las actualizaciones. Así que se recomienda emplear la segunda opción. Una ventaja adicional al deshabilitar las actualizaciones automáticas es que el número de procesos en segundo plano es menor, lo que permite aislar mejor el comportamiento del navegador.

Configuración del navegador

Una vez decidido el entorno o entornos que se van a utilizar es el momento de preparar el navegador seleccionado. El único requisito para aplicar la metodología es añadir un nombre de usuario y una contraseña al llavero del navegador. Sin embargo, la configuración concreta del navegador dependerá de lo que se vaya a probar. Por ejemplo, esta metodología podría utilizarse para determinar si merece la pena instalar una extensión que elimina las *cookies* creadas por un sitio web después de salir de él. En este caso, serían necesarios dos perfiles: uno con la extensión instalada y otro sin ella. Cuando se analizaran los resultados, se comprobaría si la extensión hizo su trabajo correctamente y, por lo tanto, si merece la pena usarla o no.

Otra situación donde se podría utilizar la metodología es para comprobar si la función de "Borrar los datos de navegación al salir" incluida en algunos navegadores realmente funciona correctamente. Para ello se podrían realizar ejecuciones independientes con esta opción activada y desactivada. Una vez realizadas las ejecuciones, tomando las precauciones oportunas entre una y otra para no contaminar las pruebas, se compararían los resultados para verificar si esta opción realmente disminuye la cantidad de información que se puede recuperar.

2.2.2. Monitorización de los cambios

En esta fase se debe establecer como se monitorizarán los cambios realizados por el navegador en el sistema de ficheros y como se volcará la memoria RAM del sistema. Las herramientas a utilizar variarán en función de los sistemas operativos seleccionados en la fase de configuración del entorno.

Para monitorizar los cambios hechos en el sistema de ficheros se debe seleccionar una herramienta que permita registrar cada vez que se crea, modifica o elimina un fichero o una carpeta en uno de los directorios vigilados. La opción recomendada cuando se especifican los directorios a monitorizar es incluir solo las carpetas donde está almacenado el perfil del navegador así como las carpetas temporales. La otra opción sería monitorizar todos los cambios realizados en todo el sistema de ficheros durante la ejecución de los experimentos. Sin embargo, esta opción complica mucho más el análisis posterior.

Para obtener un volcado del contenido de la RAM existen herramientas que permiten volcar el espacio de memoria de un proceso en particular. Sin embargo, este tipo de herramientas no es válido para el propósito de esta metodología por dos motivos:

1. Al volcar solamente la memoria asignada al proceso del navegador en un momento concreto, no se estarían volcando otras áreas de memoria que pudieron estar asignadas previamente al navegador, pero que fueron liberadas.

2. No sería posible volcar la memoria asociada al proceso del navegador después de haberlo cerrado o después de haber reiniciado el equipo. Esto es debido a que el proceso del navegador ya no existiría y, por lo tanto, no sería posible volcar su espacio de memoria.

Debido a los motivos anteriores es necesario seleccionar una herramienta que permita obtener un volcado completo de la RAM. Para ello, se puede utilizar tanto hardware especializado como diversas soluciones software [116].

2.2.3. Navegación

En esta fase debe diseñarse una sesión de navegación que genere datos para su posterior análisis. El tener esta sesión planificada de antemano tiene dos ventajas principales: 1) es fácilmente reproducible y 2) permite un análisis más dirigido en

la siguiente fase. Esta sesión debe incluir las acciones más comunes que se realizan al usar un navegador web. Tal y como mencionan en [136], algunas de estas tareas son la descarga de ficheros, la realización de búsquedas, la visualización vídeos en *streaming* o la previsualización de documentos PDF. Con estas tareas en mente, y teniendo en consideración las metodologías descritas en [136, 138], se ha creado un esquema para una sesión de navegación que incluye una variante de estas acciones, así como otras nuevas. Como nuevas actividades cabe destacar el inicio de sesión en un sitio web, la introducción de una URL en la barra de direcciones pero sin acceder a ella y la utilización de la información de inicio de sesión almacenada en el llavero del navegador.

El esquema completo y detallado de la sesión de navegación es el siguiente:

1. Acceder a una página web que sirva contenido multimedia, como música o vídeos. Realizar una búsqueda y reproducir uno de los elementos devueltos por la búsqueda.
   Motivación: ¿Es posible recuperar las palabras introducidas en el campo de búsqueda y el nombre del elemento reproducido?

2. Abrir una nueva pestaña y acceder a una página web que almacene una *cookie* en el navegador.
   Motivación: ¿Es posible recuperar la *cookie* creada por el sitio web?

3. Abrir una nueva pestaña y acceder a un sitio web que aloje algún tipo de archivo que pueda previsualizarse en el navegador sin necesidad de descargarlo. Proceder a la previsualización de dicho fichero.
   Motivación: ¿Es posible recuperar el nombre y el contenido del archivo previsualizado?

4. Abrir una nueva pestaña e introducir una URL en la barra de direcciones. Sin acceder al sitio web introducido, borrar la URL escrita. Es importante no utilizar una URL que pueda ser fácilmente encontrada en memoria o en disco. Por ejemplo, debe evitarse utilizar una URL como `bing.com`, ya que producirá muchos falsos positivos debido a que es uno de los motores de búsqueda incorporados en la mayoría de los navegadores. Independientemente de la URL seleccionada, es aconsejable asegurarse de que no hay ninguna coincidencia antes de realizar las

pruebas. Para ello se debe buscar por la URL en el directorio donde el navegador almacena el perfil del usuario así como en un volcado de la memoria creado con solo el navegador en ejecución.
Motivación: ¿Es posible recuperar la URL introducida?

5. Abrir una nueva pestaña y acceder al sitio web cuya información de inicio de sesión está almacenada en el llavero del navegador. Iniciar sesión con las credenciales guardadas.
Motivación: ¿Es posible obtener toda la información almacenada en el llavero del navegador en texto claro directamente de los volcados de memoria?

6. Abrir una nueva pestaña y acceder a una página web distinta de la anterior que también solicite iniciar sesión. Introducir la información solicitada por el sitio web para iniciar sesión (nombre de usuario, contraseña...). Los datos introducidos en el formulario pueden ser valores arbitrarios y no tienen por qué corresponder con los datos de una cuenta válida.
Motivación: ¿Es posible recuperar la información introducida?

2.2.4. Adquisición de evidencias

El objetivo de esta fase es recopilar los datos necesarios de la máquina de pruebas para su posterior análisis. En concreto, se trata de obtener la lista de cambios realizados en el disco duro así como un volcado completo de la memoria RAM. Para que las ejecuciones de prueba sean lo más "limpias" posible, la captura de los cambios en disco debe realizarse en una ejecución completamente independiente a cuando se vuelca el contenido de la RAM. Además, solo se puede probar un navegador a la vez. Las siguientes subsecciones describen en detalle como y cuando deben realizarse las diferentes adquisiciones.

Disco duro

En este caso, los pasos a realizar son muy sencillos:

1. Ejecutar la herramienta de monitorización elegida en la Sección 2.2.2.

2. Realizar la sesión de navegación diseñada según el esquema descrito en la Sección 2.2.3.

3. Cerrar el navegador.

4. Detener la herramienta de monitorización.

Memoria

En términos generales, la memoria debe volcarse en cuatro momentos diferentes:

T1. Con el navegador en ejecución y tras haber completado la sesión de navegación diseñada.

T2. Después de cerrar el navegador.

T3. Tras reiniciar el ordenador.

T4. Después de que el ordenador haya estado apagado durante 10 segundos.

Al igual que en el caso de la adquisición de los cambios realizados en disco, las ejecuciones tienen que ser completamente independientes. Es decir, en una ejecución determinada solo se puede volcar la memoria en uno de los momentos descritos anteriormente. De este modo, se obtienen los volcados de memoria más "limpios" posibles.

2.2.5. Análisis

En esta fase, el objetivo es recuperar la mayor cantidad de información posible sobre la sesión de navegación a partir de todas las evidencias capturadas. Por un lado, hay que analizar la lista de ficheros obtenida por la herramienta que monitoriza los cambios en el disco duro. Hay que examinar cada uno de esos ficheros para comprobar si es posible obtener información sobre alguna de las actividades realizadas. Por otro lado, hay que analizar los diferentes volcados de memoria, intentando recuperar la mayor cantidad de información posible de cada uno de ellos.

Para analizar la lista de ficheros obtenida se recomienda desarrollar *scripts* que realicen búsquedas por palabras clave relacionadas con la navegación realizada. Además, cuando el navegador se ejecuta en una máquina virtual, también se deben repetir las búsquedas directamente sobre los discos virtuales. Esto asegura que el navegador no haya escrito en uno de los directorios que no estaba siendo monitorizado. En el caso de los volcados de memoria se recomienda el uso de las siguientes herramientas multiplataforma: Volatility Framework [227] y wxHexEditor [247]. Además,

para Volatility Framework existe el *plugin* Actaeon [2, 84], el cual facilita el análisis de volcados de memoria que contienen máquinas virtuales en ejecución. Este *plugin* puede ser muy útil cuando se analiza un volcado de memoria en el que el navegador se está ejecutando en una máquina virtual. Como en el caso de los ficheros, también se pueden escribir *scripts* para automatizar la búsqueda de información relacionada con la navegación.

Para el caso particular de recuperar el contenido completo del llavero, las herramientas a utilizar varían en función del navegador. Sin embargo, existen herramientas multiplataforma, como HackBrowserData [91], que permiten recuperar las contraseñas guardadas, el historial, las cookies y los marcadores de distintos navegadores.

> Las aportaciones más significativas de este capítulo se han publicado y extraído del siguiente artículo:
>
> - Xosé Fernández-Fuentes, Tomás F. Pena y José C. Cabaleiro, "Digital forensic analysis methodology for private browsing: Firefox and Chrome on Linux as a case study", *Computers & Security*, Volume 115, April 2022, 102626. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2022.102626. Contribución: Conceptualización, Metodología, Investigación, Redacción (borrador original).

Capítulo 3

Recuperación de información de sesiones de navegación privada

En el capítulo anterior se presentó la metodología creada para analizar el modo privado de los navegadores, destacando en primer lugar cuál era el propósito de la metodología para posteriormente describir cada una de las fases y pasos que la componen. Una de nuestras principales prioridades cuando se estaba diseñando la metodología era que tenía que ser multiplataforma. De poco serviría a día de hoy diseñar una nueva metodología para una única plataforma, ya que disponemos de una gran variedad de dispositivos con los que poder navegar por la red. Actualmente disponemos de un navegador en ordenadores, móviles, tabletas, consolas portátiles e incluso en televisores.

En este capítulo se presentan dos casos de uso. Ambos están centrados en las plataformas mayoritarias y en los navegadores más populares. Aunque, tal y como se mencionó anteriormente, es posible navegar por la red con otro tipo de dispositivos, el número de usuarios y el tráfico generado por ellos es bastante pequeño. Por lo tanto, los casos de uso están centrados en la privacidad ofrecida por sistemas destinados para PC y a sistemas que ejecutan Android.

3.1. Firefox y Chrome en Linux

El primer caso de uso trata de estudiar el modo privado de los navegadores Mozilla Firefox y Google Chrome ejecutándose en cuatro entornos diferentes basados en Linux. En el primer entorno, el navegador se ejecutó en un ordenador con Ubuntu. En el segundo, se ejecutó en un Ubuntu ligeramente fortificado. En el tercero, se ejecutó en una máquina virtual de Ubuntu, con Ubuntu como sistema anfitrión y VirtualBox como hipervisor. Por último, el cuarto entorno es el mismo que el tercero, pero utilizando VMware como hipervisor.

El hecho de probar los navegadores en diferentes entornos permitió verificar que el modo privado seguía funcionando correctamente, incluso cuando se utilizaba en situaciones distintas o menos comunes. Como sistema operativo se decidió utilizar una distribución de Linux porque, hasta donde sabemos, no hay ningún trabajo centrado en él. El único trabajo que realizó las pruebas en Linux fue [12]. En dicho trabajo, los autores pusieron a prueba el comportamiento de Google Chrome ejecutándose en Ubuntu, pero no estudiaron el funcionamiento del modo privado ni analizaron el contenido de la RAM. Por lo tanto, el comportamiento del modo privado en Linux sigue siendo un área pendiente de exploración.

Las secciones que vienen a continuación detallan como se aplicó cada una de las fases de la metodología así como los resultados obtenidos.

3.1.1. Configuración del entorno

Todos los experimentos se realizaron en los cuatro entornos descritos, los cuales serán referenciados como entorno A, entorno B, entorno C y entorno D. Se desplegaron en un PC con un procesador Core i7-4700K y 8 GB de memoria RAM DDR3-1600 MHz. Como se recomienda en la Sección 2.2.1, este equipo se utilizó exclusivamente para la ejecución de las pruebas. Todas las tareas posteriores de análisis y procesamiento se realizaron en un equipo distinto.

En el entorno A, el navegador se ejecutó directamente en el sistema operativo anfitrión. El sistema elegido en este caso fue Ubuntu 20.04 con la versión 5.4.0-26-generic del *kernel*. Se realizó una instalación limpia y solo se realizaron tres cambios en el sistema:

1. Se desactivaron las actualizaciones automáticas.

2. Mozilla Firefox se actualizó a la versión 95.0.

3. Se instaló la versión 96.0.4664.110 de Google Chrome.

Las versiones de ambos navegadores eran las más recientes en el momento de la realización de las pruebas.

El entorno B es casi igual al entorno A. La diferencia es que se añadieron dos opciones de arranque: `init_on_alloc=1` y `init_on_free=1`. Como puede leerse en la descripción del *commit* donde se introdujeron estas opciones [50], `init_on_alloc=1` pone a cero las nuevas páginas de memoria así como los objetos del montón (*heap*), mientras que `init_on_free=1` pone a cero las páginas que han sido liberadas así como los objetos del montón (*heap*) que han sido borrados. El propósito de este entorno era probar la eficacia de estas opciones y determinar si proporcionaban alguna mejora en términos de privacidad.

En el entorno C, el navegador se ejecutó en una máquina virtual de VirtualBox. Para ello, partiendo del entorno A, se instaló VirtualBox 6.1.30 y se creó una máquina virtual con 4 GB de RAM asignadas. En ella se ejecutó el mismo sistema operativo con los mismos cambios mencionados para el entorno A.

En el entorno D, el navegador también se ejecutó en una máquina virtual pero, en este caso, utilizando VMware como hipervisor. Como en el caso anterior, partiendo del entorno A, se instaló VMware Workstation Pro 16.2.1 y se creó una máquina virtual con 4 GB de RAM asignadas. El sistema operativo ejecutado en la máquina virtual y las modificaciones realizadas fueron las mismas que en el entorno C.

La finalidad de estos dos últimos entornos era determinar si, desde el punto de vista de la privacidad, había alguna ventaja en aislar el navegador en una máquina virtual.

Configuración del navegador

La configuración del navegador utilizada fue la predeterminada. La única preparación realizada fue añadir un nombre de usuario y una contraseña al llavero del navegador. Para ello, se accedió a la página web `https://mail.protonmail.com/login` y se introdujo `test` como nombre de usuario y `1234` como contraseña. Una vez introducidos, se añadieron estos datos al llavero. Esta acción fue realizada utilizando el modo de navegación normal porque, a pesar de que Mozilla Firefox sí permite añadir nuevas

entradas desde el modo privado, Google Chrome no lo permite. Una vez guardada la información de inicio de sesión, se borraron el historial, la caché y las *cookies*.

3.1.2. Monitorización de los cambios

En esta sección se describen las herramientas utilizadas para registrar los cambios realizados por el navegador. Todos los datos obtenidos en esta fase se presentarán en la siguiente sección, donde se comprobará qué rastros dejaron los navegadores.

Sistema de ficheros

Para comprobar que el navegador no escribía en disco ningún dato que pudiera revelar la actividad realizada en el modo privado fue necesario monitorizar los cambios realizados en el sistema de ficheros. Para ello se utilizó la herramienta `inotifywait`. Esta forma parte del paquete `inotify-tools` [104], que agrupa un conjunto de programas para monitorizar eventos del sistema de ficheros.

Esta herramienta permite capturar una gran variedad de eventos. Para este caso de uso concreto, la captura se limitó a los siguientes tipos:

- `create`. Se ha creado un archivo o directorio dentro de un directorio vigilado.
- `modify`. Se ha escrito en un archivo vigilado o en un archivo dentro de un directorio vigilado.
- `delete`. Se ha eliminado un archivo o directorio dentro de un directorio vigilado.

A la hora de configurar la herramienta es obligatorio especificar los ficheros o carpetas a vigilar. En el caso de Mozilla Firefox, los directorios vigilados fueron:

- `~/.mozilla/`
- `~/.cache/mozilla/`

Estos son los dos directorios dentro de la carpeta personal del usuario donde Mozilla Firefox escribe datos. Estas rutas, que pueden obtenerse accediendo a la dirección `about:profiles`, se denominan “Directorio raíz” y “Directorio local”, respectivamente.

En el caso de Chrome, los directorios monitorizados fueron:

- `~/.config/google-chrome/`
- `~/.cache/google-chrome/`

Estas son las carpetas que contienen los datos del usuario según la documentación del navegador [46].

Memoria

La herramienta utilizada para volcar la memoria por completo fue LiME [204] (*commit* ID fa37b69). LiME es un módulo cargable del núcleo que, cuando se carga, vuelca el contenido de la memoria volátil en el archivo que se le pasa como argumento. Este archivo puede utilizarse posteriormente, por ejemplo, con la herramienta Volatility Framework.

Una herramienta alternativa a LiME es AVML [19]. La principal diferencia con respecto a LiME es que no es necesario conocer, *a priori*, la distribución de Linux o la versión del *kernel* sobre la que se va a ejecutar.

3.1.3. Navegación

Tomando como base el esquema de la Sección 2.2.3, la secuencia específica de pasos que formaron la sesión de navegación utilizada es la siguiente:

1. Entrar en `https://www.youtube.com` y buscar `kernel bugs`. Reproducir el vídeo con el título `Syzbot and the Tale of Thousand Kernel Bugs - Dmitry Vyukov, Google`. Después de 15 segundos, pausar el vídeo.
2. Abrir una nueva pestaña y acceder a `https://stackoverflow.com`.
3. Abrir una nueva pestaña y acceder a `https://meltdownattack.com`. Hacer clic en `Spectre Paper`. El navegador mostrará el contenido del archivo `spectre.pdf`.
4. Abrir una nueva pestaña y escribir `myurl.com` en la barra de direcciones. Sin acceder a este sitio web, borrar la URL escrita.
5. Acceder a `https://mail.protonmail.com/login`. Intentar iniciar sesión utilizando las credenciales almacenadas en el llavero del navegador. El inicio de sesión fallará.

6. Abrir una nueva pestaña e ir a `https://www.google.com/gmail`. Pulsar en el botón de `Iniciar sesión` e introducir `virtual112233@gmail.com` como nombre de usuario y `@thisis4testing1` como contraseña. El inicio de sesión fallará.

3.1.4. Adquisición de evidencias

Para obtener los cambios realizados en el disco por los navegadores se utilizó la herramienta mencionada en la Sección 3.1.2 y se siguieron los pasos descritos en la Sección 2.2.4.

En el caso de la RAM, el proceso es más complejo, ya que el procedimiento varía en función de si el navegador estaba siendo ejecutado en una máquina virtual o directamente en el sistema operativo anfitrión.

Cuando se estaba ejecutando directamente en el sistema anfitrión, los pasos realizados para crear los diferentes volcados fueron los siguientes:

T1. Encender el ordenador, ejecutar el navegador en modo privado, realizar la sesión de navegación y volcar la RAM.

T2. Encender el ordenador, ejecutar el navegador en modo privado, realizar la sesión de navegación, *cerrar el navegador, esperar un minuto* y volcar la RAM.

T3. Encender el ordenador, ejecutar el navegador en modo privado, realizar la sesión de navegación, *reiniciar el ordenador y, una vez iniciado,* volcar la RAM.

T4. Encender el ordenador, ejecutar el navegador en modo privado, realizar la sesión de navegación, *apagar el ordenador, esperar 10 segundos, encender el ordenador y, una vez iniciado,* volcar la RAM.

La Figura 3.1 muestra un resumen de las distintas acciones realizadas para obtener cada uno de los volcados.

En cambio, si el navegador estaba siendo ejecutado en una máquina virtual, el procedimiento es ligeramente diferente. En este caso, los pasos efectuados para crear los distintos volcados fueron los que se describen a continuación:

T1. Encender el ordenador, iniciar la máquina virtual, ejecutar el navegador en modo privado, realizar la sesión de navegación y volcar la RAM.

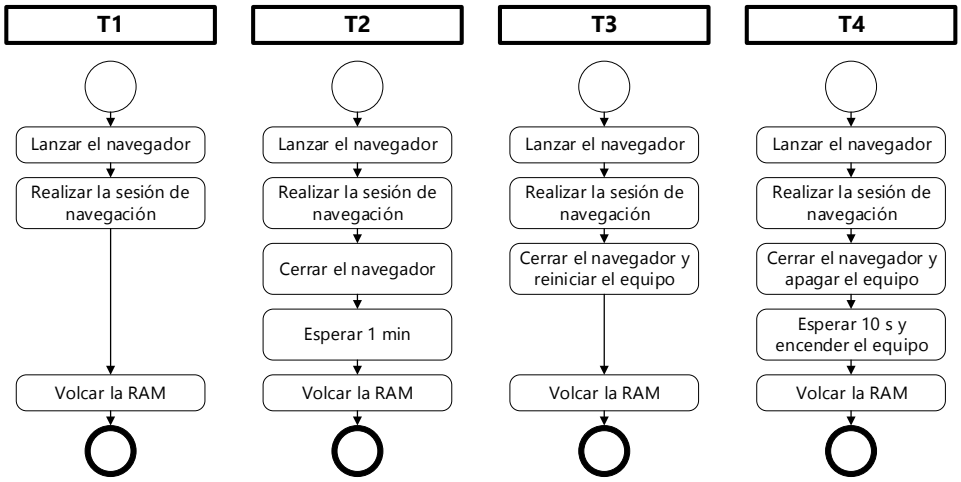

Figura 3.1: Esquema de los pasos realizados para obtener los diferentes volcados de memoria cuando el navegador se ejecutaba directamente en el equipo.

T2. Encender el ordenador, iniciar la máquina virtual, ejecutar el navegador en modo privado, realizar la sesión de navegación, *cerrar el navegador, apagar la máquina virtual* y volcar la RAM.

T3. Encender el ordenador, iniciar la máquina virtual, ejecutar el navegador en modo privado, realizar la sesión de navegación, cerrar el navegador, apagar la máquina virtual, *reiniciar el ordenador y, una vez iniciado,* volcar la RAM.

T4. Encender el ordenador, iniciar la máquina virtual, ejecutar el navegador en modo privado, realizar la sesión de navegación, cerrar el navegador, apagar la máquina virtual, *apagar el ordenador, esperar 10 segundos, encender el ordenador y, una vez iniciado,* volcar la RAM.

La Figura 3.2 muestra un diagrama de los pasos realizados con esta configuración.

En los cuatro entornos desplegados siempre se capturó toda la memoria del sistema anfitrión. En otras palabras, el módulo del núcleo LiME se cargó siempre en el anfitrión.

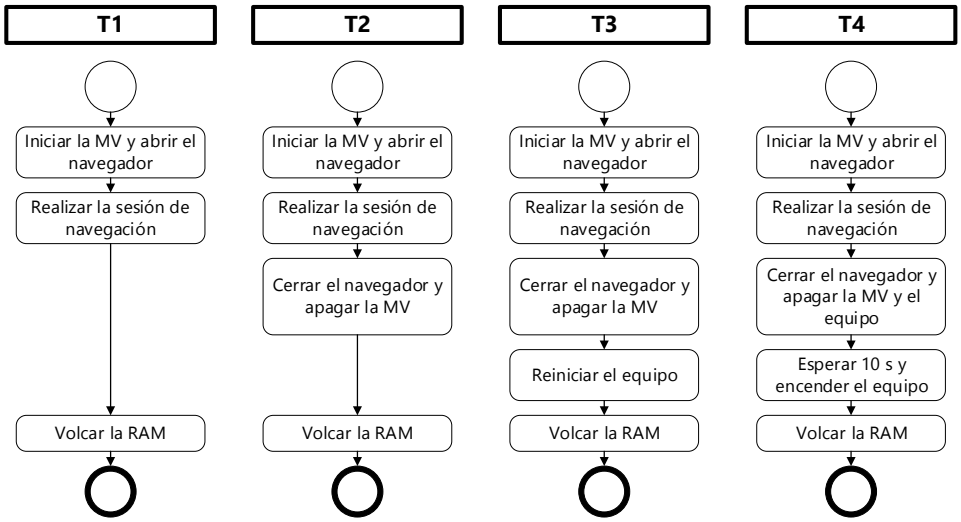

Figura 3.2: Esquema de los pasos ejecutados para obtener los diferentes volcados de memoria cuando el navegador se ejecutaba en una máquina virtual.

Es importante mencionar que hay algunas BIOS que borran el contenido de la memoria RAM al reiniciar [116], lo que impide recuperar información potencialmente interesante tras realizar un reinicio.

3.1.5. Análisis

Como se describe en la Sección 2.2.5, en esta fase se intenta recuperar toda la información posible tanto del disco duro como de los volcados de memoria.

Dada la sesión de navegación descrita en la Sección 3.1.3, se seleccionó el siguiente grupo de palabras clave para ser buscado en todos los archivos indicados por la herramienta de monitorización de disco:

1. `kernel bugs`
2. `kernel%20bugs`
3. `kernel+bugs`
4. `kernel%2Bbugs`

5. `Syzbot and the Tale of Thousand Kernel Bugs - Dmitry Vyukov, Google`
6. `prov=`
7. `spectre.pdf`
8. `myurl.com`
9. `virtual112233@gmail.com`
10. `virtual112233%40gmail.com`
11. `@thisis4testing1`
12. `%40thisis4testing1`
13. La clave `Chrome Safe Storage`

Los cuatro primeros elementos son búsquedas relacionadas con las palabras introducidas en el campo de búsqueda de YouTube. El número 2 es igual que el número 1 salvo que se ha sustituido el espacio por su código HTML (%20). Lo mismo se ha hecho en el número 4, donde el '+' se ha sustituido por su código hexadecimal (%2B). El número 5 corresponde al nombre completo del vídeo de YouTube reproducido. El número 6 corresponde a la *cookie* creada por el sitio `https://stackoverflow.com`. El número 7 es el nombre del archivo PDF previsualizado en el navegador. El número 8 corresponde a la URL escrita en la barra de direcciones, pero que no fue accedida. Los cuatro elementos siguientes son el nombre de usuario y la contraseña introducidos en la página de inicio de sesión de Gmail. Al igual que en los primeros ítems, la '@' ha sido sustituida por su código HTML (%40) en los ítems 10 y 12. Por último, si el navegador utilizado era Google Chrome, también se buscó por la clave `Chrome Safe Storage`. Esta clave se utiliza para cifrar, entre otras cosas, el contenido del llavero.

Para automatizar el análisis se desarrollaron varios *scripts* que buscaban por estas palabras clave en los distintos archivos indicados por la herramienta `inotifywait`. Además, en el caso de los entornos C y D, también se ejecutaron estos *scripts* sobre los discos virtuales para asegurarse de que el navegador no había escrito en ninguno de los directorios que no estaban siendo monitorizados.

En el caso de los volcados de memoria, el análisis se divide en dos partes. La primera parte consistió en buscar por las palabras clave mencionadas anteriormente en los distintos volcados para obtener el número de ocurrencias de cada uno. Para ello, se utilizó tanto el editor hexadecimal wxHexEditor como se crearon varios *scripts* para automatizar las búsquedas, al igual que como se hizo para el análisis del disco. La segunda parte consistió en recuperar ficheros directamente de los volcados de memoria. Concretamente, los ficheros a recuperar fueron el PDF previsualizado en el navegador y los archivos que almacenaban los llaveros de los navegadores. En el caso de Mozilla Firefox los ficheros eran `cert9.db`, `key4.db`, y `logins.json`. Y, en el caso de Google Chrome, el fichero a recuperar era `Login Data`. La principal herramienta utilizada para esta tarea fue Volatility Framework (*commit* ID 703b29b), aunque también se desarrolló un pequeño programa que buscaba por el contenido completo de estos ficheros e indicaba si era posible recuperarlos en su totalidad. De esta forma, se conseguía automatizar ligeramente el proceso de análisis.

En cuanto a las herramientas utilizadas para obtener el contenido de los llaveros, en el caso de Mozilla Firefox se utilizó Firefox Decrypt [75] (*commit* ID 557bb60) y, en el caso de Google Chrome, se desarrolló un *script* basado en Chrome-Password-Grabber [45]. En este caso en concreto se utilizaron herramientas distintas para cada navegador. Sin embargo, como se mencionó en la Sección 2.2.5, existen herramientas compatibles con varios navegadores y que son multiplataforma.

3.1.6. Resultados

Cada ejecución, ya fuera para monitorizar los cambios en el disco o para obtener un volcado de memoria, se repitió varias veces con el fin de verificar los resultados. Entre cada ejecución, se tomaron las siguientes precauciones para evitar contaminar los resultados:

- El contenido de las carpetas `~/.mozilla` y `~/.config/google-chrome` se restauró con un perfil limpio. Como se mencionó en la Sección 3.1.1, el único cambio realizado en los perfiles fue añadir un nombre de usuario y una contraseña al llavero del navegador.
- Se eliminó el contenido de las carpetas `~/.cache/mozilla/` y `~/.cache/google-chrome/`.

- En el caso de los entornos C y D, se restauró una instantánea limpia de la máquina virtual.
- Se apagó el ordenador y se tuvo desenchufado de la toma de corriente durante un mínimo de 1 minuto. El objetivo era intentar comenzar cada ejecución con una RAM "limpia". Como se describe en [89], el número de bits correctos que pueden recuperarse físicamente de la memoria DDR3 a temperatura ambiente es inferior al 50 % tras solo 10 segundos. También señalan que con este tipo de memoria, la única información recuperable tras un arranque en frío (*cold boot* en inglés) son patrones de ruido. Estudios más recientes [23, 251] demuestran que es posible descifrar el contenido de las memorias DRAM DDR3 y DDR4 realizando un ataque de arranque en frío (*cold boot attack* en inglés).

Los resultados obtenidos en cada uno de los escenarios descritos anteriormente se muestran en detalle en las siguientes subsecciones.

Resultados del análisis del sistema de ficheros

Mientras se llevaba a cabo la sesión de navegación se monitorizaron los cambios realizados en el sistema de archivos, tal y como se describe en la Sección 3.1.2. Tras analizar los ficheros indicados por la herramienta de monitorización, así como los discos virtuales utilizados en los entornos C y D, no se encontró ninguna información asociada a la navegación en ninguno de los ficheros.

Resultados del análisis de los volcados de memoria

El resultado de analizar los distintos volcados de memoria obtenidos en los entornos A, B, C y D se puede ver en las Tablas 3.1, 3.2, 3.3 y 3.4, respectivamente. Estas tablas muestran el número de veces que se encontró cada palabra clave, así como si fue posible o no recuperar los archivos mencionados en la fase de análisis (Sección 3.1.5).

A modo de ejemplo, en la Figura 3.3 se muestra el resultado de buscar por la palabra clave `kernel+bugs` en el editor hexadecimal y, en la Figura 3.4, el resultado de buscar por el nombre de usuario y la contraseña introducidos en la página de inicio de sesión de Gmail.

El proceso realizado con la herramienta Volatility Framework para recuperar el fichero PDF descargado, así como la suma de verificación SHA1 del fichero recuperado

Entorno A	Firefox				Chrome			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
Búsquedas por palabras clave								
`kernel bugs`	13	-	-	-	48	-	-	-
`kernel%20bugs`	18	-	-	-	14	-	-	-
`kernel+bugs`	40	-	-	-	58	-	-	-
`kernel%2Bbugs`	34	-	-	-	25	6	-	-
`Syzbot and the Tale of...`	17	-	-	-	38	3	-	-
`prov=`	2	-	-	-	1	-	-	-
`spectre.pdf`	167	2	-	-	72	1	-	-
`myurl.com`	3	-	-	-	0	-	-	-
`virtual112233@gmail.com`	11	11	-	-	17	-	-	-
`virtual112233%40gmail.com`	13	3	-	-	6	1	-	-
`@thisis4testing1`	25	2	-	-	2	-	-	-
`%40thisis4testing1`	4	2	-	-	3	-	-	-
Clave `Chrome Safe Storage` (*solo Chrome*)	-	-	-	-	4	4	4	3
Recuperación de ficheros								
`spectre.pdf`	Sí	No	No	No	Sí	No	No	No
`cert9.db` (*solo Firefox*)	No	Sí	No	No	-	-	-	-
`key4.db` (*solo Firefox*)	No	Sí	No	No	-	-	-	-
`logins.json` (*solo Firefox*)	No	Sí	No	No	-	-	-	-
`Login Data` (*solo Chrome*)	-	-	-	-	Sí	No	No	No

Tabla 3.1: Resumen del análisis de los volcados de memoria obtenidos en el entorno A. La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

Entorno B	Firefox				Chrome			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
Búsquedas por palabras clave								
`kernel bugs`	8	-	-	-	42	-	-	-
`kernel%20bugs`	11	-	-	-	25	-	-	-
`kernel+bugs`	37	-	-	-	55	-	-	-
`kernel%2Bbugs`	27	-	-	-	27	-	-	-
`Syzbot and the Tale of...`	14	-	-	-	36	5	-	-
`prov=`	2	-	-	-	2	-	-	-
`spectre.pdf`	159	-	-	-	75	-	-	-
`myurl.com`	3	-	-	-	0	-	-	-
`virtual112233@gmail.com`	7	-	-	-	17	-	-	-
`virtual112233%40gmail.com`	7	-	-	-	6	-	-	-
`@thisis4testing1`	7	-	-	-	2	-	-	-
`%40thisis4testing1`	2	-	-	-	2	-	-	-
Clave `Chrome Safe Storage` (*solo Chrome*)	-	-	-	-	5	4	2	2
Recuperación de ficheros								
`spectre.pdf`	Sí	No	No	No	Sí	No	No	No
`cert9.db` (*solo Firefox*)	No	Sí	No	No	-	-	-	-
`key4.db` (*solo Firefox*)	No	No	No	No	-	-	-	-
`logins.json` (*solo Firefox*)	No	Sí	No	No	-	-	-	-
`Login Data` (*solo Chrome*)	-	-	-	-	Sí	No	No	No

Tabla 3.2: Resumen del análisis de los volcados de memoria obtenidos en el entorno B. La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

Entorno C	Firefox				Chrome			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
Búsquedas por palabras clave								
`kernel bugs`	13	4	-	-	43	-	-	-
`kernel%20bugs`	26	13	-	-	17	6	-	-
`kernel+bugs`	41	27	-	-	56	1	-	-
`kernel%2Bbugs`	47	18	-	-	31	11	-	-
`Syzbot and the Tale of...`	16	4	-	-	40	-	-	-
`prov=`	2	-	-	-	2	-	-	-
`spectre.pdf`	156	60	-	-	76	11	-	-
`myurl.com`	3	2	-	-	0	-	-	-
`virtual112233@gmail.com`	8	1	-	-	20	2	-	-
`virtual112233%40gmail.com`	10	4	-	-	10	1	-	-
`@thisis4testing1`	24	2	-	-	4	-	-	-
`%40thisis4testing1`	4	-	-	-	4	-	-	-
Clave `Chrome Safe Storage` (*solo Chrome*)	-	-	-	-	2	-	-	-
Recuperación de ficheros								
`spectre.pdf`	Sí	No	No	No	Sí	No	No	No
`cert9.db` (*solo Firefox*)	Sí	No	No	No	-	-	-	-
`key4.db` (*solo Firefox*)	Sí	No	No	No	-	-	-	-
`logins.json` (*solo Firefox*)	Sí	No	No	No	-	-	-	-
`Login Data` (*solo Chrome*)	-	-	-	-	Sí	No	No	No

Tabla 3.3: Resumen del análisis de los volcados de memoria obtenidos en el entorno C. La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

Entorno D	Firefox				Chrome			
	T1	**T2**	**T3**	**T4**	**T1**	**T2**	**T3**	**T4**
Búsquedas por palabras clave								
`kernel bugs`	17	-	10	-	81	-	8	-
`kernel%20bugs`	51	15	26	-	41	8	36	-
`kernel+bugs`	62	30	16	-	109	2	16	-
`kernel%2Bbugs`	67	26	49	-	57	22	68	-
`Syzbot and the Tale of...`	21	4	12	-	60	-	10	-
`prov=`	3	-	-	-	3	-	-	-
`spectre.pdf`	254	216	230	-	110	23	45	-
`myurl.com`	6	6	10	-	0	-	-	-
`virtual112233@gmail.com`	26	10	16	-	34	27	26	-
`virtual112233%40gmail.com`	16	20	18	-	11	8	8	-
`@thisis4testing1`	54	26	63	-	8	5	4	-
`%40thisis4testing1`	3	6	6	-	7	7	14	-
Clave `Chrome Safe Storage` (*solo Chrome*)	-	-	-	-	6	5	2	-
Recuperación de ficheros								
`spectre.pdf`	Sí	No	No	No	Sí	No	No	No
`cert9.db` (*solo Firefox*)	Sí	Sí	No	No	-	-	-	-
`key4.db` (*solo Firefox*)	Sí	Sí	No	No	-	-	-	-
`logins.json` (*solo Firefox*)	Sí	Sí	No	No	-	-	-	-
`Login Data` (*solo Chrome*)	-	-	-	-	Sí	Sí	No	No

Tabla 3.4: Resumen del análisis de los volcados de memoria obtenidos en el entorno D. La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

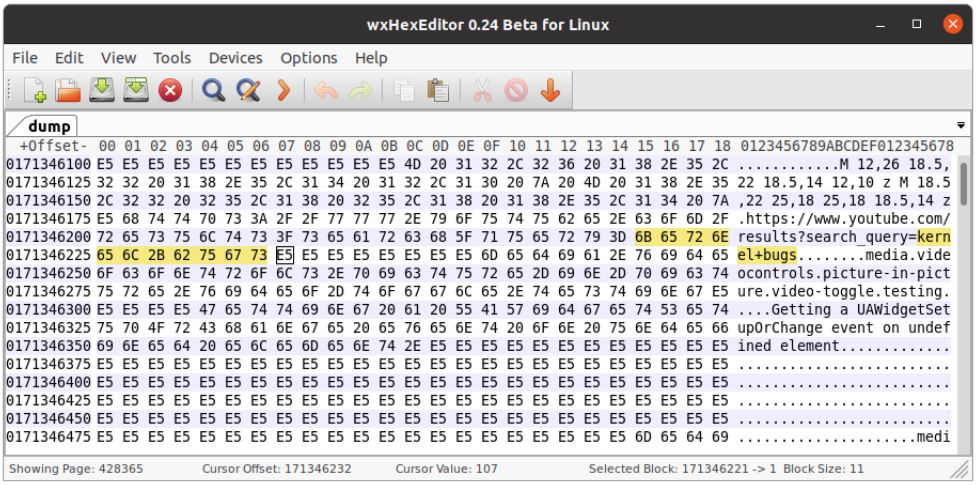

Figura 3.3: Ejemplo de una de las coincidencias encontradas al buscar las palabras clave introducidas en el campo de búsqueda de YouTube.

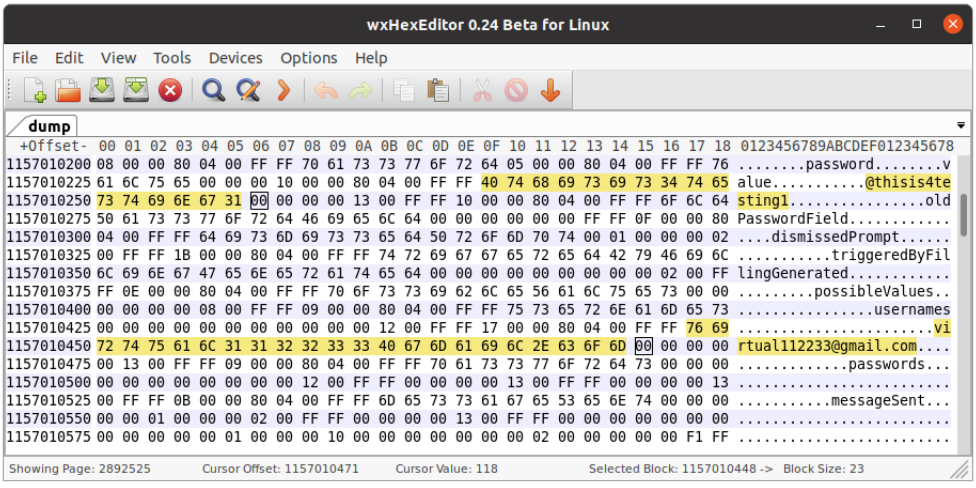

Figura 3.4: Ejemplo de una de las coincidencias encontradas al buscar la contraseña y el nombre de usuario introducidos en la página de inicio de sesión de Gmail.

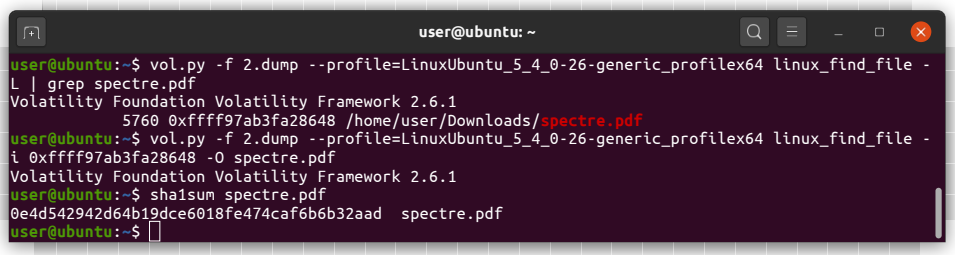

Figura 3.5: Recuperación del archivo PDF previsualizado en el navegador del volcado de memoria.

Navegador web	Coincidencias totales
Mozilla Firefox	2531
Google Chrome	1825

Tabla 3.5: Número total de coincidencias obtenidas para cada navegador teniendo en cuenta todos los volcados obtenidos en los distintos entornos probados.

Entorno	Coincidencias totales
A	677
B	591
C	822
D	2266

Tabla 3.6: Número total de coincidencias obtenidas para cada entorno teniendo en cuenta los dos navegadores probados.

para verificar que estaba intacto, se puede ver en la Figura 3.5. Básicamente lo que se hizo fue buscar la dirección *inode* asociada al fichero y luego se le pidió a la herramienta que lo extrajera.

La recuperación del contenido del llavero de Mozilla Firefox se puede ver en la Figura 3.6. Lo primero que se hizo fue recuperar del volcado los tres ficheros necesarios (`cert9.db`, `key4.db` y `logins.json`) y luego ejecutar la herramienta Firefox Decrypt. El procedimiento para recuperar el contenido del llavero de Google Chrome es prácticamente idéntico, lo único que cambia es la herramienta que se ejecuta para obtener el contenido del llavero.

En las Tablas 3.5 y 3.6 se muestran agrupadas por navegador y por entorno todas

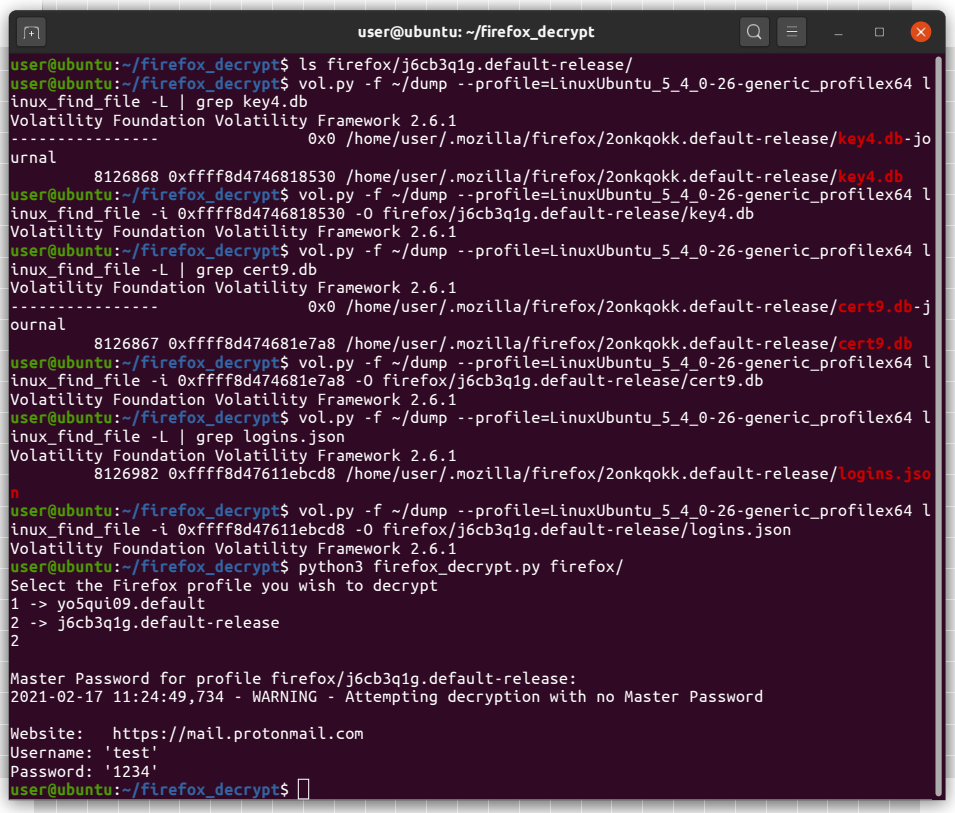

Figura 3.6: Recuperación del contenido completo del llavero de Mozilla Firefox desde el volcado de memoria.

las coincidencias obtenidas en todos los volcados. De esta forma, es posible visualizar qué entorno dejó más artefactos en memoria o qué navegador tiene menos fugas de información.

3.1.7. Discusión

En las siguientes subsecciones se discuten los resultados obtenidos tras analizar la RAM y el disco duro en las distintas configuraciones.

Disco duro

Los resultados de analizar los datos escritos en disco, tanto por Mozilla Firefox como por Google Chrome, muestran que ninguno de ellos escribió información que revelara la navegación realizada en modo privado.

A la hora de utilizar los llaveros integrados en ambos navegadores se detectó una diferencia que es importante mencionar. Pero, antes de eso, es necesario tener presente que acciones puede, o no, ofrecer un navegador a la hora de gestionar y utilizar el llavero. Estas acciones son las siguientes:

1. Añadir manualmente nuevas entradas desde las preferencias.
2. Guardar la información de inicio de sesión en el momento que uno se autentifica en un sitio web utilizando el modo de navegación normal.
3. Crear una nueva entrada tras haber iniciado sesión en un sitio web utilizando el modo de navegación privada.
4. Utilizar las entradas almacenadas desde el modo normal.
5. Hacer uso las entradas guardadas desde el modo privado.

Google Chrome permite todas las acciones salvo la número 3. Sin embargo, Mozilla Firefox permite realizar todas las acciones mencionadas. El hecho de que Mozilla Firefox permita realizar la acción número 3, almacenar una nueva entrada desde el modo privado, podría ser un problema. Aunque en la página de soporte de Mozilla [155] se indica claramente que "Si creas contraseñas y marcadores nuevos mientras usas la navegación privada, estos no se eliminarán al dejar de usarla", quizás Mozilla podría añadir un mensaje informativo en el navegador indicando que almacenar dicha información desde el modo privado puede estar comprometiendo la privacidad del usuario. Hay que tener en cuenta que el objetivo principal del modo privado es que no quede ningún rastro de la navegación realizada en el dispositivo. Si está permitiendo guardar los datos de acceso, está creando un registro en el disco que indica que se ha accedido al sitio web. Mozilla Firefox probablemente debería añadir un mensaje de advertencia en el navegador o directamente impedir realizar esta acción.

En los experimentos realizados no se analizó el contenido de la memoria de intercambio. La memoria de intercambio contiene páginas de memoria que han sido

copiadas al disco debido a una reducida cantidad de RAM disponible. Es importante tener en cuenta que si se está utilizando un navegador en modo privado, y la cantidad de RAM es muy limitada, existe la posibilidad de que algunas de las páginas de memoria asignadas al navegador se almacenen en la *swap*. Esto podría ser un problema, ya que implicaría que páginas de memoria con posible información sensible se almacenarían en disco.

Volcados de memoria creados en T1

A diferencia del análisis del sistema de ficheros, el análisis de la memoria sí permitió recuperar artefactos relacionados con la sesión de navegación, tal y como puede verse en la sección de resultados.

Es importante destacar que si solo se hubiera volcado el espacio de memoria asociado al navegador, las regiones de memoria que estuvieron asignadas al proceso del navegador, pero que fueron liberadas, habrían quedado sin analizar. Esto podría ser un problema debido a que esas zonas de memoria todavía podrían contener información sensible. Este comportamiento se ve claramente cuando, tras reiniciar el equipo, todavía fue posible recuperar información de la actividad realizada.

La primera actividad llevada a cabo fue acceder a YouTube y ver un vídeo. Independientemente del entorno o del navegador fue posible recuperar tanto las palabras clave introducidas en el campo de búsqueda como el nombre completo del vídeo reproducido. Cabe mencionar que en la sesión de navegación preplanificada se decidió utilizar YouTube como ejemplo. Sin embargo, esto demuestra que es posible recuperar tanto las palabras introducidas en un campo de búsqueda como el resultado elegido, independientemente de la página web visitada.

La segunda actividad consistió en acceder a un sitio web para posteriormente intentar recuperar el nombre y el contenido de la *cookie* creada. Al igual que en la actividad anterior, fue posible recuperar la *cookie* creada independientemente del entorno utilizado. El valor de esta *cookie* no es especialmente interesante. Sin embargo, demuestra que es posible recuperar las *cookies* creadas por cualquier sitio web. Por ejemplo, sería posible recuperar las *cookies* asociadas a un servicio de correo web y copiarlas en otro ordenador para hacerse pasar por ese usuario, siempre y cuando la *cookie* siga siendo válida.

La actividad número tres consistió en visualizar un archivo PDF directamente en

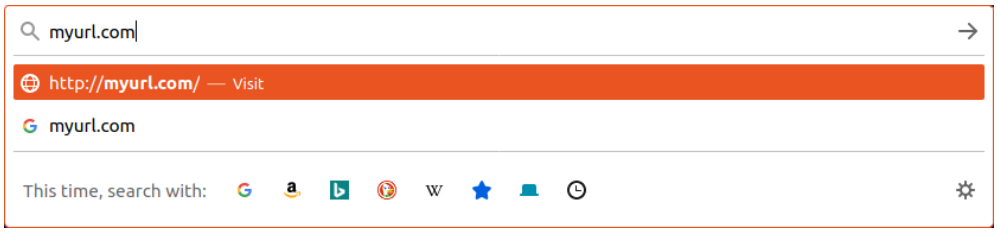

Figura 3.7: Opciones sugeridas al escribir en la barra de direcciones de Firefox.

el navegador. Como se vio en los resultados, tanto el nombre del archivo como el contenido completo del mismo se encontraban en memoria en todos los casos.

Una de las pruebas que puede parecer irrelevante es la número cuatro de la sesión de navegación. En ella, se introducía una URL en la barra de direcciones, pero no se accedía a ella. Los resultados muestran que fue posible recuperar las palabras introducidas en el caso de Mozilla Firefox. Teniendo en cuenta que las sugerencias de búsqueda vienen desactivadas por defecto en modo privado, las dos razones más probables por las que se observó este comportamiento son: 1) el navegador probablemente construyó la URL completa en segundo plano (añadiendo, por ejemplo, `https://`) a medida que se escribía y 2) el navegador preparó la URL necesaria para utilizar las palabras introducidas en la barra de direcciones como palabras clave a ser buscadas en el motor de búsqueda por defecto. La Figura 3.7 muestra este comportamiento en Mozilla Firefox al escribir en la barra de direcciones.

De forma similar a la actividad tres, la intención de la actividad número cinco era ver si se podían recuperar los archivos necesarios para obtener todas las contraseñas almacenadas en el navegador. Esto puede parecer irrelevante teniendo en cuenta que estos archivos ya estaban almacenados permanentemente en el disco. Sin embargo, esta prueba adquiere relevancia si se considera el caso en el que el perfil del navegador estuviera almacenado en un dispositivo extraíble. Este dispositivo podría conectarse a un ordenador compartido para consultar el correo electrónico, por ejemplo. Una vez terminado, el dispositivo se desconectaría y, en teoría, no debería ser posible recuperar la información de inicio de sesión introducida. Además, aunque el ordenador utilizado tuviera instalado un *keylogger*, ya sea hardware o software, la información de inicio de sesión no se habría filtrado. Esto se debe al hecho de que esta información habría sido introducida automáticamente por el llavero del navegador y no habría sido in-

troducida por teclado. El problema es que los archivos donde el navegador almacena las contraseñas podrían haber quedado en memoria y no haberse sobrescrito todavía. Esto significa que si esos archivos fueran recuperados, sería posible acceder no solo al nombre de usuario y contraseña utilizados en ese momento, sino a todos los nombres de usuario y contraseñas almacenados en el llavero. Como puede observarse en las tablas de resultados, en el caso de Mozilla Firefox, los ficheros asociados al llavero no estaban intactos en memoria en los entornos A y B. Sin embargo, sí se encontraron al ejecutar el navegador en los entornos C y D. En el caso de Google Chrome fue posible recuperar el fichero `Login Data` en todos los entornos.

La última actividad consistió en intentar iniciar sesión en Gmail. Los resultados muestran que, independientemente del entorno o del navegador, fue posible recuperar, en su totalidad, tanto el nombre de usuario como la contraseña introducidos. Aunque el inicio de sesión falló, el objetivo de este punto era ver si se podían recuperar los datos insertados.

Los resultados del análisis de los volcados de memoria con el navegador en ejecución muestran que fue posible recuperar prácticamente todas las actividades realizadas. En el caso de Mozilla Firefox no fue posible recuperar los ficheros del llavero en los dos primeros entornos y en el caso de Google Chrome no fue posible recuperar la dirección `myurl.com` en ninguno de los casos.

Volcados de memoria creados en T2

La Tabla 3.1 muestra que, en el entorno A, la cantidad de información que fue posible recuperar se redujo drásticamente tras el cierre del navegador. En el caso de Mozilla Firefox solo fue posible obtener el nombre del fichero PDF previsualizado, la información de inicio de sesión introducida en Gmail y los archivos que contenían el llavero del navegador. En el caso de Google Chrome fue posible recuperar las palabras introducidas en el campo de búsqueda de YouTube, el título del vídeo reproducido, el nombre del archivo PDF previsualizado, el nombre de usuario de Gmail y la clave `Chrome Safe Storage`. Aunque fue posible recuperar la clave, el archivo `Login Data`, que es el que contenía el llavero, no se encontró intacto en el volcado.

La Tabla 3.2, asociada al entorno B, revela una situación completamente diferente. En el caso de Mozilla Firefox solo fue posible recuperar dos de los tres archivos necesarios para obtener el contenido del llavero. El resto de la información generada por

Mozilla Firefox parece haberse borrado con éxito de la memoria gracias a la opción de arranque `init_on_free=1`. En el caso de Google Chrome solo fue posible recuperar el título del vídeo reproducido y la clave `Chrome Safe Storage`. Por lo tanto, la situación parece haber mejorado respecto al entorno anterior desde el punto de vista de la privacidad del usuario.

La Tabla 3.3, correspondiente al análisis del entorno C, presenta una situación totalmente distinta a la del entorno B. La cantidad de información que se pudo recuperar tras apagar la máquina virtual es considerable. En el caso de Mozilla Firefox fue posible recuperar todas las actividades realizadas a excepción de la *cookie* establecida por `stackoverflow.com`, el fichero PDF previsualizado y los archivos que contenían el llavero. En el caso de Google Chrome, la situación parece ligeramente mejor. Solo se pudieron recuperar las palabras introducidas en el campo de búsqueda de YouTube, el nombre del archivo PDF previsualizado y el nombre de usuario introducido en Gmail.

La Tabla 3.4, correspondiente al análisis del entorno D, muestra una situación ligeramente peor que la anterior. En el caso de Mozilla Firefox, la única información que no se pudo recuperar fue la *cookie* establecida por `stackoverflow.com` y el archivo PDF previsualizado. En el caso de Google Chrome no se pudo recuperar el título del vídeo reproducido, la *cookie* creada por `stackoverflow.com`, la dirección `myurl.com` ni el fichero PDF previsualizado. Cabe destacar que en ambos navegadores fue posible recuperar el llavero completo después de apagar completamente la máquina virtual.

Volcados de memoria creados en T3

Tras reiniciar el ordenador no se pudo recuperar ningún artefacto asociado a la navegación realizada en ninguno de los tres primeros entornos. Es cierto que la clave `Chrome Safe Storage` pudo recuperarse en los dos primeros entornos. Esto se debe a que la clave se almacena en el llavero del sistema, el cual se desbloquea cuando el usuario inicia sesión. Como en el entorno C la máquina virtual no se inició tras reiniciar el ordenador, el llavero del sistema operativo invitado no se desbloqueó.

La situación presentada por el entorno D es totalmente opuesta al resto de los entornos. En el caso de Mozilla Firefox se pudieron recuperar las evidencias suficientes como para recuperar prácticamente toda la sesión de navegación. Los únicos artefactos que no se encontraron en memoria fueron la *cookie* establecida por `stackoverflow.com`, el archivo PDF previsualizado y los archivos del perfil que contenían el llavero. En

el caso de Google Chrome no se encontró la *cookie* creada por `stackoverflow.com`, la dirección `myurl.com`, el archivo PDF previsualizado ni el archivo `Login Data`.

Estos resultados muestran que el uso de un hipervisor puede afectar a la gestión de la memoria, lo que puede ir en contra de la privacidad del usuario. Se trata de un resultado sorprendente, ya que la máquina virtual representa un entorno en el que la cantidad de RAM es más limitada, por lo que cabría esperar que la información en memoria se sobrescribiera con mayor facilidad, haciendo más difícil su recuperación.

También se puede comprobar que un simple reinicio no impidió que se recuperara información sensible en el entorno D. Como se mencionó en la Sección 2.2.4, algunas BIOS borran el contenido de la memoria RAM al reiniciar. Si la BIOS del ordenador utilizado para las pruebas hubiera borrado la memoria RAM al reiniciar, los resultados obtenidos en T3 habrían sido los mismos que en T4.

Volcados de memoria creados en T4

Después de tener el ordenador apagado durante 10 segundos no fue posible recuperar ninguna información asociada a la navegación, independientemente del entorno o navegador utilizado. La única información que se pudo recuperar fue la clave `Chrome Safe Storage` en los dos primeros entornos. Como se ha explicado anteriormente, esto se debe a que la clave se almacena en el llavero del sistema, el cual se desbloquea cuando el usuario inicia sesión.

Mejor entorno y mejor navegador

A la vista de las tablas resumen (Tablas 3.5 y 3.6) se puede concluir que el entorno B fue el que más dificultó la recuperación de información de los volcados. Si se toma de referencia el entorno A, del entorno B se pudo recuperar un 12,7 % menos de información, del entorno C, un 21,42 % más y, del entorno D, un 234,71 % más. En este caso es obvio que el uso de un hipervisor puede afectar seriamente al nivel de privacidad ofrecido por un navegador.

En cuanto a qué navegador se comportó mejor está claro que fue Google Chrome. Este permitió que se recuperara un menor número de artefactos en memoria teniendo en cuenta los cuatro entornos. A la vista de los resultados, Mozilla Firefox permitió recuperar un 38,68 % más de artefactos.

3.2. Chrome, Brave, Firefox y Tor Browser en Android

El teléfono inteligente se ha convertido en un dispositivo que no nos abandona en ningún momento del día: está listo tanto para despertarnos por la mañana como para avisarnos de que es hora de acostarse. Nos ayuda a nivel personal a mantenernos en contacto con familiares y amigos o a guardar permanentemente momentos especiales en forma de fotos o vídeos. Nos ayuda a nivel profesional a organizar nuestro calendario y no perdernos ninguna reunión importante o a estar al tanto de cualquier problema o emergencia en nuestra empresa. Y, por supuesto, también nos ayuda cuando queremos "desconectar", viendo un vídeo divertido, leyendo un libro o escuchando música en nuestra plataforma en línea preferida.

Hoy, más que nunca, realizamos la mayoría de las actividades que requieren una conexión a Internet con un teléfono móvil. En 2021, el tráfico de Internet generado por dispositivos móviles, ya sean teléfonos inteligentes o tabletas, fue superior al 57 % según StatCounter [200]. Esta cifra supone un incremento sustancial teniendo en cuenta que en 2016 no llegaba al 49 %, en 2013 no llegaba al 21 % y, en 2011, solo representaba el 7 %. En cuanto a la popularidad de los sistemas operativos móviles, el 70 % de los dispositivos móviles ejecutaban Android en el año 2021.

En la literatura se pueden encontrar diferentes artículos que evalúan el correcto funcionamiento del modo privado de diferentes navegadores. Tal y como se mostró en la Sección 2.1, muchos de estos trabajos se centraron en probar el modo privado en la versión de escritorio de los navegadores. Sin embargo, el número de trabajos que probaron las versiones móviles es mucho menor. No solo es menor, sino que no hay ningún trabajo que haya realizado las pruebas en un dispositivo Android real en los últimos años. El único trabajo, hasta donde sabemos, que hizo las pruebas en un dispositivo real fue [7]. Esta investigación fue publicada hace 10 años, solamente probó un navegador, no analizó el contenido de la memoria volátil y la sesión de navegación realizada fue increíblemente pequeña.

Los dos trabajos más recientes ([252, 213]) no realizaron las pruebas con un dispositivo real, sino que uno de ellos utilizó un emulador y el otro una máquina virtual. El emulador utilizado en [252] fue Nox Player. Este emulador está enfocado en la ejecución de juegos de Android en un sistema con Windows o con macOS. Los cambios realizados al sistema Android por la empresa Nox no son públicos, por lo que es muy difícil conocer el tipo de optimizaciones o modificaciones a nivel del sistema operativo

que han realizado. Esto impide conocer si, desde el punto de vista de seguridad y privacidad, representa un entorno realista y comparable con usar un dispositivo físico y actual. El otro problema con este emulador es que la última versión disponible está basada en Android 9. Esta versión de Android fue lanzada por primera vez en agosto de 2018 y, desde enero de 2022, está discontinuada [203]. El entorno de pruebas utilizado por el segundo trabajo mencionado ([213]) fue una máquina virtual de VirtualBox ejecutando Android-x86 [10]. Android-x86 es un proyecto de código abierto impulsado por la comunidad, cuyo objetivo es permitir la ejecución de Android en un sistema x86. La última versión disponible es la 9.0-r2, basada en Android 9. Sin embargo, este proyecto está actualmente discontinuado. El principal problema de utilizar entornos virtualizados o emuladores son los métodos de adquisición de evidencias utilizados. En este último trabajo mencionado ([213]), los autores crearon una imagen del disco de la máquina virtual utilizando una herramienta (FTK Imager [78]) pensada en crear imágenes forenses de datos de un ordenador. No existe una herramienta equivalente para dispositivos móviles. Por lo que este escenario no es realista, ya que como reconocen los propios autores, es muy difícil obtener una imagen bit a bit de la unidad *flash* de un dispositivo móvil. Y, aunque se consiguiera, intentar recuperar información del espacio no asignado, tal y como ellos hicieron, es incluso más difícil, ya que el disco se encuentra totalmente cifrado.

Otro punto importante a destacar de estos trabajos recientes, aparte de que utilizaron una versión obsoleta del sistema operativo, es que ambas de las soluciones utilizadas (Nox Player y Android-x86) no son oficiales ni están soportadas por Google. La principal arquitectura de Android es ARM64, y no hay soporte oficial para procesadores x86 [174]. Ni siquiera hay soporte oficial para ARM32, ya que desde agosto de 2021, Google obliga a que todas las aplicaciones disponibles en la Play Store tengan versión de 64-bit [195]. Y no solo eso, sino que el último dispositivo lanzado por Google (el Google Pixel 7) ni siquiera es capaz de ejecutar aplicaciones de 32-bit [228]. Por lo tanto, uno de los puntos que perseguimos a la hora de realizar las pruebas es que estas fueran realizadas en un dispositivo físico y actual.

El entorno de pruebas donde se aplicó la metodología consistió en una tableta y en dos dispositivos virtuales creados con el emulador oficial de Android. De esta forma, se pudo verificar hasta qué punto son extrapolables los resultados al ejecutar los navegadores en dos arquitecturas totalmente distintas (ARM64 y x86_64). El punto

de partida del análisis fue disponer del dispositivo a estudiar desbloqueado, o si tiene configurado algún tipo de bloqueo, se asumió que se conocía el código de desbloqueo.

El punto de partida puede parecer poco factible. Sin embargo, a continuación se referencian dos escenarios (uno de ellos sucediendo este 2023) donde no importa que el dispositivo esté bloqueado, ya que hay formas de conseguirlo en el estado deseado. El primero de ellos es como capturaron al creador de SilkRoad. Los agentes de la autoridad tenían que capturar el ordenador en el momento en el que el creador iniciara sesión como administrador en el sitio web SilkRoad. Fue irrelevante que el dispositivo tuviera cifrado completo del disco o que el creador utilizara contraseñas realmente robustas. Los agentes de la autoridad solo tuvieron que buscar y esperar por la situación oportuna para arrebatarle de las manos el ordenador desbloqueado en una biblioteca [140]. En el otro escenario, del que se pueden leer noticias de este mismo año [103], el atacante se sitúa en una posición estratégica en un sitio público, como un bar, y se fija cuando la víctima desbloquea el teléfono con el fin de ver claramente el código de desbloqueo. Después, simplemente roba el dispositivo en el momento oportuno. Google ya ha anunciado que Android 13 incluirá una función llamada "Enhanced Pin Privacy" destinada a evitar este tipo de ataques [35]

Las pruebas realizadas fueron diseñadas de forma que puedan ser repetidas en otro dispositivo sin necesidad de ningún hardware específico para la adquisición de evidencias. No importa que el dispositivo esté completamente cifrado, ya que nuestro punto de partida asume que se conoce el código de desbloqueo. Por supuesto, si se desconoce el código de desbloqueo, y se asume que no existe ninguna vulnerabilidad que permita saltarse dicho código, entonces las probabilidades de recuperar información útil del dispositivo son prácticamente nulas. Sin embargo, aunque este tipo de vulnerabilidades no es muy habitual, hace solo unos meses Google corrigió una vulnerabilidad en sus teléfonos Pixel que permitía saltarse por completo el código de desbloqueo [189].

Por otro lado, a la hora de escoger los navegadores queríamos que nuestras pruebas cubrieran los siguientes puntos: probar el navegador más popular, probar navegadores con diferentes motores y probar navegadores que estén centrados en la privacidad del usuario. El primer navegador seleccionado fue Google Chrome debido a tener una cuota de mercado de más del 60 % según StatCounter. El segundo navegador seleccionado por su crecimiento en los últimos años, y por estar centrado en proteger la privacidad de los usuarios, fue Brave. Estos dos navegadores están basados en el

proyecto Chromium, lo que quiere decir que ambos utilizan el motor Blink y utilizan V8 como motor de JavaScript. Para incluir navegadores con motores distintos y así añadir más variedad a las pruebas, decidimos añadir Mozilla Firefox y Tor Browser. Mozilla Firefox utiliza el motor Gecko y SpiderMonkey como motor JavaScript. Tor Browser está basado en Mozilla Firefox y su principal característica es el anonimato, haciendo uso de la red Tor para conseguirlo. La idea detrás de probar Tor Browser era determinar si el equipo de Tor Project había hecho cambios que dificultaran la obtención de pruebas en un análisis forense.

En las siguientes subsecciones se describe en detalle como se ejecutaron cada una de las fases de la metodología.

3.2.1. Configuración del entorno

Como ya se ha comentado, decidimos probar los navegadores seleccionados en tres entornos diferentes, uno con un dispositivo real y los otros dos virtualizados. Para el primer entorno utilizamos una tableta Samsung Galaxy Tab S6 Lite LTE (versión SM-P615) ejecutando el sistema operativo de fábrica actualizado a la última versión disponible en el momento de la realización de las pruebas (Android 13 con la versión del *kernel* 4.14.113). Para los entornos virtualizados decidimos utilizar el emulador de Android [181] (versión 32.1.11.0). En concreto, creamos un Android Virtual Device (AVD) con Android 13 (API 33 y ABI x86_64) y otro con Android 9 (API 28 y ABI x86_64), ambos utilizando el perfil de hardware de un Pixel 4. El motivo de crear estos dos AVD en concreto fue por las similitudes que tenían con la tableta de Samsung. El emulador de Android 13 ejecutaba la misma versión de Android que la tableta y el emulador de Android 9 utilizaba una versión del *kernel* de la misma rama que la tableta. Ambos AVD se ejecutaron en un Windows 11 versión 22H2 (OS Build 22621.1413) corriendo en un equipo con las siguientes características: un procesador Intel Core i7-9700K, 32 GB de RAM y un SSD Intel NVMe de 1 TB.

A pesar de que el uso de este tipo de emuladores no está pensado para ser utilizado por los usuarios finales de las aplicaciones, creemos que proporciona un nuevo punto de vista a como se comportan los navegadores en diferentes circunstancias. No hay que olvidar que los trabajos más recientes en este ámbito (como [252] y [213]) utilizan algún tipo de emulador. Por lo tanto, el hecho de añadir en nuestros experimentos un entorno real nos permite verificar como de extrapolables son los resultados obtenidos

en un emulador a los resultados obtenidos en un dispositivo de verdad.

Al preparar un dispositivo para las pruebas, se tuvieron en cuenta las siguientes consideraciones:

1. Utilización de un dispositivo "limpio". Para ello se restableció el dispositivo a los valores de fábrica. Esto permitió asegurar que los artefactos encontrados en las pruebas debían haber sido creados por el navegador que se está probando y no por otra aplicación o por un uso anterior. Una vez reseteado, se instalaron las actualizaciones necesarias así como las versiones deseadas de los navegadores a probar.

2. Desactivación de las actualizaciones automáticas tanto del sistema operativo como de las aplicaciones. El motivo era evitar que las versiones del sistema operativo y del navegador cambiaran entre distintas ejecuciones, lo que podría dar lugar a resultados incoherentes.

Una vez actualizada la tableta y creados los AVD, se instalaron los siguientes navegadores:

1. Google Chrome 104.0.5112.97

2. Brave 1.49.132

3. Mozilla Firefox 111.1.1

4. Tor Browser 102.2.1-Release (12.0.4)

Una cosa que es importante señalar es que los tres primeros navegadores tienen un modo de navegación normal así como un modo de navegación privada. Sin embargo, el navegador Tor Browser solo permite navegar en modo privado. Esto se debe a que este navegador está diseñado para anonimizar el tráfico de red generado así como para asegurar que no se almacena ninguna información relacionada con la navegación en el dispositivo utilizado.

La última preparación necesaria fue *rootear* la tableta para obtener acceso completo al sistema de archivos y así poder monitorizar los cambios realizados en él. En la Sección 3.2.3 se detallarán cómo se registraron estos cambios y qué herramientas se utilizaron. En el caso de los AVD, no fue necesario realizar modificaciones adicionales

Sitio web	Nombre de usuario	Contraseña
https://www.reddit.com	Test9861	oneRandomPassword41+
https://www.reddit.com	Test9862	oneRandomPassword42+
https://forums.linuxmint.com	Test99172	oneRandomPassword46+

Tabla 3.7: Información de inicio de sesión almacenada en el llavero de cada uno de los navegadores probados.

porque los AVD creados correspondían a las versiones sin la Play Store, por lo que permitían el acceso al usuario *root* por defecto.

3.2.2. Configuración del navegador

La configuración utilizada en cada uno de los navegadores fue la predeterminada. Lo único que se hizo fue añadir varios nombres de usuario y contraseñas al llavero de cada uno de los navegadores. En concreto, se almacenaron las entradas que se pueden ver en la Tabla 3.7.

Los pasos realizados para añadir dichas entradas a los navegadores fueron los mismos que se llevaron a cabo para la versión de escritorio de los navegadores (Sección 3.1.1). Básicamente, se accedió a cada uno de los sitios web sin utilizar el modo privado y se inició sesión. Una vez iniciada, el navegador automáticamente mostró un mensaje emergente ofreciendo la posibilidad de guardar las credenciales utilizadas. Otras opciones para añadir las entradas en los llaveros de los distintos navegadores serán discutidas en la Sección 3.2.8. Una vez configurados los navegadores, se eliminaron la caché, el historial y las *cookies* en cada uno de ellos.

3.2.3. Monitorización de los cambios

Esta sección describe como se registraron los cambios realizados por los navegadores en el sistema de ficheros así como el procedimiento utilizado para volcar la memoria del dispositivo y de los dos emuladores.

Sistema de ficheros

Para registrar las modificaciones realizadas en el sistema de ficheros se utilizó la herramienta `inotifywait for Android` [105]. Esta versión está basada en la herra-

Browser	Profile path
Google Chrome	`/data/data/com.android.chrome`
Brave	`/data/data/com.brave.browser`
Mozilla Firefox	`/data/data/org.mozilla.firefox`
Tor Browser	`/data/data/org.torproject.torbrowser`

Tabla 3.8: La ruta donde se encuentra el perfil de usuario en el sistema de ficheros para cada uno de los navegadores probados.

mienta `inotifywait` del paquete `inotify-tools` [104], pero adaptada para Android. Su funcionamiento y configuración es igual que la herramienta original.

Como se comentó en la Sección 3.1.2, el único parámetro necesario cuando se utiliza `inotifywait` es especificar el directorio, o directorios, a monitorizar. En este caso, los directorios supervisados fueron las carpetas donde los distintos navegadores almacenaban los perfiles del usuario. En concreto, la ruta monitorizada para cada uno de los navegadores puestos a prueba puede verse en la Tabla 3.8.

Aprovechando que la herramienta ofrece la posibilidad de restringir los tipos de eventos a monitorizar, decidimos que únicamente se registraran los siguientes tipos: creación de ficheros o directorios, modificación de ficheros y borrado de ficheros o directorios. Gracias a esta opción, el procesamiento posterior del archivo de registro se simplificó al evitar el registro de eventos "inocuos", como la apertura de un archivo para leer su contenido o el cierre de un archivo que estaba abierto en modo de solo lectura.

Memoria

Para volcar la memoria de la tableta utilizada para las pruebas empleamos la herramienta Samsung Upload Client [184] desarrollada por B. Kerler. Esta herramienta se basa en el trabajo de Nitay Artenstein y Gilad Goldman [16], quienes descubrieron que podían ejecutar código arbitrario en el Bootloader de Samsung y desarrollaron una herramienta (sboot_dump [188]) para volcar la memoria del dispositivo. El único requisito para utilizar la herramienta es habilitar en el dispositivo la función de depuración llamada `S-Boot Upload Mode`. Para ello, es necesario marcar `#9900` y, en la pantalla emergente, establecer la opción `Nivel de depuración` a `Alto`. Una vez activada, el procedimiento para volcar la memoria es el siguiente:

1. Forzar el reinicio del dispositivo pulsando los botones "Encendido" + "Bajar volumen"[1].

2. En cuanto se apague la pantalla, pulsar los botones "Encendido" + "Subir volumen" hasta que el dispositivo entre en el modo `S-Boot Upload Mode`.

3. Conectar el dispositivo a un ordenador.

4. Ejecutar la herramienta SUC (Samsung Upload Client) con las opciones deseadas para iniciar el volcado. El volcado puede ser de un rango específico, de particiones específicas o de toda la memoria.

La principal ventaja de este método es que no requiere *rootear* el dispositivo ni reemplazar el *kernel*. Además, una vez habilitada la opción de depuración, el proceso de volcado de memoria puede iniciarse con el dispositivo bloqueado. Por supuesto, existen otras opciones para volcar la memoria de un dispositivo Android, como LiME [204, 205], AMExtractor [249] o AMD [250].

LiME es un módulo cargable del *kernel* que vuelca la RAM a un archivo cuando se carga. La principal dificultad para usarlo en Android es que requiere desbloquear el cargador de arranque del dispositivo, así como recompilar el *kernel* para permitir la carga de módulos, ya que esta opción está desactivada por defecto.

AMExtractor hace uso del dispositivo virtual `/dev/kmem`, evitando así la necesidad de cargar un módulo en el *kernel*. No obstante, requiere privilegios de superusuario para poder acceder a dicho dispositivo virtual. Ahora bien, la principal limitación de esta herramienta es que el dispositivo `/dev/kmem` no está disponible en arquitectura ARM de 64 bits (AArch64). Por lo tanto, esta opción será válida dependiendo del dispositivo Android que se esté utilizando.

La herramienta AMD aprovecha los protocolos de actualización del *firmware* para volcar la memoria de un dispositivo Android. Las principales ventajas son que no es necesario *rootear* el dispositivo, ni utilizar un *kernel* personalizado, ni habilitar ninguna opción de depuración. Además, esta herramienta funciona incluso si el dispositivo está bloqueado y el proceso de volcado puede iniciarse sin reiniciar el dispositivo. El único inconveniente es el limitado número de dispositivos que soporta, ya que actualmente solo está preparada para funcionar con ciertos modelos de Samsung y de LG.

[1] Los botones a pulsar pueden variar de un dispositivo a otro.

Para volcar la memoria de las dos instancias de AVD, en lugar de utilizar una herramienta de terceros, hicimos uso de la funcionalidad que proporciona Windows 11 para esta tarea. En concreto, los pasos a realizar para obtener un volcado de memoria de un proceso específico son los siguientes: abrir el Administrador de Tareas de Windows en la pestaña Detalles, buscar el proceso correspondiente y seleccionar la opción "Crear archivo de volcado" del menú contextual. En este caso, el nombre del proceso que ejecutaba los emuladores era `qemu-system-x86_64.exe`.

3.2.4. Navegación

A partir del esquema presentado en la Sección 2.2.3, la sesión de navegación diseñada para ejecutarse en los tres entornos de Android fue la siguiente:

1. Entrar en https://www.youtube.com y buscar `kernel bugs`. Reproducir el vídeo con el título `Syzbot and the Tale of Thousand Kernel Bugs - Dmitry Vyukov, Google`. Después de 15 segundos, pausar el vídeo.

2. Abrir una nueva pestaña y visitar https://github.com.

3. Abrir una nueva pestaña y acceder a https://file-examples.com/index.php/sample-audio-files/sample-mp3-download/. Hacer clic en el botón "Download sample MP3 file" correspondiente a la opción de 700 KB. Pausar el audio después de 5 segundos.
   A diferencia de la sesión de navegación utilizada para los sistemas de escritorio, en este caso no se pudo utilizar un fichero PDF debido a que ninguno de los navegadores probados en Android permitieron su previsualización. La única opción que ofrecían era la descarga del fichero para abrirlo con otra aplicación, lo cual no era una opción válida para esta situación. Por lo tanto, en este caso se decidió utilizar un fichero MP3, el cual sí podía ser reproducido directamente.

4. Abrir una nueva pestaña y escribir myurl.com en la barra de direcciones. Sin acceder a este sitio web, borrar la URL escrita.

5. Acceder a https://reddit.com/login. Iniciar sesión utilizando las credenciales almacenadas en el llavero del navegador. En concreto, usar las credenciales del usuario `Test9861`. Es importante señalar que solo se utilizó la primera de las

tres credenciales que aparecen en la Tabla 3.7.
Cuando se utilizó el navegador Tor Browser, este paso no pudo completarse con éxito en ninguna de las ejecuciones. Esto se debe a que este navegador no permitió el uso de las credenciales almacenadas. Por lo tanto, cuando se estaba probando Tor Browser, simplemente se accedió al sitio web `https://reddit.com/login` y no se realizó ninguna otra acción. Discutiremos este comportamiento extraño (permitió añadir entradas pero no utilizarlas) en la Sección 3.2.8.

6. Abrir una nueva pestaña e ir a `https://www.google.com/gmail`. Pulsar en el botón de `Iniciar sesión` e introducir `virtual112233@gmail.com` como nombre de usuario y `@thisis4testing1` como contraseña. El inicio de sesión fallará.

3.2.5. Adquisición de evidencias

La adquisición de los cambios realizados en el sistema de ficheros se realizó siguiendo los pasos descritos en la Sección 2.2.4 con la herramienta seleccionada en la Sección 3.2.3.

Para obtener cada uno de los cuatro volcados descritos en la Sección 2.2.4 se ejecutaron los siguientes pasos:

T1. Encender el dispositivo, lanzar el navegador en modo privado, realizar la sesión de navegación y volcar la RAM.

T2. Encender el dispositivo, lanzar el navegador en modo incógnito, realizar la sesión de navegación, *cerrar el navegador, esperar un minuto* y volcar la RAM.

T3. Encender el dispositivo, lanzar el navegador en modo incógnito, realizar la sesión de navegación, *cerrar el navegador, reiniciar el dispositivo* y volcar la RAM.

T4. Encender el dispositivo, lanzar el navegador en modo incógnito, realizar la sesión de navegación, *cerrar el navegador, apagar el dispositivo, esperar 10 segundos, encender el dispositivo y, una vez iniciado*, volcar la RAM.

3.2.6. Análisis

La primera parte de esta fase consistió en analizar los cambios en el sistema de ficheros. Dada la sesión de navegación descrita en la Sección 3.2.4, el conjunto de

palabras clave buscadas en los archivos indicados por la herramienta de monitorización, así como en las carpetas que contenían los perfiles, fueron las siguientes:

1. `kernel bugs`
2. `kernel%20bugs`
3. `kernel+bugs`
4. `kernel%2Bbugs`
5. `Syzbot and the Tale of Thousand Kernel Bugs - Dmitry Vyukov, Google`
6. `_gh_sess=`
7. `_octo=`
8. `myurl.com`
9. `Test9861`
10. `oneRandomPassword41+`
11. `Test9862`
12. `oneRandomPassword42+`
13. `Test99172`
14. `oneRandomPassword46+`
15. `virtual112233@gmail.com`
16. `virtual112233%40gmail.com`
17. `@thisis4testing1`
18. `%40thisis4testing1`

Debido a que varias de las actividades de la sesión de navegación son las mismas que las realizadas cuando se probó la versión de escritorio de los navegadores, hay algunos términos de búsqueda que son los mismos en ambos casos. Los cuatro primeros términos están relacionados con la búsqueda realizada en youtube.com. Los números 2 y 4 son iguales a los números 1 y 3 pero sustituyendo, respectivamente, el espacio en blanco y el símbolo '+' por sus correspondientes códigos HTML. El número 5 corresponde al título completo del vídeo reproducido. Los números 6 y 7 corresponden al nombre de las *cookies* establecidas por el sitio web github.com. El número 8 es la URL que se escribió en la barra de direcciones, pero que no se llegó a visitar. Los elementos 9, 11 y 13 corresponden a los nombres de usuario almacenados en el llavero del navegador. Los puntos 10, 12 y 14 son las contraseñas correspondientes a cada uno de los nombres de usuario anteriores. El número 15 es la dirección de correo electrónico introducida para iniciar sesión en google.com/gmail y, el 16, es la misma dirección pero con la '@' sustituida por su código HTML. El número 17 corresponde a la contraseña introducida en Gmail y, el último elemento, es la misma contraseña pero con la '@' sustituida por su código HTML.

Una vez finalizado el análisis del sistema de archivos, la siguiente tarea fue la extracción de información de los volcados de memoria. Esta tarea estuvo compuesta de dos partes. Para la primera parte se desarrolló un *script* que buscara por las mismas palabras clave que las utilizadas para el análisis del sistema de archivos. La salida de este *script* fue un archivo CSV que indicara el número de ocurrencias encontradas para cada uno de los términos de búsqueda en los diferentes volcados. Para la segunda parte se implementó un programa que, dado un fichero, indicara si ese fichero se encontraba íntegro en el volcado o no. En este caso, los ficheros a buscar fueron los siguientes: el fichero MP3 previsualizado en el navegador y los ficheros que almacenaban el contenido del llavero de los diferentes navegadores.

3.2.7. Resultados

El aplicar la metodología descrita exigió repetir la sesión de navegación varias veces y capturar evidencias en momentos distintos en cada ejecución. Esto implicó que había que tomar precauciones a la hora de evitar contaminar los resultados de una ejecución con artefactos que quedaron de una ejecución previa. Hay varias formas de evitar este problema, desde una solución radical y que consume mucho tiempo hasta una más

sencilla y rápida.

La primera de las soluciones es la más radical y consiste en empezar cada ejecución con un dispositivo totalmente limpio. Para ello, hay que realizar los siguientes pasos antes de iniciar una nueva prueba:

1. Si se va a utilizar un dispositivo físico, hay que restablecerlo a la configuración de fábrica. Sin embargo, si se va a utilizar un emulador, simplemente hay que crear un nuevo dispositivo virtual.
2. Instalar y configurar el navegador a probar.
3. Apagar el dispositivo o emulador por completo. Si se está utilizando un dispositivo físico es importante esperar al menos un minuto antes de volver a encenderlo para garantizar que se empieza con una RAM "limpia".
4. Encender el dispositivo o emulador y empezar con la prueba correspondiente.

La segunda solución consiste en desinstalar y volver a instalar el navegador a probar. En Android, cuando se desinstala una aplicación, automáticamente se eliminan todos los datos del usuario. Por lo tanto, cuando se vuelve a instalar, no hay ningún rastro de la configuración anterior. A continuación, se enumeran los pasos a realizar antes de iniciar una nueva ejecución con esta solución:

1. Desinstalar el navegador.
2. Volver a instalar el navegador.
3. Configurar el navegador.
4. Apagar el dispositivo o emulador por completo. Al igual que antes, si se está utilizando un dispositivo físico es importante esperar al menos un minuto antes de volver a encenderlo para garantizar que se empieza con una RAM "limpia".
5. Encender el dispositivo o emulador y empezar con la prueba correspondiente.

La tercera y última opción consiste en utilizar las opciones de limpieza incluidas en los propios navegadores. Esta opción es la más sencilla y la que requiere menos tiempo de preparación de las tres. En este caso, los pasos a realizar antes de una nueva ejecución son los siguientes:

1. Limpiar las *cookies*, el historial y la caché utilizando la opción disponible en las preferencias del navegador. Si el navegador probado en la ejecución anterior es distinto al que se va a probar a continuación, se debe hacer esta limpieza en ambos navegadores (el anterior y el siguiente).

2. Apagar el dispositivo o emulador por completo. De nuevo, si se está utilizando un dispositivo físico es importante esperar al menos un minuto antes de volver a encenderlo para garantizar que se empieza con una RAM "limpia".

3. Encender el dispositivo o emulador y empezar con la prueba correspondiente.

Después de probar las tres soluciones con los distintos navegadores, no encontramos ninguna diferencia en los resultados obtenidos. Los resultados presentados en las subsecciones siguientes son el resultado de ejecutar varias veces las mismas pruebas con el fin de corroborar los resultados. Para ser lo más eficientes posibles en la preparación del entorno, utilizamos un híbrido de dos de las soluciones presentadas. Se utilizó la primera solución cada vez que empezábamos las pruebas de un nuevo navegador. Por ejemplo, si acabábamos de probar Brave e íbamos a probar Tor Browser, entonces restaurábamos los dispositivos por completo. Sin embargo, cuando estábamos haciendo varias pruebas de un mismo navegador, entonces utilizábamos la tercera opción. Es decir, utilizábamos las operaciones de limpieza que proporcionaba el propio navegador.

Resultados del análisis del sistema de ficheros

Tras examinar todos los archivos indicados por la herramienta de monitorización utilizada, y también tras repetir las búsquedas en los directorios donde los navegadores almacenaban los perfiles de usuario, no se encontró ningún artefacto que revelara las acciones efectuadas en modo privado.

Un detalle descubierto al procesar los directorios que contenían los perfiles de los navegadores fue que tanto Google Chrome como Brave dejaron los nombres de usuario (no las contraseñas) almacenados en el llavero del navegador en texto claro en el disco. En concreto, en el fichero `Login Data` que se encontraba en el directorio que contenía el perfil de usuario. Sin embargo, al repetir estas búsquedas en los directorios de Mozilla Firefox y Tor Browser no se obtuvo ningún resultado. Este punto no está relacionado

con el funcionamiento del modo privado, pero es una diferencia que se encontró al analizar los cambios en el sistema de archivos.

Resultados del análisis de los volcados de memoria

Las Tablas 3.9, 3.10 y 3.11 contienen los resultados de procesar los volcados obtenidos con los navegadores en los escenarios T1, T2 y T3, respectivamente.

La primera parte de las tablas (Búsquedas por palabra clave) contiene el número de coincidencias para cada término de búsqueda descrito anteriormente. La segunda parte de las tablas (Recuperación de ficheros) indica si fue posible recuperar el archivo MP3 previsualizado así como los archivos que contenían los llaveros del navegador. En el caso de Google Chrome y Brave, el nombre del archivo que contenía los nombres de usuario y contraseñas almacenados en el navegador era `Login Data` y, en el caso de Mozilla Firefox y Tor Browser, era `logins2.sqlite`.

Los resultados obtenidos al analizar los volcados creados después de haber tenido apagado los dispositivos durante 10 segundos (escenario T4) revelaron que no fue posible recuperar ningún artefacto en ninguno de los casos. La única excepción fue cuando se ejecutó Google Chrome en el emulador con Android 9. En este caso, se encontraron dos coincidencias de cada uno de los nombres de usuario guardados en el llavero del navegador. O dicho de otro modo, se hallaron dos ocurrencias de `Test9861`, dos de `Test9862`, y otras dos de `Test99172`.

En las Tablas 3.12 y 3.13 se muestran agrupadas por navegador y por entorno todas las coincidencias obtenidas en todos los volcados. Estas tablas resumen permiten ver como de extrapolables son los resultados obtenidos con los emuladores y también permiten determinar que navegador dejó más artefactos en memoria.

3.2.8. Discusión

En las subsecciones siguientes se analizan los resultados obtenidos con los cuatro navegadores en las distintas situaciones.

Sistema de ficheros

El objetivo principal de este caso de uso de la metodología era determinar qué información se podía recuperar de un dispositivo Android tras utilizar un navegador

T1	Chrome			Brave			Firefox			Tor Browser		
	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13
Búsquedas por palabras clave												
kernel bugs	14	18	10	9	11	23	14	15	12	13	13	11
kernel%20bugs	4	4	-	6	-	11	5	5	5	5	-	5
kernel+bugs	40	42	130	35	31	46	15	14	11	15	23	18
kernel%2Bbugs	47	49	99	64	46	48	34	33	28	24	20	22
Syzbot and the Tale of...	27	26	23	24	25	27	16	13	14	14	14	13
_gh_sess=	-	1	1	-	1	3	2	2	2	2	2	2
_octo=	6	5	6	7	5	5	4	3	2	4	3	2
myurl.com	8	6	23	3	4	19	4	-	19	3	-	21
Test9861	39	38	44	29	74	73	27	34	44	-	-	-
oneRandomPassword41+	1	1	1	-	1	1	2	5	6	-	-	-
Test9862	2	2	2	2	2	2	4	9	17	-	-	-
oneRandomPassword42+	-	-	-	-	-	-	5	5	4	-	-	-
Test99172	2	2	2	2	2	2	-	-	-	-	-	-
oneRandomPassword46+	-	-	-	-	-	-	-	-	-	-	-	-
virtual112233@gmail.com	59	52	68	46	48	86	59	14	24	35	10	31
virtual112233%40gmail.com	6	5	6	5	7	7	9	9	12	10	9	10
@thisis4testing1	6	9	6	9	12	9	22	25	23	15	18	19
%40thisis4testing1	2	5	5	5	8	3	3	1	1	3	2	1
Recuperación de ficheros												
Browser keychain	Sí	No	No	No	No	No	No	No	No	No	No	No
file.mp3	No	No	No	Sí	No	No	Sí	Sí	Sí	Sí	Sí	Sí

Tabla 3.9: Resumen del análisis de los volcados de memoria creados con el navegador aún en ejecución (**T1**). La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

T2	Chrome			Brave			Firefox			Tor Browser		
	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13
Búsquedas por palabras clave												
kernel bugs	9	13	18	7	12	10	12	11	3	12	12	13
kernel%20bugs	4	-	1	6	-	-	1	1	1	2	2	5
kernel+bugs	53	36	30	25	27	39	11	6	8	9	12	34
kernel%2Bbugs	56	51	43	69	49	51	28	18	15	7	8	20
Syzbot and the Tale of...	25	26	21	19	26	24	14	10	2	13	9	14
_gh_sess=	-	-	-	-	-	1	-	-	-	1	1	2
_octo=	3	5	3	5	6	6	2	1	1	2	2	2
myurl.com	7	3	18	3	-	19	3	-	19	3	-	21
Test9861	37	42	43	65	39	49	8	6	26	-	-	-
oneRandomPassword41+	1	1	1	-	1	1	2	1	2	-	-	-
Test9862	2	2	2	2	2	2	7	2	2	-	-	-
oneRandomPassword42+	-	-	-	-	-	-	2	2	2	-	-	-
Test99172	2	2	2	2	2	2	-	-	-	-	-	-
oneRandomPassword46+	-	-	-	-	-	-	-	-	-	-	-	-
virtual112233@gmail.com	51	27	27	18	65	42	56	9	4	18	4	11
virtual112233%40gmail.com	3	1	1	12	9	7	3	2	2	7	1	5
@thisis4testing1	10	4	-	9	10	4	13	9	1	11	16	7
%40thisis4testing1	4	2	2	5	4	6	2	1	-	2	1	-
Recuperación de ficheros												
Browser keychain	Sí	No	No	No	No	No	No	No	No	No	No	No
file.mp3	No	No	No	Sí	No	No	No	No	No	No	No	Sí

Tabla 3.10: Resumen del análisis de los volcados de memoria creados un minuto después de cerrar el navegador (**T2**). La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

T3	Chrome			Brave			Firefox			Tor Browser		
	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13	Samsung	Android 9	Android 13
Búsquedas por palabras clave												
kernel bugs	2	5	-	-	5	-	2	2	-	-	3	-
kernel%20bugs	1	4	-	3	4	-	-	-	-	-	-	-
kernel+bugs	3	20	1	2	16	-	-	2	1	-	-	-
kernel%2Bbugs	1	24	2	1	15	1	1	5	-	-	1	-
Syzbot and the Tale of...	7	5	-	-	6	-	-	1	-	-	1	-
_gh_sess=	-	-	-	-	1	-	-	-	-	-	-	-
_octo=	1	-	-	-	1	-	-	-	-	-	-	-
myurl.com	2	-	-	-	-	-	-	-	-	-	-	-
Test9861	9	8	-	1	4	-	2	5	-	-	-	-
oneRandomPassword41+	-	1	-	-	-	-	-	2	-	-	-	-
Test9862	-	4	-	-	2	-	1	1	-	-	-	-
oneRandomPassword42+	-	-	-	-	-	-	-	1	-	-	-	-
Test99172	-	4	-	-	2	-	-	-	-	-	-	-
oneRandomPassword46+	-	-	-	-	-	-	-	-	-	-	-	-
virtual112233@gmail.com	1	2	-	3	-	-	2	-	-	1	1	-
virtual112233%40gmail.com	-	-	-	-	5	-	2	-	-	4	-	-
@thisis4testing1	1	-	-	-	-	-	4	2	-	-	1	-
%40thisis4testing1	-	-	-	-	-	-	-	-	-	-	-	-
Recuperación de ficheros												
Browser keychain	No	No	No	No	No	No	No	No	No	No	No	No
file.mp3	No	No	No	No	No	No	No	No	No	No	No	No

Tabla 3.11: Resumen del análisis de los volcados de memoria creados tras el reinicio del dispositivo (**T3**). La primera parte de la tabla (Búsquedas por palabra clave) muestra el número de coincidencias para cada uno de los términos de búsqueda. La segunda parte (Recuperación de ficheros) muestra si fue posible o no recuperar esos archivos. Para facilitar la lectura de la tabla, en lugar de poner un cero en los casos donde no se encontró ninguna coincidencia, se utilizó un guion.

Navegador web	Coincidencias totales
Google Chrome	1756
Brave	1583
Mozilla Firefox	898
Tor Browser	713

Tabla 3.12: Número total de coincidencias obtenidas para cada navegador teniendo en cuenta todos los volcados obtenidos en los tres entornos.

Entorno	Coincidencias totales
Tablet	1612
Emulator Android 9	1579
Emulator Android 13	1759

Tabla 3.13: Número total de coincidencias obtenidas para cada entorno teniendo en cuenta los cuatro navegadores probados.

en modo privado. Los resultados muestran que, de los navegadores probados, ninguno escribió información relacionada con la navegación en el sistema de ficheros. Los únicos datos encontrados fueron los nombres de usuario (no las contraseñas) almacenados en los llaveros de los navegadores Google Chrome y Brave. Mozilla Firefox y Tor Browser no almacenaron los nombres de usuario en texto claro, por lo que la búsqueda en los directorios de estos navegadores no devolvió ningún resultado.

Siguiendo con los llaveros integrados en los navegadores probados, cabe destacar el diferente comportamiento de cada uno de ellos. Pero, antes de eso, es importante recordar las acciones que puede permitir un navegador a la hora de administrar y usar el llavero. Estas acciones, mostradas en la Sección 3.1.7, son las siguientes:

1. Añadir manualmente nuevas entradas desde las preferencias.
2. Guardar la información de inicio de sesión en el momento que uno se autentifica en un sitio web utilizando el modo de navegación normal.
3. Crear una nueva entrada tras haber iniciado sesión en un sitio web utilizando el modo de navegación privada.
4. Utilizar las entradas almacenadas desde el modo normal.
5. Hacer uso las entradas guardadas desde el modo privado.

Google Chrome y Brave permitieron realizar las acciones 2, 4 y 5. Es decir, la única forma de añadir entradas fue accediendo al propio sitio web e iniciando sesión sin emplear el modo incógnito. Además, todas las entradas almacenadas podían ser utilizadas desde ambos modos de navegación. Curiosamente, la versión de escritorio de Google Chrome permitió realizar a mayores la acción número 1. Una medida adicional que incorporaron en las versiones móviles de Google Chrome y de Brave fue que, si el dispositivo no dispone de ningún mecanismo de bloqueo de pantalla, no permiten que las contraseñas se muestren en texto claro en el gestor de contraseñas.

En el caso del navegador Mozilla Firefox, este permitió realizar todas las acciones. A modo recordatorio, este es el mismo comportamiento que la versión de escritorio. Tal y como se comentó en la Sección 3.1.7, el equipo de Mozilla debería reconsiderar el hecho de permitir la opción número 3, ya que están permitiendo que quede un registro en disco de los sitios web visitados desde el modo privado. Otro problema de la versión móvil de Mozilla Firefox es que permitió mostrar las contraseñas guardadas incluso sin tener configurado un mecanismo de bloqueo de pantalla. Cuando no hay un PIN o una contraseña de desbloqueo configurados, Mozilla Firefox mostró una advertencia y aconsejó configurar uno para evitar que cualquiera pudiera ver las contraseñas. El único problema es que este mensaje solo apareció la primera vez que se accedió al gestor de contraseñas.

Por último, Tor Browser solo permitió realizar la acción número 1. Es realmente extraño que permitiera añadir entradas manualmente al llavero si no iba a permitir utilizarlas en ninguna situación. Hemos verificado este comportamiento en la versión de escritorio de Tor Browser usando un PC con Windows y otro con Linux. En ambos sistemas operativos, descubrimos que el navegador no permite añadir manualmente entradas al llavero y tampoco pregunta si se quiere guardar la información de acceso cuando se entraba en un sitio web. En otras palabras, en la versión de escritorio, el llavero de Tor Browser está completamente deshabilitado. Por lo tanto, es un poco confuso que la versión de Android permitiera añadir manualmente entradas al llavero. Otro detalle de la versión móvil es que cuando no se tiene configurado ningún mecanismo de bloqueo y se accede al llavero para ver las contraseñas, Tor Browser muestra la misma advertencia que Mozilla Firefox, indicando que se debería configurar uno para evitar que cualquiera pueda ver las contraseñas.

Volcados de memoria creados en T1

Los navegadores probados no almacenaron en el sistema de ficheros ninguna información relacionada con la navegación realizada en modo privado. Sin embargo, como puede verse en la sección de resultados, el análisis de la de memoria sí permitió recuperar un gran número de artefactos relacionados con la navegación.

En la situación T1, los volcados fueron creados una vez finalizada toda la sesión de navegación y con el navegador aún en ejecución. Estos volcados permiten establecer un punto de partida con el que comparar el resto de volcados, los cuales fueron creados cuando el navegador ya no estaba en ejecución. En otras palabras, los volcados creados en T1 representan el mejor escenario a la hora de recuperar las acciones realizadas en modo privado.

La primera actividad efectuada fue acceder a YouTube y ver un vídeo. Independientemente del navegador utilizado, fue posible recuperar las palabras clave introducidas en el campo de búsqueda, así como el título completo del vídeo reproducido. Aunque se utilizó YouTube, al igual que en el caso de uso de Linux presentado en la Sección 3.1, esta prueba muestra que es posible recuperar los términos introducidos en un campo de búsqueda, así como determinar qué resultado fue el seleccionado, independientemente del sitio web visitado.

La segunda actividad consistió en entrar en `https://github.com` para, en la fase de análisis, intentar recuperar las *cookies* creadas. En concreto, cuando se accedió a GitHub, se almacenaron tres *cookies* en el navegador: `_gh_sess`, `_octo`, y `logged_in`. En la fase de análisis, se decidió buscar únicamente por las dos primeras *cookies*. El problema con la tercera *cookie* era el nombre. Era muy genérico y generaba un gran número de falsos positivos al buscar por ese nombre. Esto no significa que el valor de la tercera *cookie* no estuviera en el volcado. Simplemente requeriría un análisis más específico como, por ejemplo, utilizando expresiones regulares para recuperar su valor. En cuanto a la primera *cookie*, no fue posible recuperarla cuando se utilizó Google Chrome o Brave en la tableta, pero fue posible recuperarla en todos los demás casos. En cuanto a la segunda *cookie*, fue posible recuperarla en todas las situaciones.

La tercera actividad consistió en escuchar un archivo MP3 directamente en el navegador. Como puede verse en la Tabla 3.9, fue posible recuperar el archivo cuando se utilizó Mozilla Firefox o Tor Browser. Con Google Chrome, el archivo no estaba intacto en los volcados obtenidos y, con Brave, solo fue posible recuperarlo intacto

cuando se ejecutaba en la tableta, no en los emuladores. Aunque en este caso se utilizó un archivo MP3, esta prueba demuestra que es posible recuperar, en algunos casos, archivos previsualizados directamente en el navegador que no fueron guardados en disco.

La cuarta actividad fue escribir una URL en la barra de direcciones pero no acceder a ella. Curiosamente, fue posible recuperar esta URL en todos los casos excepto cuando se ejecutaron Mozilla Firefox y Tor Browser en el emulador con Android 9. Al igual que se comentó en el caso de uso de Linux, las dos posibles razones de este comportamiento son las siguientes: 1) el navegador construye la URL completa (añadiendo `https://`, por ejemplo) en segundo plano y 2) el navegador prepara la URL necesaria para utilizar las palabras introducidas como términos de búsqueda en un motor de búsqueda. Por ejemplo, en el caso de Google Chrome, uno de los resultados obtenidos al buscar `myurl.com` en el volcado era la URL que el navegador había construido para buscar esa palabra en el buscador de Google: `https://www.google.com/search?q=myurl.com&client=chrome-mobile&sourceid=chrome-mobile&ie=UTF-8`.

El siguiente paso realizado con los navegadores fue acceder a `https://reddit.com` e iniciar sesión con una de las credenciales que se habían guardado en el llavero. Como se comentó anteriormente, Tor Browser no permite hacer uso de las credenciales guardadas, lo que se refleja perfectamente en los resultados, donde no fue posible recuperar ninguna de las credenciales almacenadas. Como se indica en la Tabla 3.7, se almacenaron tres credenciales en el llavero de cada uno de los navegadores, dos para `https://reddit.com` (Test9861 y Test9862) y una para `https://forums.linuxmint.com` (Test99172). Con los otros tres navegadores que sí permiten el uso del llavero, se utilizaron las credenciales del usuario Test9861 para iniciar sesión en `https://reddit.com`. En el caso de Google Chrome, fue posible recuperar los tres nombres de usuario almacenados en el llavero, pero solo se encontró la contraseña del usuario Test9861 en texto claro. En el caso de Brave, también fue posible recuperar los nombres de usuario. Sin embargo, solo fue posible recuperar la contraseña del usuario que había iniciado sesión cuando Brave se ejecutaba en los emuladores. Por último, en el caso de Mozilla Firefox, se pudieron recuperar los nombres de usuario y las contraseñas guardados para el sitio web `https://reddit.com`, pero no se encontró ni el nombre de usuario ni la contraseña para el sitio web `https://forums.linuxmint.com`. Por lo tanto, parece que Mozilla Firefox descifra del llavero todas las credenciales asociadas

a un sitio web concreto, aunque solo se llegue a utilizar una de ellas.

Continuando con la recuperación de credenciales, la siguiente parte del análisis consistió en determinar si era posible recuperar los archivos que almacenaban los llaveros. En el caso de Google Chrome, se pudo obtener el archivo completo (`Login Data`) cuando se estaba ejecutando en la tableta. Observando el código fuente de Chromium [199] (el proyecto de código abierto en el que se basa Google Chrome), se puede ver que, en el caso de Android, todas las credenciales almacenadas en el llavero están cifradas mediante el algoritmo AES de 128 bits en modo CBC, utilizando `peanuts` como contraseña y `saltysalt` como sal (*salt* en inglés). Por lo tanto, en el caso de Google Chrome, teniendo acceso únicamente al volcado de memoria, fue posible la obtención de todas las credenciales almacenadas en el llavero, incluyendo las dos contraseñas, la de `https://reddit.com` y la de `https://forums.linuxmint.com`, que no estaban en texto claro en el volcado. Brave también está basado en Chromium, por lo que si el archivo `Login Data` se hubiera recuperado intacto en cualquiera de los entornos, también habría sido posible obtener todas las credenciales almacenadas de la misma forma que en Google Chrome. En el caso de Mozilla Firefox y Tor Browser, el fichero que contenía el llavero (`logins2.sqlite`) no estaba completo en ninguno de los volcados de memoria, por lo que obtener toda la información almacenada en el llavero tampoco fue posible en este caso.

La última actividad de la sesión de navegación diseñada consistió en iniciar sesión en Gmail. Independientemente del entorno y del navegador utilizado, se pudieron recuperar tanto el nombre de usuario como la contraseña. Aunque se decidió utilizar específicamente Gmail, esta prueba indica que es posible recuperar la información de inicio de sesión introducida en cualquier sitio web.

Volcados de memoria creados en T2

Estos volcados se crearon un minuto después de cerrar el navegador. Comparando los resultados obtenidos en T1 (Tabla 3.9) con los obtenidos en T2 (Tabla 3.10), no existen grandes diferencias, siendo posible recuperar prácticamente toda la sesión de navegación realizada.

Independientemente del entorno y del navegador, fue posible recuperar, al igual que en T1, las palabras introducidas en el campo de búsqueda de YouTube así como el título del vídeo reproducido. A la hora de recuperar las *cookies* establecidas por el

sitio web https://github.com, sí hay diferencias en los resultados. En T2 solo se pudo recuperar la primera *cookie* (_gh_sess) cuando se ejecutaba Brave en el emulador de Android 13 y en todas las ejecuciones de Tor Browser en los diferentes entornos. Sin embargo, la segunda *cookie* se pudo recuperar en todos los casos. Con respecto a la URL escrita en la barra de direcciones, pero que no fue accedida (myurl.com), esta se pudo recuperar en los mismos casos que en T1 a excepción de que en esta situación no fue posible recuperarla cuando se utilizaba Brave en el emulador con Android 9. Los resultados obtenidos tras buscar por los nombres de usuario y las contraseñas almacenados en los llaveros de los navegadores tras haber iniciado sesión en https://reddit.com fueron exactamente iguales que en T1. Y lo mismo sucede a la hora de recuperar la información de inicio de sesión introducida en Gmail, fue posible recuperarla en todos los casos. Por último, con respecto a la recuperación de ficheros, el resultado de recuperar los archivos que contenían el llavero del navegador fue el mismo que en T1. Sin embargo, a la hora de recuperar el fichero MP3 previsualizado en el navegador, este solo se pudo recuperar en las dos siguientes situaciones: cuando se ejecutaba Brave en la tableta y cuando se utilizaba Tor Browser en el emulador de Android 13.

Volcados de memoria creado en T3

Como era de esperar, la cantidad de información que se pudo recuperar de los volcados creados tras reiniciar los dispositivos (Tabla 3.11) fue significativamente menor que en los casos anteriores, pero no es en absoluto despreciable.

Las palabras introducidas en el campo de búsqueda de YouTube pudieron ser recuperadas en todos los casos salvo cuando se utilizaba Tor Browser en la tableta o en el emulador con Android 13. El título del vídeo reproducido solo pudo ser recuperado en los siguientes casos: con Google Chrome ejecutándose en la tableta o en el emulador de Android 9 y con Brave, Mozilla Firefox y Tor Browser ejecutándose en el emulador de Android 9. La primera *cookie* establecida por https://github.com solo pudo ser recuperada cuando se utilizaba Brave en el emulador de Android 9 y, la segunda *cookie*, cuando se navegaba con Google Chrome en la tableta y con Brave en el emulador de Android 9. La URL introducida en la barra de direcciones, pero no accedida, únicamente pudo ser recuperada en el caso de Google Chrome corriendo en la tableta. El nombre de usuario utilizado para iniciar sesión en https://reddit.com

(`Test9861`) se pudo recuperar en los mismos casos que en T2, salvo que en T3 no fue posible recuperarlo en ninguna de las ejecuciones con el emulador de Android 13. La contraseña de este usuario de `https://reddit.com` solo se pudo recuperar cuando se utilizaban los navegadores Google Chrome y Mozilla Firefox corriendo en el emulador de Android 9. El nombre del otro usuario de `https://reddit.com` almacenado en el llavero (`Test9862`) se pudo recuperar cuando se utilizaba Google Chrome, Brave y Mozilla Firefox en emulador de Android 9 y con Mozilla Firefox ejecutándose en la tableta. Y la contraseña de `Test99172` únicamente pudo ser recuperada en la ejecución de Mozilla Firefox en el emulador de Android 9. Con respecto al tercer usuario almacenado en el llavero para el sitio `https://forums.linuxmint.com`, este solo se pudo recuperar con Google Chrome y Brave ejecutándose en el emulador de Android 9. Y la contraseña de este tercer usuario no se encontró en ningún escenario, al igual que en T1 y en T2. Con respecto a la dirección de correo utilizada para iniciar sesión en Gmail, esta no pudo ser recuperada ni cuando se hacía uso del emulador de Android 13 ni cuando se ejecutaba Mozilla Firefox en el emulador de Android 9. Y la contraseña introducida solo fue posible recuperarla cuando se utilizaba la tableta con Google Chrome y Mozilla Firefox y cuando se ejecutaba el emulador de Android 9 con Mozilla Firefox y Tor Browser. Finalmente, y a diferencia de T1 y T2, en T3 no se pudo recuperar ningún fichero en ninguno de los entornos.

Volcados de memoria creados en T4

Tal y como se comentó en la Sección 3.2.7, tras haber tenido apagado el dispositivo o emulador durante 10 segundos, la única situación donde se pudo recuperar algo fue cuando se ejecutaba Google Chrome en el emulador con Android 9. Esta información no proporcionó ninguna pista de las actividades realizadas en modo privado y tampoco fue particularmente útil, dado que ya se encontraba almacenada en texto claro en el disco.

Mejor entorno y mejor navegador

Gracias a las tablas resumen (Tablas 3.12 y 3.13) es posible ver rápidamente los entornos y los navegadores que mejor se comportaron desde el punto de vista de dificultar la recuperación de información.

En cuanto al navegador, Tor Browser fue el que dejó menos artefactos en memoria. Sin embargo, no es una comparación justa, ya que este navegador tiene el llavero deshabilitado, por lo que no fue posible realizar de forma completa la sesión de navegación. Por lo tanto, en una situación más habitual, el navegador más privado fue Mozilla Firefox. Tomando a este de referencia, Brave permitió que se recuperara un 76,28 % más de información y, Google Chrome, un 95,55 % más.

En cuanto a los entornos, está claro que el entorno más real fue el de la tableta. Tomando a este como base, el emulador de Android 9 permitió que se recuperara un 2,05 % menos de información y, el emulador de Android 13, un 9,12 % más. Esto muestra que el uso de emuladores puede dar resultados sesgados, lo que impide conocer con un alto grado de certeza el verdadero comportamiento del modo privado.

Escritorio vs. móvil

Tras haber analizado diversos navegadores ejecutándose tanto en sistemas de escritorio como en sistemas móviles, uno se podría preguntar cuál de las dos plataformas es preferible para sacarle el máximo partido al modo de navegación privada.

Desde el punto de vista de la adquisición de las evidencias, es mucho más difícil obtener información de un dispositivo móvil. En un sistema de escritorio es posible desmontar el disco duro y analizarlo externamente desde otro equipo. Además, como la configuración predeterminada de muchos sistemas operativos es no cifrar el sistema de ficheros, es posible acceder a toda la información almacenada sin muchos problemas. En el caso de un dispositivo móvil, el primer problema es la incapacidad de desmontar el disco del dispositivo, ya que suele estar soldado en la propia placa base. Pero aunque se consiguiera extraer, sería totalmente inútil, ya que el sistema de ficheros entero está cifrado. Y, en este caso, no es una elección del usuario, sino que viene habilitado por defecto desde la versión 6.0 de Android [82]. En el caso de la RAM, también es mucho más difícil obtener un volcado de un dispositivo móvil. En el caso de un ordenador, existen varias herramientas (muchas incluso disponen de una interfaz gráfica) para distintos sistemas operativos que permiten obtener una copia del contenido de la memoria con bastante facilidad. El único requisito suele ser tener permisos de administrador. Sin embargo, en el caso de un dispositivo móvil, es increíblemente complicado. No existe una herramienta universal que permita obtener un volcado de forma fácil. Algunas de las soluciones mencionadas en este trabajo implican modificar

el *kernel*, lo cual no es una tarea sencilla en los dispositivos modernos. En nuestro caso, hicimos uso de las tareas de depuración propias de Samsung, pero esto no es aplicable a otros fabricantes.

Desde el punto de vista de los resultados obtenidos, es un poco más complicado hacer una comparación directa entre ambas plataformas. Con el fin de poder compararlas a grandes rasgos, la comparación se va a limitar a los entornos más reales y populares (el entorno A en el caso de sistemas de escritorio y el entorno de la tableta en el caso de sistemas móviles) y a los navegadores Mozilla Firefox y Google Chrome (por ser los únicos que fueron probados en ambas plataformas). Los resultados obtenidos del análisis del sistema de ficheros revelaron que no fue posible recuperar ninguna información relativa a la navegación realizada en ninguna de las plataformas. Sin embargo, sí fue posible recuperar de los volcados de memoria prácticamente todas las actividades realizadas en ambas plataformas. Curiosamente, solo fue posible obtener información después de haber reiniciado los dispositivos en el caso de la tableta, no en el caso del ordenador (a modo recordatorio, la comparación se está limitando al entorno A). Si uno se centra solo en el número de artefactos que se pudieron recuperar, y teniendo en cuenta que las sesiones de navegación realizadas en cada plataforma no fueron idénticas, entonces hay que reconocer que fue el sistema de escritorio el que dejó menos artefactos en memoria, un 22 % menos aproximadamente. De nuevo, es importante recalcar que no es una métrica exacta y que las sesiones de navegación, aunque siguieron el mismo esquema, no fueron exactamente iguales.

A la hora de escoger el entorno más privado, uno no debería darle una importancia excesiva al número de artefactos en memoria. En este caso, uno debe guiarse por las dificultades de adquirir información de un dispositivo. Y, en este caso, es evidente que es mucho más difícil realizar un análisis forense de un dispositivo móvil actual.

Las aportaciones más significativas de este capítulo se han publicado y extraído del siguiente artículo:

- Xosé Fernández-Fuentes, Tomás F. Pena y José C. Cabaleiro, "Digital forensic analysis methodology for private browsing: Firefox and Chrome on Linux as a case study", *Computers & Security*, Volume 115, April 2022, 102626. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2022.102626. Contribución: Conceptualización, Metodología, Investigación, Redacción (borrador original).

Capítulo 4

Malware de rescate

El *malware* de rescate, más conocido como *ransomware*, es un tipo de software malicioso que amenaza a las víctimas con la pérdida de todos sus documentos o con el bloqueo completo del equipo hasta que se cumplan las condiciones establecidas por los atacantes. Estas condiciones incluyen siempre el pago de un rescate, y pueden exigir condiciones adicionales, tal y como se verá en algunos de los *ransomware* mencionados más adelante. Por supuesto, los ciberdelincuentes no quieren que los pagos sean fácilmente rastreables, por lo que desde los últimos diez años obligan a las víctimas a abonar el rescate en criptomonedas. Aunque estas pueden ser rastreadas, resulta mucho más difícil de localizar a la persona, o grupo de personas, detrás de una cartera de criptomonedas.

Como era de esperar, no todos los *malware* de rescate operan del mismo modo. En términos generales, se pueden distinguir tres tipos:

- **Bloqueador.** Bloquea el equipo de la víctima y muestra un mensaje de alerta. Para asustar a la víctima y ejercer más presión, los atacantes se suelen hacen pasar por alguna autoridad (como el FBI) y exigen el pago de una multa para desbloquear el equipo. De los tres tipos, este es el menos dañino. Además, las implementaciones no suelen ser muy robustas y suele ser posible desbloquear el equipo sin demasiadas complicaciones.

- ***Cifrador.*** Hace uso de diferentes algoritmos de cifrado para impedir el acceso a los ficheros. En algunos casos, este tipo de *malware* cifra el disco por completo, dejando el equipo completamente inoperativo. En otros casos, cifra solo archivos

que sean de un determinado tipo, como extensiones concretas de documentos o de imágenes.

- **Híbridos.** Combina el funcionamiento de los dos tipos anteriores.

De los tres tipos, el más popular, sobre todo en los últimos años, es con diferencia el *cifrador*. Sin embargo, también es más difícil de operar y de programar que un bloqueador. Esto es debido a que, si el *cifrador* no gestiona apropiadamente las claves utilizadas, el ataque puede ser revertido sin necesidad de pagar el rescate. Es importante recalcar que uno de los mayores incentivos que tiene la víctima para realizar el pago es el desconocimiento de la clave.

Antes de continuar, es muy importante conocer los tipos de cifrado que se suelen utilizar, ya que los *cifradores* habitualmente se clasifican en función del número de claves y de como las utilizan [177]. Lo más habitual es que hagan uso de uno de los tres tipos de cifrado que se describen a continuación:

1. **Cifrado asimétrico**. Se compone de dos claves, una pública y otra privada. La clave pública se utiliza para cifrar los archivos. La clave privada se utiliza para descifrar los archivos y solo la conoce el atacante. Este tipo de cifrado, aunque muy robusto, es lento y costoso desde el punto de vista computacional, sobre todo a medida que aumenta el tamaño de los archivos a cifrar. Algunos ejemplos de los sistemas criptográficos de clave pública más populares a día de hoy son RSA [179] y el intercambio de claves Diffie-Hellman [95].

2. **Cifrado simétrico**. Se compone de una sola clave, la cual se utiliza tanto para cifrar como para descifrar los archivos. Este tipo de cifrado es más rápido y eficiente que el cifrado asimétrico, a costa de ser menos robusto. Algunos ejemplos de este esquema de cifrado son Advanced Encryption Standard (AES) [62] y los cifrados de flujo Salsa20 [27] y ChaCha20 [28] (este es una variante de Salsa20).

3. **Cifrado híbrido**. Combina los dos tipos de cifrado anteriores. En primer lugar, hace uso del cifrado simétrico. Para ello, genera aleatoriamente una clave y la utiliza para cifrar los archivos de la víctima. En segundo lugar, utiliza el cifrado asimétrico para proteger la clave generada. Concretamente, cifra la clave generada de forma aleatoria con la clave pública del atacante. Por lo tanto, para recuperar la clave de cifrado simétrico utilizada se necesita la clave privada del

atacante, la cual está fuera del alcance de la víctima. De este modo, se consigue un cifrado robusto (gracias al cifrado asimétrico) y rápido (gracias al cifrado simétrico).

Los creadores de este tipo de software malicioso suelen utilizar cifrados estandarizados e increíblemente robustos. Por lo tanto, si la clave empleada tiene la suficiente entropía, los esfuerzos de recuperarla utilizando técnicas habituales, como la fuerza bruta, están totalmente descartados. Sin embargo, esto no siempre fue así. En las próximas secciones se hace un repaso a la historia del *malware* de rescate, desde sus orígenes hasta el día de hoy, donde se puede ver como fue evolucionando hasta convertirse en la amenaza cibernética más importante de los últimos años.

4.1. 1989-2010: Origen del *ransomware*

Parece que el *ransomware* es una invención muy reciente y que es un ataque completamente nuevo que pone en jaque las medidas de seguridad existentes. Sin embargo, el primer caso con cierta relevancia es de finales de los ochenta y muchos de los ataques más populares y devastadores que se han visto hasta el día de hoy podrían haber sido evitados siguiendo una serie de consejos. Y lo más curioso es que son consejos que llevan en los libros de seguridad informática desde mediados de los ochenta [79].

4.1.1. El primer caso

En el año 1989 se identificó la primera muestra (*sample* en inglés) de software malicioso que, poco después, se consideraría como una de las primeras muestras de *ransomware*. La creación de dicha muestra se atribuye al Dr. Joseph Popp, un biólogo evolutivo licenciado en Harvard [22, 139]. Este *malware* secuestraba el fichero `AUTOEXEC.BAT` para ejecutarse en cada arranque. Para pasar desapercibido, durante los 89 encendidos siguientes, el *malware* se quedaba en estado durmiente. Sin embargo, en el encendido número 90, este virus cifraba todos los nombres de los ficheros del disco C:, dejando el contenido intacto, y mostraba una nota de rescate, la cual se puede ver en la Figura 4.1. En ella, la empresa PC Cyborg Comportation indicaba que se debía pagar una factura por el alquiler del software, la cual era de 189 USD o 378 USD en función de la licencia escogida. El dinero debía enviarse a un apartado de correos en la Ciudad de Panamá. Una vez enviado, se recibirían las instrucciones

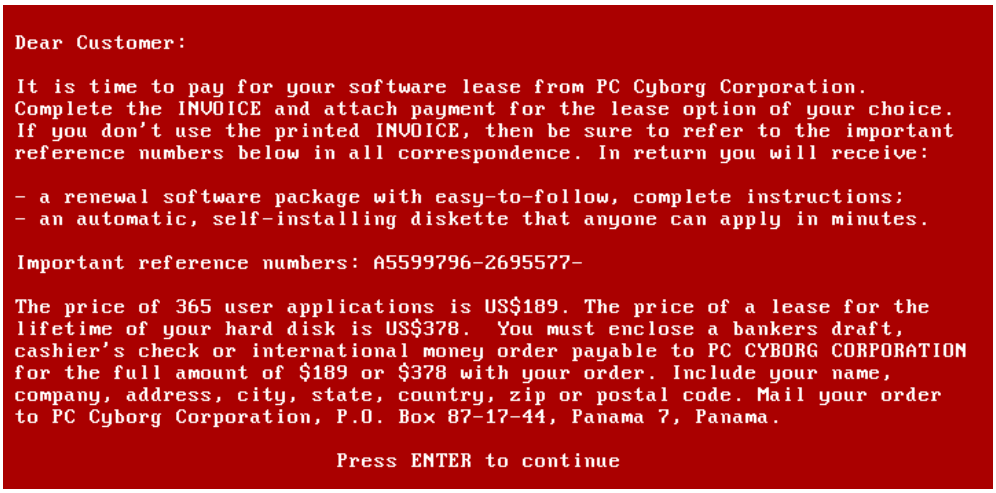

Figura 4.1: Mensaje de rescate mostrado por el *ransomware* AIDS. Fuente: Wikimedia Commons

para restaurar los datos. Para propagar el software malicioso, el Dr. Joseph Popp robó las direcciones de los asistentes de la conferencia sobre el SIDA de la Organización Mundial de la Salud y las direcciones de los suscriptores de la revista PC Business World. A cada uno de ellos les envió un disquete con el virus y se estima que llegó a infectar unos 20 000 equipos. Este primer *ransomware* fue bautizado con el nombre de AIDS (SIDA en inglés), aunque también es conocido como Aids Info Disk o como PC Cyborg Trojan.

El mundo del *ransomware* no vio una gran actividad en los años posteriores debido a la reducida expansión de Internet y la falta de divisas digitales. Este último punto era realmente crítico, ya que los delincuentes tenían problemas para obtener los pagos de las víctimas de forma que no fueran fácilmente rastreables.

4.1.2. Los inicios

Después de más de 10 años sin ningún avance notable, en mayo de 2005 se detectó al que posteriormente se denominó el primer *ransomware* moderno: GP Code, también conocido como GPCoder. Incorporaba un cifrado simétrico hecho a medida, el cual era bastante pobre y fácilmente superable. El principal mecanismo de propagación era mediante ficheros adjuntos en correos electrónicos. Cuando se abrían los adjuntos, el

ransomware cifraba los ficheros que encontraba de Microsoft Office así como diversos ficheros multimedia [101], exigiendo 200 USD de rescate [210].

En el año 2006, el *ransomware* empezó a coger fuerza y apareció Archiveus Trojan, el primer ejemplo en utilizar un cifrado asimétrico [15]. El funcionamiento de Archiveus Trojan consistía en copiar todos los archivos de la carpeta de "Mis Documentos" en un único archivo, cifrarlo utilizando RSA y eliminar los ficheros originales [120]. Una de las curiosidades de esta muestra era la forma de pago exigida para recuperar la clave: le pedía a las víctimas que compraran medicamentos en determinadas tiendas en línea y que enviaran el identificador de la compra [15] al atacante.

En el año 2008, apareció GPcode.AK, una variante de GP Code. En este caso, en lugar de utilizar el cifrado simétrico, hacía uso del cifrado asimétrico RSA con un tamaño de clave de 1024-bit. Esto hizo que esta nueva variante fuera más robusta y más difícil de descifrar [83]. El rescate exigido por los atacantes era un pago de entre 100 USD y 200 USD en e-gold (una moneda de oro digital) o en Liberty Reserve (un servicio de moneda digital centralizada con sede en Costa Rica) [218].

4.1.3. Los bloqueadores

Todas las muestras mencionadas hasta ahora utilizaban algún tipo de cifrado, ya sea para cifrar los nombres de los ficheros o para cifrar el contenido. Sin embargo, muchas de las implementaciones utilizadas eran bastante deficientes y fácilmente reversibles sin necesidad de pagar el rescate. Esto llevó a la aparición de cada vez más muestras que simplemente se dedicaban a bloquear el equipo. Al no cifrar ningún fichero, la implementación del software malicioso era mucho más sencilla. Por supuesto, para desbloquear el equipo era necesario pagar un rescate [79].

El primer *ransomware* en emplear este *modus operandi* fue Trojan.Ransom.C a principios de 2008. Este se dedicaba a falsificar al Centro de Seguridad de Windows indicando que era necesario reactivar la licencia del software de seguridad. Para obtener una renovación de la licencia había que llamar a un número de tarificación adicional. Mientras la "licencia" no fuera reactivada, el ordenador quedaba totalmente bloqueado [186].

En el año 2009 apareció la muestra que realmente le dio un impulso a este tipo de *ransomware*: WinLock [210]. Aunque las primeras infecciones se detectaron en Rusia, no tardó en extenderse por Europa y por Estados Unidos en el año 2010 [176].

Para desbloquear el equipo, WinLock exigía que las víctimas enviaran un SMS con una tarifa con recargo de 10 USD para recibir el código de desbloqueo [244]. En 2010, autoridades rusas detuvieron a varios sospechosos relacionados con WinLock y reportaron que habían ganado unos 16 millones de dólares en tan solo un mes [128].

4.2. 2011-2015: Auge del *ransomware*

El mayor problema de los ciberdelincuentes hasta la fecha era como cobrar los rescates de forma que el dinero no fuera fácilmente rastreable. Un problema que perdió relevancia con la aparición de las criptomonedas.

4.2.1. El nacimiento de Bitcoin

Entre finales del año 2008 y principios de 2009 el panorama del *malware* de rescate dio un cambio enorme debido al nacimiento de bitcoin, la primera criptomoneda. A diferencia de las divisas tradicionales, Bitcoin no tiene un emisor central, sino que es la propia red entre pares la que regula, mediante consenso de la red, las transacciones y la emisión de bitcoins. Todas las transacciones son verificadas por los nodos de la red mediante el uso de criptografía y registradas en un libro de contabilidad público distribuido llamado blockchain [30]. A día de hoy, sigue siendo la criptomoneda más utilizada así como la divisa alternativa más popular [180].

Dada la mayor facilidad para anonimizar los pagos, el número de nuevas muestras de *ransomware* subió a un ritmo sin precedentes año tras año. Así, en el año 2011, nacieron 60 000 nuevas muestras, alentadas por el incremento en popularidad de los servicios de pago anónimos, como las criptomonedas.

4.2.2. Kits de herramientas

Con el fin de facilitar la creación y distribución de *ransomware*, nacieron los primeros kits de herramientas en el año 2012. Uno de los más conocidos fue Citadel, el cual tenía un coste de aproximadamente 3 000 USD [191]. Otro kit también muy popular fue Lyposit, que fue utilizado para crear Reveton. Reveton fue otra de las muestras que impulsó el *ransomware* bloqueador, y se caracterizaba por hacerse pasar por distintas fuerzas de seguridad, indicando que se había descargado material protegido por

derechos de autor o que se había detectado actividad criminal. Por consiguiente, bloqueaban el equipo hasta que se pagara una "multa" [167]. Para asustar a las víctimas, Reveton tomaba el control de las cámaras web de las víctimas y les hacía creer que estaban siendo grabadas.

Con la llegada del año 2013, el número de nuevas muestras siguió en aumento. En julio, aparecieron las primeras versiones de Reveton dirigidas a usuario de macOS pero utilizando un método diferente. En este caso utilizaba JavaScript para abrir numerosos *iframes* mostrando mensajes que indicaban que el usuario había infringido diversas leyes y que por ello se había bloqueado el ordenador. La multa a pagar era de 300 USD mediante una tarjeta de prepago. Curiosamente en este caso, para eliminar Reveton solo había que utilizar la opción de resetear Safari, la cual se podía encontrar en el menú del propio navegador [175].

4.2.3. Un *ransomware* híbrido famoso

En agosto de 2013 apareció uno de los *ransomware* más famosos, CryptoLocker [176]. Fue el primero en exigir el pago del rescate en bitcoin. Tras su ejecución, CryptoLocker hacía uso del cifrado RSA con un tamaño de clave de 2048-bit para secuestrar 67 tipos de archivos diferentes, incluyendo ficheros de AutoCAD, de Microsoft Office, de PhotoShop e incluso certificados de usuario [94]. Además, no se limitaba a cifrar los archivos en el disco duro local, sino que también cifraba cualquier fichero que encontrara en las unidades en red montadas.

Las primeras versiones de CryptoLocker eran distribuidas por correo electrónico e iban dirigidas a profesionales. El contenido de estos correos normalmente trataba sobre presuntas quejas de un cliente o sobre el envío de alguna documentación solicitada [109]. Cuando la víctima descargaba el fichero adjunto, normalmente con extensión `zip`, y lo abría, automáticamente empezaba la descarga y ejecución del *ransomware* [92]. El rescate exigido por los atacantes era de 2 bitcoins (aproximadamente 100 USD en aquel momento), aunque aceptaban otros métodos de pago como Paysafecard o MoneyPak. Además, para presionar a las víctimas, establecían un límite de tres días para pagar el rescate. Variantes posteriores cambiaron este requisito y, si no se pagaba el rescate en esos tres días, aumentaban el precio del rescate. En la Figura 4.2 se puede ver el asistente para enviar el dinero a los atacantes, incluyendo una descripción de qué es bitcoin y un enlace web con una guía de primeros pasos.

A pesar de que era posible eliminar el *ransomware* de los equipos, esto no permitía recuperar el acceso a los ficheros. Debido a que utilizaba criptografía asimétrica, la única solución para recuperar los ficheros era restaurarlos desde una copia de seguridad reciente. Otra opción sería pagar el rescate, aunque hubo usuarios que afirmaron que no obtuvieron la clave de descifrado después de haber realizado el pago. Afortunadamente, este *ransomware* no tenía las capacidades para extenderse por la red e infectar a otros equipos [94]. Aun así, de septiembre de 2013 a mayo de 2014, se estima que el *ransomware* infectó a más de 500 000 víctimas y que el 1,3 % de ellas pagó el rescate. En junio de 2014, gracias a la colaboración de distintas fuerzas policiales, profesionales e investigadores, los servidores de distribución de CryptoLocker fueron desmantelados por completo. Además, dos empresas (FireEye y Fox-IT) encontraron la base de datos con todas las claves y lanzaron un servicio que permitía el descifrado gratuito por parte de todas las víctimas [100].

4.2.4. El millón de dólares

En febrero de 2014 apareció CryptoDefense, un *ransomware* con una implementación bastante débil. Sin embargo, fue la base para que en abril apareciera una versión mejorada, CryptoWall. Esta explotaba una vulnerabilidad en Java y se distribuía a través de anuncios maliciosos en la red. Al parecer, llegó a generar beneficios de más de un millón de dólares [167].

4.2.5. La red Tor

Previamente se mencionó que el nacimiento de Bitcoin hizo que el panorama del *ransomware* diera un cambio enorme, apareciendo un gran número de muestras nuevas que exigían el pago del rescate en criptomonedas. En mayo de 2015, los creadores de *ransomware* vieron que podían anonimizar sus comunicaciones gracias al uso de la red Tor.

La red Tor proporciona una forma de navegar por Internet de forma anónima. Cuando un usuario se conecta a la red Tor, se establece lo que se conoce como un circuito. Un circuito es una serie de *routers* por los que van a viajar los paquetes antes de llegar a su destino final. Los circuitos son rotados cada cierto tiempo y están formados por un mínimo de tres *routers* Tor: un nodo de entrada, uno o más nodos

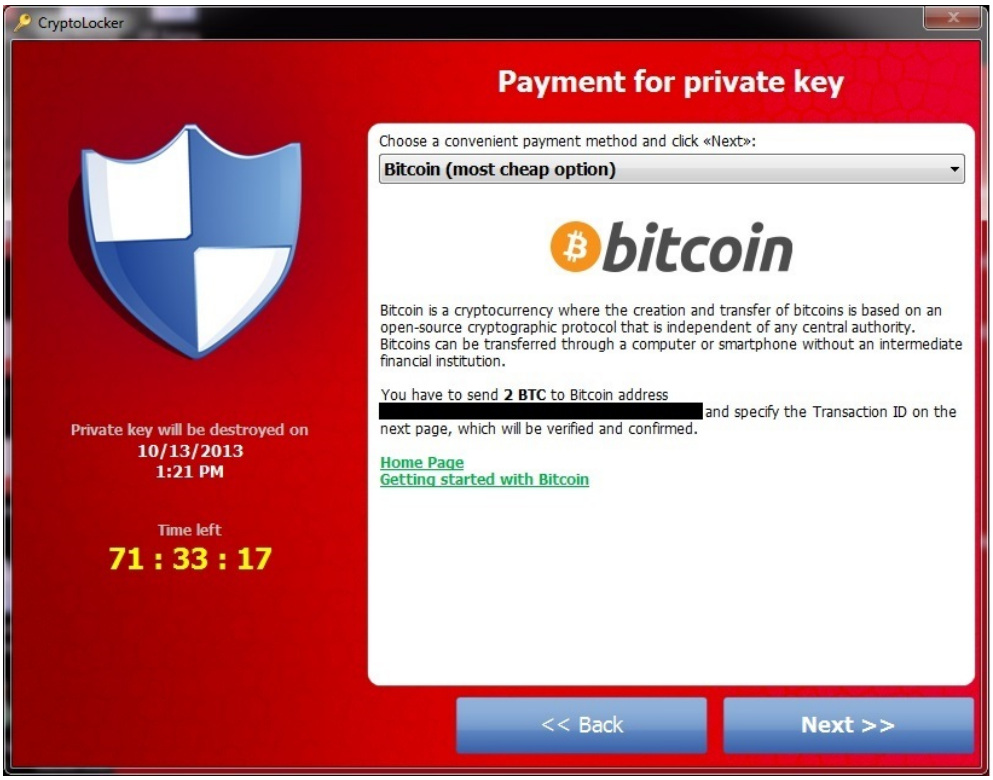

Figura 4.2: Asistente de CryptoLocker para pagar el rescate y obtener la clave necesaria para recuperar el acceso a los ficheros. Fuente: Wikimedia Commons

intermedios y un nodo de salida [133]. Una vez establecido un circuito es posible comenzar el envío de datos. Para ello, los datos son encapsulados en varias capas de cifrado. Primero los datos son cifrados con la clave pública del nodo de salida, luego con las de los nodos intermedios y, por último, con la del nodo de entrada. Esta estrategia de cifrado se conoce como enrutamiento cebolla (*onion routing* en inglés) [81]. En la red Tor, cada nodo solo tiene conexión directa con el nodo inmediatamente anterior y posterior. De esta forma, el nodo de entrada conoce el origen de los datos, pero desconoce su contenido. Los nodos intermedios no tienen forma de determinar si el nodo anterior es el origen de los datos o si simplemente están redirigiendo un paquete. Y, los nodos de salida, conocen el contenido de los datos, pero desconocen por completo

su origen. Gracias al uso de la red Tor, todos los datos enviados por un usuario llegarán a su destino como si procedieran del nodo de salida [230], protegiendo así la ubicación y la IP de lo usuario.

Hacia finales del año 2015, apareció un nuevo *ransomware* llamado TeslaCrypt. Sus métodos de propagación eran distintos a los vistos hasta el momento. Las primeras versiones hacían uso de ingeniería social para engañar a los usuarios para que hicieran clic en un enlace dentro de un correo electrónico de *phising*. En versiones posteriores, hicieron uso de *malvertising*, una técnica que consiste en poner anuncios maliciosos en sitios web legítimos. El objetivo de los atacantes era redirigir a las víctimas a diferentes sitios web de Wordpress controlados por los atacantes. En estos sitios web se encontraban instalados kits de explotación, los cuales intentaban explotar diferentes vulnerabilidades en el equipo de la víctima para conseguir ejecutar el *ransomware* [209]. Por supuesto, TeslaCrypt utilizaba cifrado AES-256 para cifrar los ficheros, RSA-4096 para cifrar la clave AES empleada y un servidor de Mando y Control (C&C por sus siglas en inglés) alojado en la red Tor. Afortunadamente en 2016, los creadores de TeslaCrypt detuvieron sus operaciones e hicieron pública la clave de descifrado maestra para que cualquier víctima pudiera recuperar sus ficheros [144].

4.2.6. El *ransomware* se expande a otras plataformas

A finales de año 2015, se empezaron a ver *ransomware* para otras plataformas. En concreto, en septiembre apareció LockerPin. Estaba dirigido a sistemas Android y se dedicaba a cambiar el PIN de desbloqueo. Para obtener el nuevo código era necesario pagar un rescate de 500 USD [167].

En octubre, se descubrió un *ransomware* para Linux conocido como Linux.Encoder.1. El *ransomware* se dedicaba a cifrar los ficheros del usuario así como ficheros asociados con aplicaciones web que estuvieran alojadas en la máquina.

4.3. 2015-2017: Creación de *ransomware* como negocio

A mediados del 2015 apareció el *ransomware* como servicio (en inglés, *ransomware-as-a-service* o, abreviadamente, RaaS). La idea consiste en que un grupo crea y mantiene un determinado *malware* de rescate y lo pone a disposición de otros grupos para que lo utilicen en sus ataques. Este tipo de servicio suele estar en sitios web alojados

en la red Tor. A través de ellos, y de forma anónima, prácticamente cualquiera puede crear su *ransomware*. Además, las propias plataformas gestionan todos los aspectos necesarios para desplegar un ataque de *ransomware*, incluido el pago, donde las plataformas se quedan con un porcentaje de las ganancias obtenidas en cada rescate. Este porcentaje suele oscilar en torno al 20 % aproximadamente [167].

El año 2016 vio un crecimiento nunca visto, ya que aparecieron muestras con una capacidad de expansión espectacular y que fueron capaces de recaudar grandes cantidades de dinero. El *ransomware* como servicio evolucionó más que nunca, llegando a emplear JavaScript para realizar ataques multiplataforma. Para febrero, ya había miles de sitios de WordPress infectados por este tipo de *ransomware* [176]. Sin embargo, fue el *ransomware* como servicio Cerber [235] descubierto en marzo, el que empezó a revolucionar el año 2016. Este *ransomware* se distribuía como un fichero adjunto en correos electrónicos de *phishing*. El fichero adjunto podía ser de dos tipos, un `dot` o un `wsf`. La extensión `dot` se corresponde con una plantilla de Microsoft Word que puede contener macros. Por supuesto, el fichero adjunto contenía una macro maliciosa que, cuando la víctima pulsaba en "Habilitar contenido", instalaba el *malware* en el equipo. La extensión `wsf` se corresponde con un script ejecutable para Windows. En el correo electrónico, se animaba a la víctima tanto a habilitar las macros como a abrir el fichero `wsf`, en función de la versión recibida. Tan pronto era ejecutado, automáticamente descargaba e instalaba el software malicioso en el dispositivo. Curiosamente, lo primero que hacía este *ransomware* era determinar el país donde se encontraba la máquina infectada, ya que si determinaba que estaba en Armenia, Azerbaiyán, Bielorrusia, Georgia, Kirguistán, Kazajistán, Moldavia, Rusia, Turkmenistán, Tayikistán, Ucrania o Uzbekistán detenía su ejecución. Si no se encontraba en ninguno de esos países, el *malware* esperaba a que el usuario hubiera estado inactivo durante un tiempo y habilitaba el salvapantallas. Cuando la víctima intentaba retomar su actividad, empezaban a aparecer falsas alertas que obligaban al usuario a reiniciar el equipo. Tan pronto era reiniciado, Cerber empezaba a cifrar, tanto del disco local como de unidades en red, ficheros de 442 tipos distintos con los algoritmos AES-256 y RSA. Versiones posteriores de Cerber también convertían a la máquina víctima en participante de una *botnet*, ayudando a la realización de ataques distribuidos de denegación de servicio (DDoS). El rescate inicial exigido por Cerber era de unos 500 USD. Por supuesto, para realizar el pago, la víctima recibía instrucciones de como acceder al si-

tio .onion alojado en la red Tor y como enviar la cantidad correspondiente en bitcoin. Para julio, se estimaba que ya habían infectado unos 150 000 usuarios de Windows a través de 161 campañas, generando en todo el año 2016 un total de 2,3 millones de dólares. Gracias a que los creadores de Cerber lanzaban actualizaciones casi todas las semanas, estuvo activo mucho más tiempo de lo habitual, cesando su actividad en 2018 [36].

Otro *ransomware* que nació a principios de 2016 fue Jigsaw. A simple vista, este *ransomware* era muy similar a otras muestras ya conocidas. Su método de difusión solían ser los correos electrónicos no deseados, estaba escrito en .NET y utilizaba el algoritmo AES para cifrar más de 200 tipos de ficheros distintos. Cuando era ejecutado, utilizaba como nombre de proceso `firefox.exe` o `drpbx.exe` y se añadía a la lista de programas que se ejecutaban al arranque para garantizar la permanencia en el equipo. Una característica que, afortunadamente, este *ransomware* no tenía era la de moverse lateralmente por la red, limitándose a encriptar solo los ficheros locales [124]. Lo curioso (y aterrador) de este *ransomware* era lo que hacía una vez que terminado el proceso de cifrado. Abría una ventana con la información para pagar el rescate junto con una cuenta atrás de una hora. Cuando este temporizador llegaba a cero, eliminaba un fichero. Este comportamiento se repetía durante 72 horas, momento en el que si no se había pagado el rescate, el *ransomware* borraba todos los archivos. Si durante esas 72 horas el equipo era reiniciado, se borraban 1 000 ficheros a modo de castigo. El *ransomware* también indicaba en la nota de rescate que, si se intentaba detener el proceso del *malware*, también se eliminarían 1 000 ficheros. Sin embargo, este último punto resultó ser falso [58]. El rescate exigido por los delincuentes era de unos 150 USD en bitcoin durante las primeras versiones, llegando a 5 000 USD en versiones posteriores [124]. Para el año 2021, se habían detectado 48 variantes, muchas de las cuales incluían notas de rescate traducidas a una variedad de idiomas, incluido alemán, francés, turco, portugués y español [112].

4.3.1. Cifrado de la tabla maestra de archivos

Para algunos desarrolladores de *ransomware*, el hecho de cifrar determinados documentos de las víctimas no era suficiente. De esta idea parece haber nacido Petya. Cuando conseguía infectar un equipo, lo primero que hacía era sobreescribir por completo el registro de arranque principal (*master boot record* en inglés). A continuación,

forzaba el reinicio del equipo, momento en el cual aprovechaba para cifrar la tabla maestra de archivos (*master file table* en inglés) utilizando el cifrado de flujo Salsa20 al mismo tiempo que mostraba la nota de rescate. Por supuesto, el cifrar la tabla maestra de archivos impedía tanto arrancar el propio sistema operativo como acceder a ninguno de los ficheros del disco duro [25]. El método de propagación de este *ransomware* era el envío de correos electrónicos de *phising* dirigidos a personal del departamento de recursos humanos de empresas. Este correo contenía un enlace de Dropbox a un supuesto currículum. El rescate exigido a las víctimas empezaba en 0,99 bitcoin y se duplicaba al cabo de una semana [154].

4.3.2. El inicio de los ataques dirigidos

Los creadores de *ransomware* no disminuyeron el ritmo en el 2016 y siguieron apareciendo nuevas amenazas, como SamSam. Su concepción era muy distinta a las muestras vistas hasta el momento. En primer lugar, no era un *ransomware* creado para ataques en masa, sino que estaba diseñado para ataques dirigidos. En segundo lugar, las víctimas objetivo no eran individuos, sino que eran organizaciones sanitarias, aunque también atacó a gobiernos, escuelas y empresas privadas. Y, en tercer y último lugar, no dependía de que una víctima abriera un fichero malicioso adjunto en un correo electrónico para ejecutarse. En su lugar, SamSam utilizaba una amplia gama de *exploits* y de técnicas de fuerza bruta. Las primeras versiones explotaban vulnerabilidades en servidores JBoss. En 2018, SamSam evolucionó y era capaz de explotar vulnerabilidades en protocolos de escritorio remoto, en servidores web basados en Java y en servidores FTP [32]. El precio del rescate no era fijo, sino que una vez que los atacantes conseguían acceso al sistema, estudiaban el volumen de datos y la capacidad de pago de la víctima para determinar un precio apropiado [164]. Entre los casos más impactantes, los cuales se produjeron en el año 2018, se pueden destacar los dos siguientes: 1) un hospital de Indiana [32], el cual se vio obligado a trabajar con lápiz y papel hasta que determinaron que era más barato pagar el rescate que reparar manualmente sus equipos y 2) el departamento de Transporte de Colorado, el cual se negó por completo a pagar el rescate y se centraron en mitigar los daños, esfuerzo que le costó al estado 1,7 millones de dólares [152]. A finales del año 2018, la cartera de bitcoin de los atacantes tenía un valor de más de 390 000 USD [164].

4.3.3. *Ransomware* para macOS y Android

En 2016 también llegó este tipo de amenazas a los equipos de Apple. En concreto, KeRanger, el cual se hacía pasar por una actualización de software del programa Transmission (un cliente BitTorrent) para infectar a las víctimas [100]. Una vez infectada una máquina, quedaba en estado durmiente durante tres días y estaba programado para cifrar más de 300 tipos de ficheros diferentes [114]. El *ransomware* destinado a plataformas Android también continuó viendo el nacimiento de nuevas muestras con nuevas funcionalidades. Por ejemplo, un software malicioso bautizado como Xbot se dedicaba a cifrar los ficheros del dispositivo y, además, intentaba robar datos bancarios [113].

4.3.4. 500 000 correos electrónicos al día y mil millones de dólares

Según una estimación del FBI, solamente los primeros meses del año 2016 le reportaron a los creadores de *ransomware* unos 209 millones de dólares [76]. Sin embargo, esta cantidad se quedaría pequeña para finales de año debido a la aparición de Locky. Locky se convirtió rápidamente en una de las variantes de *ransomware* más utilizadas y populares de todos los tiempos. Para finales de 2016 ya había recaudado más de mil millones de dólares [115].

Locky era un software malicioso muy sofisticado que infectaba equipos a través de archivos adjuntos de Microsoft Word que contenían macros maliciosas [130]. Estos documentos eran enviados de forma masiva vía correos electrónicos. Los mensajes trataban sobre una factura sin pagar y se persuadía a las víctimas a que abrieran el fichero de Word adjunto. Cuando lo abrían, aparecía un texto ilegible junto a un mensaje que indicaba que se activaran las macros si la factura no se visualizaba correctamente. En el momento en el que se activaban las macros, automáticamente se descargaba un instalador desde el servidor de C&C a una carpeta temporal, el cual se encargaba de cargar directamente el código de Locky en memoria y comenzar la ejecución [55]. Desde el punto de vista técnico, Locky utilizaba los algoritmos RSA-2048 y AES-1024, guardaba las claves en un servidor remoto, exigía el pago en bitcoin y utilizaba la red Tor para todas las conexiones entre las víctimas y sus servidores [144]. Lo más impactante de Locky era el ritmo al que sus creadores enviaban los correos electrónicos de *phising*. Eran capaces de enviar hasta 500 000 correos al día. Para ponerlo en contexto,

una campaña típica de *phising* en el año 2020 enviaba 500 000 mensajes en todo el año [129]. Locky llegó a afectar a 114 países, con cientos de infectados en cada uno de ellos [79].

4.3.5. Se filtra EternalBlue

El 14 de abril de 2017, un grupo conocido como Shadow Brokers filtró en su cuenta de Twitter una herramienta de *hacking* con el nombre de EternalBlue. Esta herramienta fue desarrollada por la Agencia de Seguridad Nacional (NSA) de los Estados Unidos como parte de su controvertido programa de descubrir y explotar vulnerabilidades de seguridad como ciberarmas, en lugar de reportar los fallos encontrados a los fabricantes de software correspondientes [37]. La Agencia de Seguridad Nacional dedicó casi un año a buscar un fallo en el software de Microsoft y a desarrollar un *exploit* que aprovechara dicho fallo. Una vez creada esta herramienta de *hacking*, la utilizaron en innumerables misiones de recopilación de inteligencia durante cinco años antes de notificar a Microsoft sobre la vulnerabilidad. Al parecer, este era uno de los *exploits* más útiles en el arsenal de la agencia [151]. El problema, como se mencionó anteriormente, fue que el grupo Shadow Brokers consiguió hacerse con una copia de este *exploit* y la liberó.

Retrocedemos un mes exacto, al 14 de marzo de 2017. Microsoft publicó el parche MS17-010 para corregir la vulnerabilidad identificada como CVE-2017-0144; vulnerabilidad que aprovechada EternalBlue. Todo el problema estaba en el protocolo SMBv1 (*Server Message Block version 1*). Este protocolo de comunicación se utiliza para que las máquinas Windows se puedan comunicar entre ellas y puedan acceder a servicios remotos como archivos, impresoras o puertos. En concreto, la versión 1 de este protocolo se desarrolló por primera vez en a principios de 1983. Los analistas de la NSA encontraron un fallo en la gestión de paquetes mal formados o, visto de otro modo, especialmente formados. EternalBlue explotaba esta vulnerabilidad y conseguía la ejecución remota de comandos en máquinas a las que se dirigía el *exploit* sin necesidad de ninguna interacción por parte del usuario [37].

Con un *exploit* tan potente a disposición del público no es muy difícil imaginarse lo que sucedió a continuación. Menos de un mes desde la filtración de EternalBlue, apareció WannaCry, el primer *ransomware* que le dio un "buen" uso a este *exploit*. Desde el punto de vista técnico, utilizaba el algoritmo AES para cifrar cada fichero

con una clave diferente y luego cifraba todas las claves individuales con el algoritmo RSA de 2048 bits [146]. Y no se limitaba a cifrar los archivos de los discos locales, sino que cifraba tanto dispositivos extraíbles como unidades de red [231]. Por supuesto, utilizaba la red Tor para la comunicación con el C&C y exigía el pago en bitcoin. En concreto, pedía 300 USD si se pagaba en las 6 primeras horas, pasando después a 600 USD durante 3 días. Pasados esos días, WannaCry amenazaba con la eliminación de los ficheros si no se pagaba en un plazo de una semana. La ventana donde se mostraba el mensaje de rescate incluía también dos cuentas atrás, una para el tiempo que faltaba para que se incrementara el precio y otra para el tiempo que faltaba para que se eliminaran todos los ficheros [169]. En muy pocos días, WannaCry llegó a infectar a más de 300 000 equipos en más 150 países [42], convirtiéndose en el peor ataque de *ransomware* de la historia. Sin embargo, se cree que este *ransomware* no generó grandes beneficios, principalmente porque existía el rumor de que pagar el rescate no descifraba los ficheros [43].

Aunque se desconoce la cifra exacta recaudada por WannaCry, sí existe una estimación de los costes potenciales de este ataque: más 4 000 millones de dólares [29]. No hay que olvidar que, para algunas empresas, pudo suponer perder toda la información, sin forma de recuperar acceso a los ficheros. A modo de ejemplo, se pueden destacar dos casos afectados por WannaCry. El primero fue el servicio nacional de salud (*National Health Service* en inglés) del Reino Unido [187]. Miles de operaciones y otras citas tuvieron que ser canceladas, ya que anunciaron que tardarían semanas en reparar los equipos así como sustituir sistemas anticuados. Miles de empleados del servicio de salud tuvieron que volver a utilizar el papel y el bolígrafo para realizar su trabajo. Uno de los motivos por los que eran vulnerables al ataque era que disponían de múltiples equipos ejecutando aún Windows XP (que ya carecía de soporte en 2017). Por supuesto, el servicio nacional de salud del Reino Unido no era el único que utilizaba un sistema obsoleto como Windows XP. Vista la situación, Microsoft sacó un parche de emergencia que solucionaba esta vulnerabilidad en sistemas operativos ya no soportados, entre ellos Windows XP y Windows Server 2003 [41]. En el caso del servicio nacional de salud del Reino Unido, aún utilizaba Windows XP debido a que determinado equipamiento, como máquinas de rayos X o escáneres de resonancia magnética, solo tenían soporte para ese sistema operativo. Por supuesto, esta circunstancia no era única de este caso, y se pueden encontrar muchísimas más

situaciones donde por motivos de compatibilidad, muchas empresas siguen utilizando software sin soporte. El segundo caso a destacar fue el de Telefónica, la empresa española de telecomunicaciones. Afortunadamente, el impacto pareció haberse limitado a varios equipos de la red interna y no interrumpieron la prestación de servicios ni el funcionamiento de la red. Tras darse a conocer este caso, rápidamente otras empresas, en este caso españolas, como Iberdrola, Gas Natural o Vodafone tomaron medidas preventivas y desconectaron equipos y cortaron acceso a Internet a determinadas redes para evitar ser comprometidos [207].

La propagación de WannaCry fue espectacular hasta que dos eventos permitieron reducir su expansión. El primero fue la liberación por Microsoft del parche de seguridad para sistemas muy usados pero obsoletos. El segundo evento fue el descubrimiento por parte del analista de seguridad Marcus Hutchins de un interruptor de apagado (*kill switch* en inglés) en WannaCry. Cuando estaba haciendo ingeniería inversa descubrió que antes de empezar a cifrar los archivos, WannaCry comprobaba si un determinado dominio estaba registrado y activo. El analista descubrió que dicho dominio estaba sin registrar, por lo que no dudó en comprarlo. Esto permitió detener la expansión de WannaCry en menos de una hora [143].

4.3.6. *Ransomware*?

El año 2017 vio la aparición de un tipo de *malware* que se hacía pasar por *ransomware* cuando en realidad sus intenciones eran totalmente otras. Se trata del *"ransomware"* conocido como NotPetya descubierto en junio de 2017. NotPetya hizo uso de lo mejor hasta el momento. Sus creadores cogieron la capacidad de destrucción del *ransomware* Petya y el *exploit* EternalBlue y lo dirigieron contra distintos objetivos en Ucrania [239]. Cuando infectaba un equipo, destruía por completo la tabla maestra de archivos, dejando inoperativo el sistema por completo. El problema era que la intención de los creadores no era recaudar dinero, ya que ni se molestaron en generar un identificador de infección válido [197] ni en proporcionar una dirección de correo electrónica operativa a la que mandar el pago [239]. Es más, análisis posteriores revelaron que ni los propios creadores podrían restaurar el acceso a los ficheros [79]. Su verdadera intención era dejar inoperativos equipos pertenecientes a grandes empresas, como bancos o proveedores de energía [25]. Por lo tanto, NotPetya no era un *ransomware*, sino que era un tipo de *malware* conocido como *wiper*, el cual se hacía

pasar por *ransomware*, y era utilizado como una ciberarma [197].

4.3.7. Descargas con sorpresa

A finales del año 2017 apareció una nueva muestra, Bad Rabbit, dirigida principalmente a algunos grandes medios de comunicación rusos [150]. A diferencia de la mayoría del *ransomware*, que hacían uso del envío masivo de correos electrónicos de *phishing* para propagarse, Bad Rabbit se propagaba a través de descargas no autorizadas en sitios web comprometidos. En concreto, se hacía pasar por un instalador de Adobe Flash. Esto significaba que una persona podía estar expuesta a este *malware* simplemente por visitar un sitio web malicioso o comprometido y descargando archivos que creía que eran en realidad actualizaciones de Adobe [233]. Por supuesto, existía la posibilidad de que los propietarios de estos sitios que alojaban el software malicioso ni siquiera tuvieran conocimiento de que estaban ayudando a Bad Rabbit a propagarse [3]. En el momento en el que el supuesto instalador de Adobe Flash era ejecutado, automáticamente cifraba los archivos y mostraba el siguiente texto: "If you see this text, your files are no longer accessible. You might have been looking for a way to recover your files. Don't waste your time." El pago exigido era de unos 280 USD en bitcoin y las víctimas disponían de 40 horas para hacer el pago. Hubo víctimas que informaron que realizar el pago sí permitía desbloquear los archivos, a diferencia de otras muestras de *ransomware*, con las cuales era inútil pagar el rescate [234]. Análisis posteriores de Bad Rabbit revelaron similitudes con el *"ransomware"* NotPetya. La primera era que parte del código de Bad Rabbit ya se había visto en NotPetya. La segunda era que ambos incluían la misma lista de dominios que se utilizaron para propagar Bad Rabbit. Algunos de estos dominios habían sido comprometidos ya en junio, pero no habían sido utilizados. La tercera y última era que ambos utilizaban el *exploit* EternalRomance (también desarrollado por la NSA y filtrado por Shadow Brokers) para expandirse a otros ordenadores pertenecientes a la misma red [150].

4.4. 2018-2020: "Caza mayor" (*Big game hunting*)

En el año 2018, y en concreto a mediados de año, apareció un nuevo *ransomware* con el nombre de Ryuk. Su principal objetivo era atacar grandes empresas y pedir los rescates más grandes jamás vistos. Pero, antes de eso es importante conocer su

funcionamiento. Ryuk se trataba de un *ransomware* como servicio, donde cualquier persona o grupo interesado podía enviar una solicitud junto con su currículum para solicitar la membresía al programa de afiliados privado [223]. Su principal método de difusión eran los correos electrónicos de *phising*, donde las víctimas eran engañadas para ejecutar un *dropper*. Un *dropper* es un programa en el que se apoyan muchos tipos de *malware* para la fase inicial de la infección. El propio *dropper* no realiza ninguna actividad maliciosa y suele ser un programa muy pequeño. Su único objetivo es abrir una vía para el ataque posterior, descargando, descomprimiendo e instalando los módulos realmente maliciosos [217]. Ryuk hacía uso de un *dropper* debido a que no "viajaba" solo, sino que trabajaba conjuntamente con Emotet y Trickbot. El primero en ejecutarse era Emotet. Este se dedicaba a crear procesos ficticios y a inyectar código malicioso en otros procesos del sistema para conseguir la persistencia. Los métodos de persistencia incluían diversas técnicas como la modificación de diferentes claves del registro, programación de tareas en segundo plano o la creación de accesos directos en la carpeta de inicio [223]. Cuando terminaba su trabajo, Emotet descargaba e instalaba Trickbot. Trickbot se encargaba de allanarle el camino a Ryuk realizando varias tareas de recopilación de información y deteniendo cualquier proceso que pudiera entorpecer el trabajo de Ryuk. Esto incluía detener hasta 180 servicios y 40 procesos, muchos de ellos relacionados con antivirus y sistemas de seguridad [240]. Sin embargo, también aprovechaba para robar todas las credenciales encontradas, buscar carteras de criptomonedas, utilizar los recursos del sistema para el minado y extenderse por la red. A pesar de todas estas actividades, su función más importante era desplegar a Ryuk [240]. Ryuk buscaba por documentos, fotos, vídeos y bases de datos. Para cada fichero que encontraba, generaba una clave única y procedía a cifrarlo utilizando AES-256. Para dificultar aún más la recuperación, cifraba todas las claves AES generadas con RSA-4096. Sin embargo, la ejecución de Ryuk no se detenía ahí, sino que también cifraba todos los recursos compartidos que encontraba en la red, llegando a utilizar Wake-On-Lan para encender otros ordenadores de la red y cifrar así su contenido [240]. Tras terminar la fase de cifrado, Ryuk mostraba la nota de rescate con sus exigencias si se quería recuperar el acceso a los datos. A pesar de que su aparición fue en el 2018, sus grandes ataques llegaron en el 2019 y 2020. Rápidamente se convirtió en la amenaza más común para las empresas (sobre todo las grandes empresas con más de 3 000 empleados), ya que a mediados de 2019, solo Ryuk representaba el 23,9 % de

todos los incidentes de *ransomware* registrados [102]. Para el final del 2020, el equipo de Ryuk había conseguido recaudar un total de 150 millones de dólares [223]. Según el informe CrowdStrike 2020 Global Threat Report, Ryuk fue responsable de tres de las 10 mayores peticiones de rescate del año: 5,3 millones de dólares, 9,9 millones de dólares y 12,5 millones de dólares. Este último se convirtió en el rescate más alto jamás obtenido por un *ransomware*. El alcance de Ryuk fue mundial y consiguió poner en jaque a industrias enteras y a grandes empresas de todo el mundo [240].

4.4.1. Una colaboración aterradora

En el año 2018, apareció una nueva variante de *ransomware* como servicio conocida como GandCrab, que se describió muchas veces como un *ransomware* ágil [168], ya que las primeras versiones no estaban realmente bien programadas, pero cumplían con su función. Sin embargo, con el paso de las versiones, los creadores lo mejoraron hasta tal punto que se convirtió en la variante de *ransomware* más activa entre 2018 y 2019 [69]. Una de las versiones más potentes llegó en julio de 2019 e incluía nuevas características no vistas previamente [168]. Por ejemplo, ya no necesitaba un servidor de C&C, se ejecutaba sin ningún problema aunque no tuviera acceso a Internet e incorporaba el *exploit* EternalBlue para expandirse con rapidez dentro de una red local [183].

GandCrab se anunciaba en una comunidad de ciberdelincuentes rusos alojada en la *dark web*. Curiosamente, GandCrab no infectaba a máquinas en Rusia, aunque se desconoce quien formaba el equipo que estaba detrás de este *ransomware* [80]. El funcionamiento de este servicio era similar a otros que aparecieron previamente. Se trataba de un programa de afiliados, donde los participantes en este programa pagaban entre un 30 % y un 40 % de los ingresos obtenidos por los rescates a los creadores. GandCrab se distribuía mediante el uso de correos electrónicos de *phishing* y se estima que en su primer mes infectó unos 50 000 ordenadores, la mayoría de ellos en Europa [168].

Acompañado de la llegada del año 2019 aparece una colaboración y una forma de extorsión nunca vista previamente. Pero antes de eso es necesario conocer al *malware* Vidar. Vidar es un software malicioso de recolección de datos que apareció por primera vez a finales de 2018 y sigue activo en 2023 [108], cuyo objetivo es robar tanta información como sea posible del equipo infectado y mandarla a un servidor controlado por los atacantes. Esto incluye contraseñas, documentos, realización de capturas

de pantalla, *cookies* e información de autocompletado de diversos navegadores web, información sobre segundos factores de autenticación, datos bancarios, detalles de tarjetas de crédito o monederos de criptomonedas por nombrar algunos ejemplos [224]. La colaboración entre GandCrab y Vidar dio lugar al primer *ransomware* que filtraba y cifraba los datos de las víctimas.

Un grupo de delincuentes conocido como Team Snatch, que apareció en 2018 y eran socios de GandCrab, dieron inicio a una nueva tendencia [69]. En todos los *ransomware* anteriores, el principal motivo para pagar el rescate era recuperar el acceso a los datos. Sin embargo, si uno disponía de otras formas de recuperar el acceso, como restaurando los ficheros desde una copia de seguridad, no era necesario preocuparse por cumplir las exigencias demandadas por los atacantes. Team Snatch cambió este panorama y, para ejercer mayor presión sobre las víctimas, amenazaban con hacer públicos todos los datos que habían robado. La primera víctima fue CityComp, una empresa alemana de tecnologías de la información. Esta se negó a pagar el rescate exigido y vio todos sus datos publicados abiertamente en Internet para que cualquiera los descargara [69]. A este caso le siguieron varias empresas más como KCSA o BMK. Estas dos empresas vieron filtradas decenas de gigabytes con información privada y financiera de todos sus clientes [157].

El 31 de mayo de 2019 los desarrolladores de GandCrab anunciaron que cerraban el servicio tras supuestamente haber ganado más de 2 000 millones de dólares [121]. En julio, el FBI publicó las claves de descifrado para que las todas las víctimas pudieran recuperar el acceso a los ficheros. Este año también vio la desaparición de Team Snatch. Sin embargo, ya habían sentado precedentes y no tardaron en aparecer nuevas muestras de *ransomware*, como Maze o REvil, que utilizaban esta doble extorsión para motivar a las víctimas [69].

4.4.2. Maze y REvil

Los primeros ataques conocidos de Maze son de mayo de 2019. Inicialmente utilizaban dos métodos de distribución. El primero era a través de correos electrónicos de *spam* con el asunto "Missed package delivery" o "Your AT&T wireless bill is ready to view". El segundo era a través de kits de explotación [66]. Este tipo de kits contienen *exploits* para aprovecharse de una amplia variedad de vulnerabilidades y su objetivo es conseguir ejecutar la carga útil (o *payload* en inglés) en la máquina víctima [232].

En este caso particular, la carga útil era el propio código del *ransomware*. Posteriores versiones de Maze utilizaron otros métodos de distribución más dirigidos y avanzados que variaban en función de su objetivo [196]. Cuando Maze se ejecutaba en un equipo se pasaba varios días explorando el sistema, mapeando la red y recopilando la mayor información posible. Para moverse por la red, buscaba vulnerabilidades en distintos servicios, se conectaba a servidores SMB abiertos, intentaba averiguar contraseñas por fuerza bruta, buscaba archivos de texto plano que pudieran contener credenciales y un largo etcétera. Esta capacidad de trasladarse a nuevos dispositivos dificultó combatir con éxito a Maze, ya que si se conseguía eliminar de un equipo, este se podía volver a infectar desde otro dispositivo ya infectado [56].

Cuando este *ransomware* llegaba a un nuevo equipo, empezaba la transmisión de los datos encontrados a un servidor controlado por los atacantes. Una vez robados todos los datos, estos eran cifrados con dos claves, una simétrica utilizando el cifrado de flujo ChaCha20 y otra asimétrica haciendo uso del algoritmo RSA. Una vez cifrados los archivos, Maze mostraba la petición de rescate por pantalla y reproducía un mensaje de voz. El rescate exigido por los atacantes en bitcoin era realmente elevado y podía oscilar entre los 6 y los 15 millones de dólares [56].

Al igual que en el caso de GandCrab, incluso si las víctimas tenían medios para recuperar los sistemas y pensaban en no pagar el rescate, se enfrentaban a que se publicaran todos los datos. Y no solo eso, sino que Maze subió la apuesta y amenazaba con las siguientes acciones [66]: informar a los medios de comunicación sobre la brecha de seguridad, vender la información valiosa en el mercado negro, informar a las bolsas de valores en las que pudiera cotizar la empresa sobre el ataque y la pérdida de información sensible, informar a los clientes y socios de que la empresa ha sido atacada y utilizar la información robada para atacarlos a ellos también. El equipo de Maze no solo quería ganar dinero con el rescate, sino que, como se puede ver, buscó más formas de sacarle partido a los datos, como vendiéndolos al mejor postor. Tampoco hay que olvidar que la empresa atacada se podía enfrentar a multas por infringir la GDPR, ya que los atacantes tenían ahora una copia de los datos.

Algunos de los ataques más importantes llegaron a finales de 2019 y durante el 2020. Entre ellos se pueden destacar las empresas Cognizant, Canon, Xerox y la ciudad de Pensacola en Florida [238]. Cognizant es una empresa incluida en la lista Fortune 500 y uno de los mayores proveedores de servicios informáticos del mundo. Se estimó que

los daños producidos por el ataque le costaron a la empresa entre 50 y 70 millones de dólares. En el caso de Canon le llegaron a robar 10 TB de datos. A Xerox le robaron más de 100 GB. Y en el caso de la ciudad de Pensacola, se sabe que los atacantes le exigieron un pago de un millón de dólares. Tras convertirse en uno de los tipos de *ransomware* más peligrosos, Maze cesó su actividad en septiembre de 2020. Las víctimas que quisieran eliminar sus datos del sitio web podían ponerse en contacto utilizando el "chat de asistencia" [238].

Como se mencionó anteriormente, el otro *ransomware* que nació tras el cierre de GandCrab fue REvil, también conocido como Sodinokibi [172]. Cuando se descubrió por primera vez en abril de 2019, se detectaron varias similitudes con GandCrab en la forma de operar, como el parecido de las URL y de los servidores C&C utilizados. También se trataba de un *ransomware* como servicio y, en el año 2020, incorporó en sus operaciones la doble extorsión. Sus ataques fueron dirigidos contra conocidas figuras públicas y organizaciones. Además, REvil cumplió en todas las ocasiones la amenaza de publicar los datos robados a través de su propio sitio de filtraciones [172].

Para el año 2020, REvil había conseguido atacar al menos 140 organizaciones según IBM Security X-Force. Sus estimaciones indican que un tercio de las víctimas pagaron el rescate exigido y que una de cada diez tenía su información sensible subastada en la *dark web*. Al tratarse de ataques tan dirigidos a grandes empresas, el rescate exigido variaba en función de los ingresos anuales de las organizaciones víctimas. Según IBM Security X-Force, las peticiones de rescate podían llegar hasta los 42 millones de dólares y hasta el 9 % de los ingresos anuales de la víctima. Solo el año 2020 le reportó al grupo detrás de REvil 81 millones de dólares, como mínimo [51].

En el 2021, REvil no redujo el ritmo y consiguió atacar a importantes proveedores de servicios. En concreto, en abril consiguieron robar planos del gigante tecnológico Apple mediante un ataque a su proveedor, Quanta Computer. También atacaron al importante proveedor de carne JBS en mayo y, en julio, al proveedor de software informático Kaseya, por nombrar algunas de sus operaciones [172].

El método de distribución utilizado no era siempre el mismo, ya que dependía de la estrategia que utilizaran los propios afiliados para llevar a cabo los distintos ataques. Algunos de los utilizados fueron correos electrónicos de *phishing* con archivos adjuntos maliciosos, uso de credenciales comprometidas del Protocolo de Escritorio Remoto (en inglés, *Remote Desktop Protocol* o, abreviadamente, RDP) de Microsoft,

ataques de fuerza bruta a servidores RDP o la explotación de vulnerabilidades en varios servicios de cara al público. Por ejemplo, para comprometer a la empresa Kaseya, explotaron una vulnerabilidad de día cero (identificada posteriormente como CVE-2021-30116) en su software Kaseya VSA, el cual es una herramienta de monitorización y administración remota para profesionales de TI [172].

A la hora de cifrar los datos de las víctimas, REvil se distinguía de otros *ransomware* por su uso del protocolo de intercambio de claves Diffie-Hellman de curva elíptica (en lugar de RSA) y del cifrado de flujo Salsa20 (en lugar de AES) para cifrar los archivos. Estos algoritmos criptográficos utilizan claves más cortas, son muy eficientes y no se pueden descifrar si son implementados correctamente [51].

4.4.3. La magnitud de los daños

Es importante destacar que los *ransomware* descritos en este apartado representaron diferentes hitos en la evolución de este tipo de *malware*, así como revelaron la magnitud de los daños causados en algunos casos. Sin embargo, la imagen conjunta es mucho peor. Por ejemplo, el año 2019 fue catastrófico para EE.UU., donde el 90 % de las empresas Fortune 500 se vieron afectadas por un ataque de este tipo [101]. A esto hay que añadirle 966 agencias gubernamentales, 113 gobiernos y agencias estatales y municipales, 764 proveedores de asistencia sanitaria y 89 universidades, colegios y distritos escolares. Esto suma un coste potencial, para solo EE.UU., de 7,5 mil millones de dólares en el año 2019 [212].

Como se puede ver en los últimos ataques, el principal objetivo de los creadores de *ransomware* pasaron a ser las empresas, en lugar de ser los individuos. Se dieron cuenta de que podían pedir rescates mucho mayores a las empresas, las cuales pierden dinero cada minuto que no pueden estar en funcionamiento. Según varios investigadores en seguridad de la información [149], la cantidad ideal a exigir a un usuario doméstico son 300 USD. Sin embargo, en el caso de una empresa pequeña o mediana, sube hasta los 10 000 USD. Al parecer esta cantidad tiene las siguientes ventajas: 1) es lo suficientemente asequible como para que las empresas decidan pagarlo sin más dilaciones y 2) es una cantidad pequeña como para que las fuerzas de la autoridad destinen unidades para investigar casos aislados.

4.5. 2020-2022: Situación actual

Llegó el año 2020 y con él el inicio de la crisis COVID-19. Los creadores de *ransomware* no perdieron la ocasión y los centros y proveedores sanitarios se convirtieron en el principal punto de mira. Una muestra que se hizo particularmente popular aprovechando el miedo que rodeaba a la pandemia del coronavirus fue Netwalker [153]. El 10 de marzo dejó fuera de servicio a los sistemas del Distrito de Salud Pública de Champaign-Urbana en el estado de Illinois. El 14 de marzo atacó a la segunda institución médica más grande de la República Checa, el Hospital Universitario de Brno. Una de las consecuencias de este ataque fue el retraso de los resultados de docenas de pruebas de Coronavirus. Curiosamente, este ataque tuvo lugar justo dos días antes de que el presidente decretara la cuarentena nacional en el país. En España, varios hospitales también fueron víctimas de este *ransomware* el 25 de marzo. Netwalker también aprovechó para atacar a otras entidades, como ayuntamientos [153]. En mayo, la ciudad austriaca de Weiz cayó víctima de este *ransomware* debido a la apertura de correos electrónicos de *phising* que tenían como asunto "Información sobre el coronavirus". A medida que avanzaba el año, Netwalker siguió buscando nuevos objetivos. En otoño causó grandes estragos en el sector privado mundial. En esta ocasión, el objetivo fue la empresa K-Electric, el mayor proveedor privado de electricidad de Pakistán y el único que suministra energía a Karachi, la capital de la provincia de Sindh. El ataque afectó mayoritariamente a los servicios de facturación, más que al propio suministro. El rescate exigido por los ciberdelincuentes era de 3,8 millones de dólares y amenazaban con aumentarlo a 7,7 millones al cabo de una semana [153].

El nacimiento del *ransomware* Netwalker fue realmente curioso. Tras haber sido desarrollado en agosto de 2019 y haber sido utilizado en algunos ataques a partir de septiembre de 2019, el equipo detrás de Netwalker decidió ampliar su plantilla [57]. En abril de 2020, Netwalker se convirtió en un negocio de *ransomware* como servicio y publicó un anuncio en la *dark web* para buscar socios. A diferencia de otros *ransomware* como servicio descritos anteriormente, el grupo detrás de Netwalker solo quería dos socios. Estos tenían que hablar ruso, tener conocimientos en tecnologías de redes, tener experiencia y distinguirse por utilizar una estrategia de ataque única. Algunas de las características que anunciaba esta plataforma de *ransomware* como servicio era un panel de chat en la red Tor para asistencia técnica, varias opciones de cifrado, procesamiento automático del pago de los rescates, un módulo de desbloqueo, expansión

por la red, rutinas PowerShell para bloquear antivirus y pagos instantáneos [57]. Al igual que otros *ransomware* como servicio, el rescate obtenido era dividido entre los creadores y los socios.

La distribución de Netwalker varió a lo largo del tiempo. Sin embargo, el vector de ataque más utilizado, sobre todo contra el sector sanitario, fue el envío de correos electrónicos de *phising*, normalmente incluyendo en el asunto información sobre el Coronavirus [119]. Estos correos llevaban como adjunto un documento de Word, el cual incluía código VBScript que desencadenaba la instalación del *ransomware* [57]. Una vez infectado, utilizaba un programa llamado WTVConverter.exe para expandirse a otros dispositivos de la red. Para ocultar la ejecución del propio proceso de *ransomware*, los creadores utilizaron una técnica conocida como *process hollowing* o *process replacement* [59]. Lo que hacían era iniciar el proceso `explorer.exe` (un proceso totalmente legítimo) en un estado suspendido. A continuación, sustituían en la memoria el código legítimo de ese proceso por el código malicioso. De esta forma, cuando restauraban la ejecución de dicho proceso, el *ransomware* aparecía como `explorer.exe` en el Administrador de tareas de Windows y no llamaba la atención. A continuación, Netwalker empezaba el proceso de localización de ficheros de interés, transfiriéndolos a un servidor externo controlado por los atacantes y, por último, cifrándolos con Salsa20. Como se puede ver, los creadores de este *ransomware* siguieron la cada vez más popular tendencia de la doble amenaza, robo y cifrado de los datos. Esto es particularmente dañino para empresas del sector sanitario, las cuales se pueden enfrentar a fuertes multas si los datos de los pacientes son revelados. Para demostrarle a las víctimas que el ataque era real, una pequeña muestra de los datos robados era publicada en la *dark web* [119]. Al igual que la mayoría de los *ransomware* de los últimos años, para pagar el rescate exigido las víctimas tenían que acceder a un portal alojado en la red Tor. La cantidad a pagar dependía de la rapidez con la que se cumplieran las exigencias de los atacantes. A medida que pasaba el tiempo, el precio aumentaba. Por supuesto, si no se pagaba el rescate, todos los datos eran vendidos a otros grupos de ciberdelincuentes o publicados en la propia página web de Netwalker. Al igual que se mencionó anteriormente, aunque las víctimas tuvieran a disposición copias de seguridad para restaurar el servicio, en este caso no podían arriesgarse a que los datos se publicasen, ya que podían perder diferentes acreditaciones y, con ello, a la mayoría de sus clientes [57]. Claramente, el método de la doble extorsión aumentó enormemente

la visibilidad del *ransomware* así como su popularidad.

Se estima que en todo el año 2020, solo el *ransomware* Netwalker ganó más de 25 millones de dólares [69]. Por suerte, ese fue su último año. A principios de 2021, autoridades canadienses detuvieron a uno de los socios del equipo de Netwalker. Al mismo tiempo, autoridades estadounidenses y búlgaras consiguieron eliminar el sitio web de Netwalker y, con él, la amenaza de que se revelaran los datos robados. Esto permitió que algunas de las víctimas pudieran recuperar sus sistemas desde copias de seguridad sin tener que preocuparse por la revelación de los datos. Sin embargo, se considera que el equipo sigue activo, por lo que no hay garantías de que no vuelvan al ataque [57].

Como era de esperar, en el año 2020 apareció un *ransomware* con el nombre "CoronaVirus", el cual estaba dirigido a particulares. Este se distribuía a través de correos electrónicos de *phishing* que contenían un enlace a un sitio web falso de WiseCleaner, el cual alojaba un ejecutable con el nombre WSHSetup.exe. "CoronaVirus" no se limitaba a cifrar los archivos de los usuarios, sino que también incluía al troyano KPOT [206]. Este troyano se dedicaba a buscar monederos de bitcoin, obtener las *cookies* de los navegadores y robar las credenciales de inicio de sesión de programas de mensajería, redes privadas virtuales (VPN), clientes FTP, gestores de correo electrónico y plataformas de juegos, entre otros servicios. Toda la información junto con una captura de pantalla del escritorio era enviada a un servidor gestionado por los ciberdelincuentes. Una vez robada toda la información de posible interés, el *ransomware* empezaba a cifrar todos los archivos que tuvieran unas determinadas extensiones. También modificaba el MBR para mostrar el mensaje de rescate en el próximo reinicio durante 15 minutos. La cantidad a pagar para recuperar el acceso a los ficheros era de 50 USD en bitcoin [49]. Es curioso, pero a pesar de que robaban información de las víctimas, no amenazaban a las víctimas con publicar dicha información como suelen hacer las muestras dirigidas a grandes empresas. Parece que los creadores de *ransomware* destinados a individuos optaron por la opción de venderla directamente en el mercado negro, en concreto las credenciales, sin que las víctimas tengan la menor idea.

4.5.1. Emergencia nacional en Costa Rica

De las nuevas amenazas que aparecieron en 2020, una de las más importantes fue Conti. Con sede en Rusia, la variante de *ransomware* Conti fue observada por primera vez en febrero, aunque no tardó en convertirse en uno de los grupos más activos. Aunque no se llegó a confirmar oficialmente, hubo muchos indicios que indicaban que el equipo que estaba detrás de Conti era el mismo que operaba el *ransomware* Ryuk mencionado anteriormente [53]. Al igual que Ryuk, Conti era un *ransomware* como servicio y que utilizaba el método de la doble extorsión. En agosto, pusieron en marcha un sitio web para filtrar todos los documentos confidenciales obtenidos de diferentes objetivos. Para finales de año, el sitio había filtrado datos de más de 150 empresas, lo que los convirtió en uno de los tres grupos de *ransomware* que más información filtró.

Los métodos conocidos que utilizaban para infiltrarse en las redes de las víctimas eran los tres siguientes [53]: 1) campañas de correos electrónicos de *phising* dirigidas a usuarios concretos con correos especialmente preparados para llamar la atención de esas personas, 2) aprovecharse del protocolo RDP, ya sea utilizando credenciales robadas o debido al uso de contraseñas muy débiles y 3) comprar el acceso a la red a través de un intermediario. Este último método de acceso era realmente curioso y consistía en que Conti compraba el acceso a la red objetivo pagándole a otros grupos que ya tuvieran acceso a dicha red. Una vez que conseguían entrar, empleaban herramientas de penetración como Cobalt String o AdFind para propagarse por toda la red. A medida que llegaba a nuevos equipos, detectaba las soluciones de seguridad instaladas y las intentaba desactivar para evitar la detección. Además, también trataba de determinar si estaba siendo ejecutado en un entorno aislado (*sandbox* en inglés) o en una máquina virtual, lo que podría indicar que estaba siendo analizado por expertos en seguridad. Y no solo eso, sino que por donde pasaba instalaba puertas traseras, lo que le permitía volver a entrar en un futuro para realizar espionaje y vigilar la actividad de los objetivos. Este último punto lo utilizaron concretamente para estudiar como la víctima iba a contrarrestar el ataque una vez que hubieran exigido el rescate, lo que les daba una ventaja a la hora de negociar el precio [53]. Tras haber localizado los datos de alto valor, los mandaban a sus servidores y los cifraban. Además, para acelerar el proceso de cifrado, creaban 32 hilos de ejecución, lo que lo convertía en una de las variantes más rápidas [52]. Para dificultar aún más la recuperación de los sistemas por parte de las víctimas, también eliminaba todas las copias de seguridad

localizadas.

Una vez cifrados los sistemas y filtrada toda la información, empezaban las negociaciones. Conti afirmaba que si detectaban que las víctimas compartían información sobre las negociaciones con cualquier entidad externa, automáticamente terminarían las negociaciones y filtrarían directamente todos los datos robados. Y las amenazas no terminaban ahí, ya que si los chats privados de las negociaciones eran publicados una vez finalizado el ataque y que los archivos hubieran sido borrados de los servidores de Conti, el grupo elegiría los datos de otra víctima y los publicaría a modo de castigo [53].

A lo largo del año 2021 llegaron los ataques más grandes y más peligrosos de Conti [236]. En marzo, se infiltró en el Health Service Executive de Irlanda, consiguiendo acceder a un total de 70 000 dispositivos. Para mayo, había cifrado terabytes de datos, inutilizado toda la infraestructura del hospital y filtrados 700 GB de información, incluidos datos sobre pacientes, empleados y contratistas. El grupo exigió 20 millones de dólares para deshacer sus acciones. En mayo, el FBI emitió un comunicado en el que describía ataques similares contra redes sanitarias y de primeros auxilios en Estados Unidos y en todo el mundo, y solicitaba cualquier información sobre las operaciones de Conti. En septiembre, la empresa JVCKenwood reveló que algunos de sus servidores en Europa habían sido comprometidos y que los datos de sus clientes podrían haber sido filtrados. La nota de rescate fue publicada y en ella se podía ver que habían robado casi 2 TB de datos y que exigían un rescate de 7 millones de dólares.

A pesar de la magnitud de los ataques del año 2021, el ataque llevado a cabo entre abril y mayo de 2022 fue el más notorio [236]. Conti consiguió comprometer las redes y sistemas de decenas de organismos gubernamentales de Costa Rica, entre ellos el Ministerio de Hacienda y el Ministerio de Trabajo y Seguridad Social. La situación llegó a tal punto que el gobierno declaró una emergencia nacional, siendo el primer país en hacerlo en respuesta a un ciberataque. Los devastadores ataques le costaron a Costa Rica millones de dólares y demostraron que un grupo de ciberdelincuentes podía llegar a desestabilizar a todo un país. La amenaza representada por Conti en el año 2022 era tan alta que el gobierno estadounidense anunció una recompensa de hasta 10 millones de dólares por cualquier información valiosa sobre el grupo [220].

Afortunadamente, en mayo de 2022, la empresa de ciberseguridad AdvIntel declaró oficialmente muerto a este grupo criminal. Se sospecha que el grupo se disolvió por

discrepancias internas debido a la invasión rusa de Ucrania en febrero de 2022, ya que Conti anunció que apoyaba a Rusia. Unos días después, se filtraron en Twitter innumerables registros de chats internos así como el código fuente del *ransomware* Conti. Estas filtraciones demostraron lo bien organizado que estaba el grupo, que operaba como una *startup* de alta tecnología con puestos y jerarquías establecidos. El 24 de junio, un mes después de que finalizara el ataque a Costa Rica, el último sitio web del grupo Conti fue cerrado [236].

4.5.2. Sin combustible

En el mes de mayo de 2021, Colonial Pipeline, el mayor operador de oleoductos de combustible de Estados Unidos, se vio obligado a cerrar 5 500 millas de tuberías, las cuales se encargaban de transportar el 45 % del combustible de la Costa Este, debido a un ataque de *ransomware* [221]. Esto hizo que innumerables barriles de gasolina, diésel y combustible para aviones en la costa del Golfo quedaran totalmente varados [122]. El grupo responsable de este desastroso ataque de *ransomware* fue DarkSide.

El *ransomware* DarkSide apareció por primera vez en agosto de 2020, afirmando desde un principio que sus objetivos no iban a ser escuelas, hospitales, gobiernos u organizaciones sin ánimo de lucro. Todos sus ataques iban a ser dirigidos a grandes empresas que pudieran pagar grandes sumas de dinero [24]. Incluso estudiaban los registros financieros de las empresas que pensaban atacar con el fin de ajustar el rescate exigido, el cual solía oscilar entre 200 000 y 2 millones de dólares. En octubre de 2020, el grupo hizo una operación muy poco común y extraña. Donó un total de 20 000 USD, obtenidos de diferentes rescates, a un par de obras de caridad. La primera fue Children International, la cual afirmó que no se quedaría con el dinero y, la segunda, The Water Project [215].

En noviembre de 2020, DarkSide cambió por completo su funcionamiento y se convirtió en un negocio de *ransomware* como servicio, dándole la bienvenida a nuevos afiliados que se unieran a sus operaciones [24]. Por supuesto, actualizaron su *ransomware* para incluir el robo de los datos de las víctimas y así poder exigir un rescate mayor, de forma similar a los últimos *ransomware* vistos. También crearon el sitio web donde anunciarían las filtraciones de datos, así como pusieron en marcha su red de distribución de contenidos (*Content Delivery Network* en inglés) para almacenar y explotar los datos sensibles robados [24]. En marzo de 2021 liberaron la versión 2.0 de

su *ransomware*. Y, a mediados de abril, incluyeron la capacidad de realizar ataques de denegación de servicio distribuidos (DDoS) contra los objetivos para añadir presión durante las negociaciones del rescate. También anunciaron que estaban dispuestos a vender información sobre las próximas víctimas a inversores en bolsa de dudosa reputación y poco éticos. De esta forma, estos inversores podrían aprovecharse de la caída en bolsa producida por el anuncio de que dicha empresa había sido atacada [47].

Para obtener el acceso inicial a las redes de las víctimas, DarkSide realizaba ataques de fuerza bruta y aprovechaba vulnerabilidades conocidas en el protocolo RDP. Tras conseguir acceso, el *ransomware* empezaba a recopilar información sobre el entorno. Curiosamente, DarkSide comprobaba el idioma del sistema y solo continuaba la ejecución si el idioma configurado era inglés [148]. Tras las verificaciones iniciales, el *ransomware* intentaba conseguir privilegios de administrador utilizando la técnica de *bypass UAC* para, a continuación, acceder al controlador del dominio. Desde él podía robar credenciales y expandirse rápidamente a otros equipos de interés [241]. A medida que se propagaba por la red, identificaba y eliminaba cualquier copia de seguridad que encontraba así como desactivaba cualquier solución de seguridad activa [194]. Después, se preparaba para llevar a cabo el paso más arriesgado de la operación, el robo de los datos. Esta operación podía ser detectada por el equipo de seguridad de la organización. Todos los datos de interés encontrados por el grupo DarkSide eran enviados a distintos servidores de almacenamiento en la nube como, por ejemplo, Mega. Por último, cifraba todos los archivos encontrados. Curiosamente, utilizaba métodos distintos en función del sistema operativo encontrado. Si un equipo ejecutaba Linux, entonces utilizaba el cifrado ChaCha20 junto con RSA-4096. En cambio, si el sistema era un Windows hacía uso de Salsa20 con RSA-1024.

El ataque por el que probablemente el grupo DarkSide sea más conocido fue el mencionado al principio de esta sección: Colonial Pipeline. Tras haber completado con éxito el ataque, el grupo DarkSide exigió un rescate de 30 millones de dólares, amenazando con subirlo a 60 millones de dólares pasado un cierto tiempo. Por supuesto, si no recibían el dinero, publicarían todos los datos robados. Tras unas fuertes negociaciones, Colonial Pipeline consiguió reducir el pago a 11 millones de dólares y el grupo DarkSide prometió eliminar los datos robados de sus servidores y a no atacar, ni ayudar a nadie a atacar, la red de la empresa en el futuro [122]. Después de este incidente, los reguladores de energía del gobierno estadounidense multaron a Colonial

Pipeline con casi un millón de dólares por tener unas prácticas de seguridad muy pobres, desde falta de políticas de contraseñas fuertes a la ausencia de múltiples factores de autenticación [194].

Otro ataque de importancia realizado por DarkSide en el 2021 fue el de Brenntag, una empresa alemana de suministros químicos que opera en más de 77 países. Brenntag consiguió evitar que sus datos fueran publicados tras pagar un rescate de 4,4 millones de dólares en bitcoin [24]. Otras empresas no cumplieron con las exigencias y vieron sus datos publicados. Al parecer, DarkSide publicó en la *dark web* más de 2 TB de datos robados de al menos 90 víctimas.

4.5.3. Un inicio de año muy atacado

El primer ataque del año 2022 de gran relevancia fue el condado de Bernalillo (Nuevo México). Un *ransomware* dejó fuera de servicio a varios departamentos y oficinas gubernamentales. Los daños producidos por el ataque duraron semanas hasta que consiguieron empezar a restablecer los servicios [68].

En febrero, un grupo conocido como Lapsus$ consiguió atacar con éxito a la mayor empresa de chips semiconductores del mundo, Nvidia [11]. El grupo consiguió robar 1 TB de datos y exigió un millón de dólares para no publicarlos. Nvidia reconoció el ataque e incluso afirmó que se estaban filtrando credenciales de empleados así como diversa información confidencial.

Entre febrero y marzo, tres proveedores de Toyota fueron víctimas de ataques de *ransomware*, lo que obligó a Toyota a detener las operaciones en 14 de sus plantas japonesas. En abril se produjo el desastroso ataque contra Costa Rica mencionado anteriormente. Y, poco después, la aerolínea india SpiceJet también cayó víctima de un *ransomware* [11]. Según sus declaraciones, consiguieron contener con éxito el ataque. Sin embargo, esto no impidió que se produjeran retrasos en innumerables vuelos.

4.5.4. Grandes pérdidas y rescates más elevados

Un informe publicado por el FBI en el año 2022 reveló que el *ransomware* supuso más de 49,2 millones de dólares en pérdidas para Estados Unidos en el año 2021 [106]. A nivel mundial, se registraron 623,3 millones de ataques de *ransomware* en el año 2021, lo que representa un aumento del 105 % con respecto a las cifras de 2020. Las

demandas exigidas por los *ransomware* también experimentaron aumentos drásticos en los últimos años. En 2020, el pago promedio de *ransomware* aumentó un 171 % con respecto al 2019, llegando a los 312 000 USD. Y, en 2021, alcanzó a un récord de 570 000 USD [88].

Los años más "productivos" en términos de ganancias para los ciberdelincuentes fueron el 2020 y el 2021, donde según [171] recaudaron un total de 765 millones de dólares y 766 millones de dólares, respectivamente. Curiosamente, sus ganancias se redujeron drásticamente en el año 2022, donde "solo" consiguieron 457 millones de dólares. Esta diferencia es atribuida a la decisión de la mayoría de las víctimas a no pagar los rescates exigidos, ya que el número de nuevas muestras aumentó considerablemente. Según la empresa de ciberseguridad Fortinet, solo en la primera mitad de 2022 había más de 10 000 muestras de *ransomware* nuevas en plena actividad. Y, según [88], el número de ataques de *ransomware* registrados en estos primeros meses fue de 236,1 millones en todo el mundo.

Capítulo 5

Obtención de las claves de cifrado

Tras haber visto el nacimiento y la rápida evolución del *ransomware* en el capítulo anterior, en este capítulo se muestran cuáles son las posibles contramedidas. Se empezará haciendo una descripción de las soluciones y herramientas actuales para luego describir como recurrimos a técnicas de análisis forense para revertir ataques de diferentes muestras sin pagar el rescate exigido.

5.1. Soluciones existentes contra el *ransomware*

Una de los puntos débiles de un *ransomware* es el cifrado. Como se comentó anteriormente, si el algoritmo utilizado es débil o está mal implementado, el ataque es fácilmente reversible. En este caso se están obviando, por supuesto, los casos de doble extorsión. Aunque vista la reacción de algunas empresas o entidades, parece que no están dispuestas a pagar un rescate ni aunque sean amenazadas con la publicación de los datos. Lo importante es que prácticamente la totalidad de las variantes hacen uso del cifrado simétrico o híbrido. El problema del cifrado simétrico es que, para realizar su tarea, la clave debe estar presente en memoria [132]. Esto significa que la clave utilizada para bloquear los archivos está expuesta en la máquina de la víctima. En la literatura se pueden encontrar diversos trabajos que tratan de capturar las claves de cifrado empleadas antes de que sea demasiado tarde y desaparezcan de memoria. Por ejemplo, P. Bajpai *et al.* [21] usaron ciertas API como disparadores que podrían indicar un posible *ransomware*. Cuando se llamaba a una de esas API, se escaneaba automáticamente la memoria de ese proceso en busca de claves. Las claves encontradas

se enviaban a una base de datos externa cifrada. Si el proceso correspondía realmente a un *ransomware*, la clave utilizada podía recuperarse accediendo a la base de datos.

Siguiendo un esquema muy similar al anterior, Eugene Kolodenker *et al.* [118] implementaron una herramienta a la que llamaron PayBreak. La probaron contra 20 familias diferentes de *ransomware* (un total de 107 muestras diferentes) y fueron capaces de recuperar las claves de 12 de esas familias con una mínima penalización de rendimiento. Aunque publicaron el código fuente de PayBreak, la implementación solo funciona en Windows 7 de 32 bits.

Otra medida proactiva contra el *ransomware* es ShieldFS [54]. ShieldFS hace que el sistema de archivos de Windows sea resistente a los ataques de *ransomware*. Para ello, utilizaron modelos adaptativos para clasificar los procesos como benignos o maliciosos basándose en la actividad del sistema de ficheros. En cuanto un proceso era clasificado como malicioso, las operaciones realizadas en disco por ese proceso se revertían automáticamente. ShieldFS se probó con 11 familias de *ransomware* diferentes (un total de 688 muestras) y consiguió proteger el sistema con éxito en todos los casos.

Microsoft también integró medidas *antiransomware* en sus sistemas operativos Windows 10 y 11. En concreto, incorporó la función "Acceso controlado a carpetas" [170] dentro de Windows Security. Esta función impide que las aplicaciones que no sean de confianza accedan a las carpetas protegidas. Por defecto, esta función está desactivada. Si estuviera activada, probablemente impediría que algunas aplicaciones funcionaran correctamente, ya que bloquea el acceso a algunas de las carpetas del usuario. Para restablecer el funcionamiento normal de estas aplicaciones, el usuario debe marcarlas manualmente como de confianza.

También existen en la literatura soluciones que incorporaron medidas *antiransomware* en el propio *firmware* de las unidades de estado sólido (SSD), como SSD-Insider [20] o MimosaFTL [229]. En este caso se aprovecharon de la característica de "actualización fuera de lugar" de las memorias NAND *flash*, gracias a la cual fueron capaces de recuperar el archivo original después de que hubiera sido cifrado. A pesar de los resultados positivos obtenidos por las dos propuestas mencionadas, quizás el obstáculo más importante para su adopción sean los vendedores de memorias *flash*, ya que son ellos los que tendrían que incluir estas medidas en el *firmware*.

Los defensores y los atacantes están en una pelea continua en la que los defensores

despliegan una nueva barrera y los atacantes ya están pensando como saltársela, o los atacantes encuentran una nueva vulnerabilidad y los defensores tienen que diseñar un parche para evitar su explotación. Por muchas medidas que se tomen, siempre existe la posibilidad de que los atacantes tengan éxito, por lo que es necesario contar con otras medidas para detener o revertir un ataque. Una de las medidas más comunes contra el *ransomware* es tener una copia de seguridad de los datos. El problema es que las políticas de copia de seguridad suelen ser bastante deficientes, con copias obsoletas, incompletas o incluso inaccesibles. Por ejemplo, si se utiliza un volumen en red para las copias de seguridad, hay *ransomware* que no se limita a cifrar el contenido del propio ordenador, sino que busca volúmenes en red y otros ordenadores para propagarse.

Como ocurre con cualquier medida de seguridad, siempre hay formas de saltárselas, por lo que es importante disponer de otras alternativas, como herramientas reactivas, que puedan ayudar a recuperarse de ataques de *ransomware* inesperados o nunca vistos. Por ejemplo, Simon R. Davies *et al.* [63] estudiaron si era posible utilizar técnicas de análisis forense para encontrar las claves de cifrado utilizadas por distintos *ransomware*. Para ello, ejecutaron tres muestras distintas (NotPetya, Bad Rabbit y Phobos) en máquinas virtuales de VirtualBox con Windows 7 y Windows 10. Tras su ejecución, volcaron la memoria de las máquinas virtuales a intervalos regulares. A continuación, realizaron los siguientes experimentos con estos volcados: a) recuperar las claves de cifrado utilizando herramientas existentes como `findaes` e `interrogate`; b) crear una línea temporal en la que la clave utilizada por el *ransomware* estuviera en memoria; y, por último, c) verificar la validez de las claves encontradas. Demostraron que era posible encontrar la clave de cifrado de estos tres *ransomware* siempre que se volcara la memoria en la ventana temporal en la que la clave estaba expuesta en memoria. Sin embargo, realizar este tipo de análisis en máquinas virtuales tiene sus problemas, siendo uno de ellos que no representa un entorno real. Otro problema es que existen *ransomware* (y otros programas maliciosos) que incorporan técnicas para evitar ser ejecutados en entornos virtualizados, evitando así ser analizados. De hecho, Simon R. Davies *et al.* reconocieron que no consiguieron ejecutar algunos *ransomware*, como Satan, en su laboratorio.

5.2. Selección de los *ransomware* a estudiar

Tal y como se mencionó en la sección anterior, el *ransomware* necesita que la clave, o claves, que va a utilizar para cifrar los archivos resida en memoria durante algún tiempo. Esta ventana de tiempo es nuestra oportunidad para obtenerla. Gracias al uso de técnicas de análisis forense, nuestra idea consistió en volcar la memoria del equipo infectado y desarrollar las herramientas necesarias para buscar, identificar y recuperar la clave utilizada, evitando así pagar el rescate.

La investigación realizada durante la tesis sobre el *malware* de rescate (de tipo *cifrador*, por ser el más popular) nació de las siguientes ideas: 1) cómo de fácil es para un *script-kiddie*[1] realizar un ataque de *ransomware* y 2) cómo las víctimas pueden recuperarse de un ataque de este tipo sin pagar, suponiendo que estén utilizando un sistema operativo totalmente estándar. Al indicar un sistema operativo estándar, nos referimos a un sistema al que no se le ha instalado ninguna medida de seguridad adicional.

Las únicas condiciones para llevar a cabo el experimento fueron que el código del *ransomware* debía ser de código abierto, debía poder ejecutarse tanto en plataformas Windows como Linux y debía estar programado para ser malicioso. La razón de este último requisito era evitar seleccionar *ransomware* que almacenara la clave en texto claro en el disco o que utilizara una clave estática definida en el código fuente, ya que en estos casos la recuperación sería trivial.

Poniéndonos en la piel de un *script-kiddie*, el primer paso fue encontrar una plataforma donde obtener el código fuente. En este caso, la elegida fue GitHub por su popularidad y por ser la mayor plataforma de alojamiento de código abierto. El segundo paso fue hacer una búsqueda utilizando el término *"ransomware"*, la cual devolvió más de 3 000 resultados. Hay que destacar que no todos los resultados son *ransomware* operativos. Algunos de los repositorios son proyectos con herramientas de detección o eliminación de *ransomware*, otros agrupan muestras de *ransomware* activos, otros recopilan descifradores para distintos *ransomware*, etc. Al comprobar la popularidad de los diferentes lenguajes de programación devueltos por la búsqueda, llamó la atención que unos 940 proyectos estaban escritos en Python, unos 220 en C#, unos 150

[1] Un *script kiddie* es una persona que hace uso de técnicas y programas ya existentes y conocidos para encontrar y explotar vulnerabilidades [131]. Desconocen por completo como funcionan dichas herramientas y no son conscientes de los daños que pueden llegar a ocasionar. Suelen estar motivados por simples razones personales como divertirse, vengarse o llamar la atención.

en C++, unos 110 en C y el resto de lenguajes tenían menos de 100 resultados. Con esta información decidimos seleccionar Python como lenguaje de programación por dos razones: 1) su popularidad y 2) su facilidad de poder ejecutar el código tanto en plataformas Windows como Linux (cumpliendo así el segundo requisito descrito en el párrafo anterior).

Tras revisar y probar varios proyectos de *ransomware* escritos en Python, nos llamaron la atención los dos aspectos siguientes:

1. La mayoría de estos proyectos cifraban los archivos utilizando el esquema de cifrado AES. En concreto, utilizaban la implementación incluida en el paquete `cryptography` [61] o la incluida en el paquete `PyCryptodome` [158]. Para poner en contexto la relevancia de estos paquetes, ambos se encuentran entre los 360 más descargados del repositorio Python PyPI [161]. Además, con más de 100 millones de descargas mensuales, `cryptography` se encuentra entre los 20 paquetes más descargados cada mes [160].

2. Al intentar recuperar la clave de un volcado de memoria utilizando herramientas existentes para encontrar claves AES en archivos binarios (como `findaes` [73] o `interrogate` [132]), los resultados no fueron muy prometedores. O bien tardaba demasiado tiempo en procesar los volcados, o no encontraba la clave, o devolvía un buen número de posibles claves que había que comprobar una a una si era la correcta. Lo peor de todo es que había situaciones en las que una búsqueda manual por la clave de cifrado indicaba que la clave estaba en el volcado de memoria, mientras que las herramientas mencionadas no la encontraban.

Vistos los resultados de estas pruebas preliminares, empezamos a estudiar como se almacenaban en memoria las claves gestionadas por los dos paquetes de Python mencionados anteriormente (`cryptography` y `PyCryptodome`). Esto nos condujo al desarrollo de dos herramientas, una para cada paquete, que permitieran recuperar directamente de un volcado de memoria las claves de cifrado utilizadas. Además, incorporamos mecanismos para verificar la validez de las claves, de forma que las herramientas solo devolvieran la clave correcta si la localizaban. El código fuente de ambas herramientas está disponible en `https://gitlab.citius.usc.es/xose.fdez/ransom-key-recovery`. Antes de entrar en detalle de como funcionan, es necesario conocer el entorno de pruebas desplegado, el cual se utilizó tanto para la ejecución de

diferentes *ransomware* como para la comprobación del correcto funcionamiento de las herramientas desarrolladas.

5.3. Entorno de pruebas desplegado

Una de las principales consideraciones a la hora de diseñar el entorno fue que debía reflejar lo más fielmente posible una situación real. Esto significaba reservar una máquina exclusivamente para ejecutar los experimentos, en lugar de ejecutar las pruebas en máquinas virtuales.

El ordenador destinado para las pruebas tenía las siguientes características: un procesador Intel Core i7-9700K, 8 GB de RAM y un SSD Intel NVMe de 1 TB. Los sistemas operativos seleccionados fueron Windows 11 versión 21H2 (OS Build 22000.527) y Ubuntu 20.04 (*kernel* 5.4.0-26-generic). Antes de ejecutar los *ransomware*, se realizó una instalación limpia de ambos sistemas operativos, con la configuración predeterminada y sin la instalación de ninguna aplicación adicional.

Una vez que ambos sistemas operativos estuvieron listos, se copiaron varios archivos en la carpeta de documentos del usuario a modo de anzuelo. En concreto, se copiaron un total de 42 archivos que ocupaban 44 MB y que tenían las siguientes extensiones: `txt`, `c`, `cpp`, `csv`, `jpg`, `ods`, `odt`, `png`, `rtf`, `pdf`, `mp3` y `mp4`.

Como se señaló anteriormente, los *ransomware* probados están escritos en Python. Dado que no es realista pensar que una posible máquina víctima tenga instalado el intérprete de Python, así como todas las dependencias necesarias para ejecutar un proyecto concreto, era esencial disponer de una forma de empaquetar el *ransomware* en un único ejecutable. Para ello, se utilizó la herramienta `PyInstaller` [159], tanto para los *ransomware* que ya lo indicaban en su documentación como para aquellos que no indicaban nada respecto a como empaquetar el *ransomware*.

Teniendo en cuenta que todas las muestras probadas permanecían en ejecución realizando alguna tarea, como mostrando una cuenta atrás para asustar a las víctimas, se decidió volcar la memoria de dos formas diferentes: una en la que se volcaba toda la RAM y otra en la que solo se volcaba el espacio de memoria asignado al proceso del *ransomware*. Esto hace un total de cuatro volcados (dos por cada sistema operativo) por *ransomware*. Para identificarlos fácilmente vamos a etiquetarlos de la siguiente manera: `win_ram`, `win_proc`, `ubu_ram` y `ubu_proc`. Todos los volcados se crearon cinco

minutos después de la ejecución del *ransomware*. Sin embargo, los resultados no deberían verse alterados aunque la memoria se volcara más tarde. Lo importante en esta situación es que la memoria se vuelque mientras el *ransomware* está en ejecución. Esto se debe a que cuando un proceso termina, se libera la memoria que tenía asignada, lo que disminuye las posibilidades de recuperar la clave.

Las herramientas necesarias para realizar los volcados mencionados varían en función del sistema operativo utilizado. En el caso de Windows 11, se utilizó `FTK Imager` [78] versión 4.5 para volcar la memoria completa del equipo. Esta herramienta no se instaló en el sistema, sino que se ejecutó directamente desde una unidad extraíble. Para volcar la memoria de un proceso concreto, se abrió el Administrador de Tareas de Windows, se cambió a la pestaña "Detalles", se buscó el proceso correspondiente y se seleccionó la opción "Crear archivo de volcado" del menú contextual. En el caso de Ubuntu 20.04, se utilizó el módulo del *kernel* `LiME` [204] (*commit* ID `fa37b69`) para volcar toda la memoria completa y, para volcar la memoria de un proceso en particular, se utilizó `Procdump` [156] versión 1.2.

Para el análisis de los volcados obtenidos de la máquina infectada se utilizó otro ordenador con las siguientes características: un procesador Intel Core i7-6700K, 16 GB de RAM y un SSD SATA de 1 TB. El sistema operativo utilizado en este caso fue Ubuntu 20.04.

5.4. Estudio del paquete cryptography

El paquete `cryptography` [61] proporciona recetas y primitivas para algoritmos criptográficos comunes como cifrados simétricos, resúmenes de mensajes (*message digests* en inglés) y funciones de derivación de claves. Según [60], su objetivo es convertirse en la "biblioteca criptográfica estándar". Además, según las estadísticas [160], este paquete es el más descargado para cualquier aspecto relacionado con la criptografía.

El *ransomware* hace uso de este paquete para manejar la gestión de claves y el cifrado de archivos. Para estas tareas, el paquete `cryptography` implementa el método de cifrado simétrico Fernet [72]. Fernet utiliza AES de 128 bits en modo CBC con relleno PKCS7 para el cifrado y SHA256 HMAC para la autenticación. De este modo, Fernet garantiza que un mensaje cifrado no pueda leerse ni manipularse sin la clave correcta.

Antes de poder buscar una clave generada por esta implementación de Fernet en un fichero binario es necesario entender como se construye la clave. Las claves se generan mediante el método `generate_key` [198] y su implementación consiste en obtener 32 bytes aleatorios, llamando a `os.urandom`, y luego codificarlos en `base64url`. Esto significa que la clave utilizada tiene 44 caracteres, siendo el último un '`=`' debido al relleno.

Utilizando este conocimiento de la estructura de las claves, desarrollamos una herramienta que llamamos `fernet_key_seek`. Esta herramienta busca el patrón descrito anteriormente en un archivo binario. Una descripción de alto nivel de la herramienta es la siguiente:

1. Recibe dos argumentos: uno es el fichero binario con el volcado de memoria y el otro es uno de los archivos cifrados por el *ransomware*. Este archivo se utiliza para comprobar si la clave encontrada es la correcta o no.

2. El fichero binario se lee byte a byte. Si el byte leído corresponde a un '`=`', se realizan los siguientes pasos:

   a) Se leen los 43 bytes anteriores.

   b) Se comprueba si esos 43 bytes más el '`=`' final corresponden a una codificación `base64url` válida. Si es así, esta podría ser la clave correcta.

   c) Se intenta descifrar el archivo pasado como argumento utilizando la clave encontrada. Si tiene éxito, el programa termina su ejecución imprimiendo la clave por pantalla y dejando el fichero pasado como argumento descifrado en un fichero con el nombre `decrypted.pdf`.

Para comprobar la eficacia de la herramienta, la pusimos a prueba con algunos de los *ransomware* que hacen uso del paquete `cryptography`. En concreto, los *ransomware* seleccionados por su elevado número de estrellas y bifurcaciones (*forks* en inglés) respecto a otros *ransomware* disponibles en GitHub fueron Python-Ransomware y Ransom0.

Para tener una idea del nivel de detección por parte de los antivirus de estas dos muestras, los cuatro ejecutables fueron subidos a VirusTotal [226]. El ejecutable de Python-Ransomware para Windows fue detectado por el 19 % de los motores antivirus

y el de Ransom0 por el 16 %. En el caso de los ejecutables para Linux, el de Python-Ransomware fue detectado por el 8 %, mientras que el de Ransom0 solo fue detectado por el 2 %.

En las subsecciones siguientes se describe en detalle el funcionamiento de cada uno de estos dos *ransomware* para posteriormente mostrar los resultados obtenidos con la herramienta presentada.

5.4.1. Python-Ransomware

Python-Ransomware [162] es un *ransomware* que emplea el cifrado híbrido y que apareció por primera vez el 12 de diciembre de 2019. Desde entonces, su código fuente prácticamente no ha sufrido cambios, siendo la versión aquí analizada la correspondiente al *commit* `251e13e`. En el momento de escribir estas líneas, Python-Ransomware cuenta con 140 bifurcaciones y más de 400 estrellas.

Para hacerse una idea de lo que hace este *ransomware* cuando es ejectado, a continuación se incluye una descripción de alto nivel de su comportamiento:

1. Realiza una conexión a `https://api.ipify.org` para obtener la IP pública de la máquina donde se está ejecutando. Esta información no es utilizada en ningún momento y tampoco es enviada al atacante.

2. Crea una instancia del objeto `Fernet` y genera la clave simétrica que utilizará para cifrar los archivos.

3. Cifra los archivos partiendo del directorio almacenado en la variable `localRoot`. Para cada archivo, comprueba si la extensión está en la lista almacenada en la variable `file_exts`, definida directamente en el código fuente. Si la extensión no coincide con ninguna de las extensiones definidas, entonces el archivo no se cifra.

4. Almacena la clave simétrica en texto claro en disco con el nombre `fernet_key.txt`.

5. Vuelve a leer el contenido de este fichero, lo cifra con la clave pública del atacante utilizando el algoritmo RSA-2048 y lo guarda en el mismo fichero sobreescribiendo el contenido anterior. Es decir, ahora el fichero `fernet_key.txt` contiene la

clave simétrica cifrada con la clave pública del atacante.
Un detalle importante es que la clave pública no está embebida en el código, sino que se lee de un archivo externo. Si no se encuentra este archivo, el programa lanzará una excepción indicando que no se encontró el fichero y el programa terminará su ejecución.

6. Crea el fichero `EMAIL_ME.txt` en el escritorio. En él almacena la clave simétrica utilizada para cifrar los archivos cifrada con la clave pública del atacante. En la nota de rescate, el atacante le indica a la víctima que le mande por correo este fichero para que él pueda proporcionarle la clave de descifrado. Básicamente lo que tiene que hacer el atacante es descifrar este fichero con su clave privada para obtener la clave simétrica utilizada para cifrar los archivos.

7. Sobreescribe el contenido de la variable que contenía la clave simétrica en claro con la clave simétrica cifrada con la clave pública del atacante.

8. Asigna el valor 'None' a la variable que almacenaba el objeto `Fernet` creado en el paso 2.

9. Descarga una imagen de Internet y la guarda en un archivo en el escritorio con el nombre `background.jpg`. A continuación, configura la imagen como fondo de escritorio y abre la URL `https://bitcoin.org`.

10. Obtiene la fecha del sistema. Esta información no es utilizada en ningún momento.

11. Crea un fichero con el nombre `RANSOM_NOTE.txt` en el mismo directorio donde se encuentra el ejecutable del *ransomware*. En este fichero almacena la nota de rescate que está embebida en el código.

12. Lanza un hilo para mostrar la nota de rescate anterior ejecutando `notepad.exe`. Cada 10 segundos, comprueba si la ventana de `notepad.exe` es la ventana activa. Si no lo es, mata el proceso de `notepad.exe` y lo vuelve a lanzar. Este bucle para comprobar si `notepad.exe` es la ventana activa lo ejecuta durante un total de 50 segundos. Dicho de otra forma, intenta asegurarse de que la ventana que muestra el mensaje de rescate sea la ventana que está en primer plano durante un mínimo de 50 segundos.

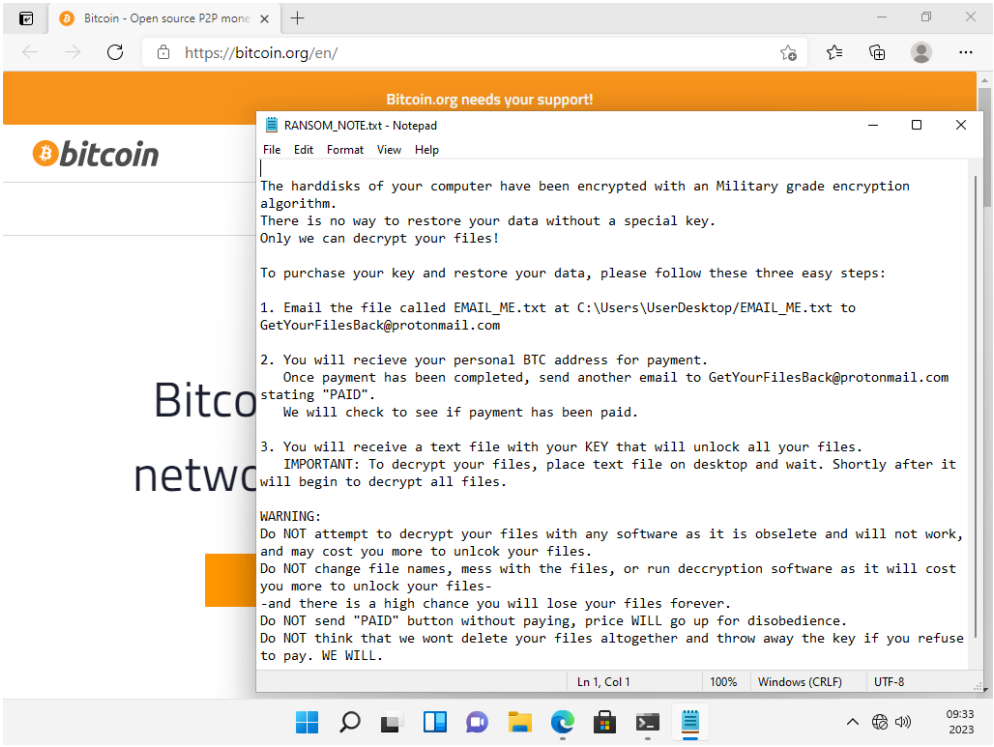

Figura 5.1: Python-Ransomware ejecutándose en Windows 11. Tras cifrar los archivos, abrió la página web de Bitcoin y la nota de rescate.

13. Lanza un segundo hilo que comprueba cada 10 segundos si hay un archivo en el escritorio con el nombre `PUT_ME_ON_DESKTOP.txt`. Se supone que el atacante envía este archivo, el cual contiene la clave de descifrado, a la víctima una vez pagado el rescate. Si este archivo existe en el escritorio, el programa utiliza su contenido como clave para descifrar el sistema. Si la clave es correcta y el descifrado se completa con éxito, el programa finaliza. Si la clave no es correcta, el programa continúa intentándolo cada 10 segundos.

Antes de poder ejecutar el código en la máquina víctima fue necesario realizar algunos ajustes en el código fuente. En concreto, se realizaron los siguientes cambios para la versión de Windows:

1. La variable `file_exts` definida en la línea 24 se modificó para incluir las extensiones de los archivos almacenados en la carpeta de documentos.

2. La ruta definida en la línea 49 se cambió para que apuntara a la carpeta de documentos del usuario. En este caso era `C:\Users\test\Documents`.

Para la versión de Linux se efectuaron los siguientes cambios:

1. La variable `file_exts` definida en la línea 24 se modificó para incluir las extensiones de los archivos almacenados en la carpeta de documentos.

2. La ruta definida en la línea 49 se cambió para que apuntara a la carpeta de documentos del usuario. En este caso era `/home/user/Documents`.

3. Las líneas 11 y 145 fueron comentadas. Estas líneas son las encargadas de cambiar el fondo de pantalla. Dado que el módulo utilizado para realizar esta acción es exclusivo de Windows, se decidió que el fondo de pantalla no se cambiaría en el caso de Linux.

4. Se cambió `notepad.exe` por `gedit` en la línea 178 para que la nota de rescate se abriera correctamente.

Con estos pequeños ajustes, se creó un ejecutable para Windows y otro para Linux utilizando `PyInstaller`. En la Figura 5.1 se puede ver el resultado de ejecutarlo en Windows.

5.4.2. Ransom0

El *ransomware* Ransom0 [166] apareció por primera vez el 28 de julio de 2020 según el *commit* inicial en el repositorio de GitHub. La versión analizada aquí corresponde al último *commit* disponible (`066b4ef`), el cual tiene fecha de 27 de junio de 2022. Sin embargo, las últimas actualizaciones son correcciones del archivo `README`, siendo el último cambio en el propio código del *ransomware* del 1 de junio de 2021. En el momento de la redacción, el repositorio cuenta con 74 bifurcaciones y 317 estrellas.

Este *ransomware* incluye un servidor de C&C (implementado en el archivo `server.py`) que consta de un servidor web y una base de datos SQLite. El servidor web está a la espera de recibir datos de las víctimas y, cada vez que recibe

nueva información, la almacena en la base de datos. A continuación se describe el funcionamiento interno del servidor:

1. Inicia un servidor web utilizando el módulo de Python `http.server`.
2. Si recibe una petición GET, responde con el código `HTTP 200 OK` junto con el mensaje "hi!".
3. Si recibe una solicitud POST, entonces realiza los siguientes pasos:
   a) Responde con el código `HTTP 200 OK`.
   b) Extrae los datos enviados por la víctima. En este caso, obtiene el identificador de la víctima, la clave de cifrado, la fecha y la hora, y la variable `decrypted`.
   c) Establece la conexión con la base de datos, la cual se almacena en un archivo con el nombre `database.db` en el directorio actual.
   d) Si la variable `decrypted` extraída en el paso 3*b* es `true`, entonces elimina toda la información de la víctima de la base de datos, asumiendo que la víctima descifró los archivos.
   e) En cambio, si la variable `decrypted` es `false`, entonces inserta toda la información recibida de la víctima en la base de datos, asumiendo que se trata de una nueva víctima.

Con el funcionamiento del servidor C&C en mente es más fácil entender lo que hace Ransom0. A continuación se incluye una descripción de alto nivel de su funcionamiento:

1. Genera un número entre 1111 y 9999 para identificar a la víctima.
2. Genera la clave que utilizará para cifrar los archivos.
3. Explora todas las carpetas y archivos empezando por el directorio raíz. Para cada directorio, comprueba si está en la lista definida en la variable `EXCLUDE_DIRECTORY`. En caso afirmativo, lo ignora. Para cada archivo que encuentra, comprueba si la extensión coincide con una de las definidas en la lista `EXTENSIONS`. Si es así, escribe la ruta completa a ese fichero en un documento con el nombre `path.txt`.

4. Para cada archivo en `path.txt`, lee su contenido, lo cifra y lo guarda sobrescribiendo el contenido original. A diferencia de otros *ransomware*, Ransom0 no añade ninguna extensión adicional.

5. Envía la siguiente información al servidor de C&C: el identificador de la víctima, la clave de cifrado, la fecha y hora del sistema y un indicador llamado `decrypted` con valor `false`.

6. Abre una ventana maximizada con el texto "SUS ARCHIVOS HAN SIDO CIFRADOS" en letras grandes. Un poco más abajo, muestra el identificador de la víctima junto con las instrucciones que debe seguir para obtener la clave de descifrado. Por último, hay un cuadro de texto donde la víctima puede introducir la clave de descifrado y recuperar así el acceso a los archivos.

7. Cuando la víctima escribe algo en el cuadro de texto y hace clic en el botón con el texto "Descifrar", el *ransomware* intenta descifrar los archivos utilizando el texto proporcionado. Si la clave es correcta, los archivos se descifran y aparece el texto "SUS ARCHIVOS HAN SIDO DESCIFRADOS". Si la clave es incorrecta, no se muestra ningún texto adicional. La víctima puede intentar introducir tantas claves como desee.

8. Esta ventana no se cierra automáticamente, ni cuando la clave introducida es válida ni cuando es inválida. Es la víctima quien debe cerrarla. Tan pronto como se cierra la ventana, el *ransomware* envía la misma información al servidor de C&C que en el paso 5, pero esta vez con la variable `decrypted` establecida a `true`. Esto quiere decir que, si la víctima cierra por error la ventana antes de conseguir descifrar los ficheros, el atacante ya no tiene forma de proporcionarle la clave de descifrado, ya que al cerrar la ventana se envió la orden al C&C de que eliminara la información de la víctima. Si se diera esta situación, el pago del rescate sería totalmente inútil.

Antes de ejecutar el *ransomware*, fue necesario realizar las siguientes dos modificaciones en el archivo `ransom0.py`:

1. Se insertó en la línea 20 la IP donde iba a estar escuchando el servidor.

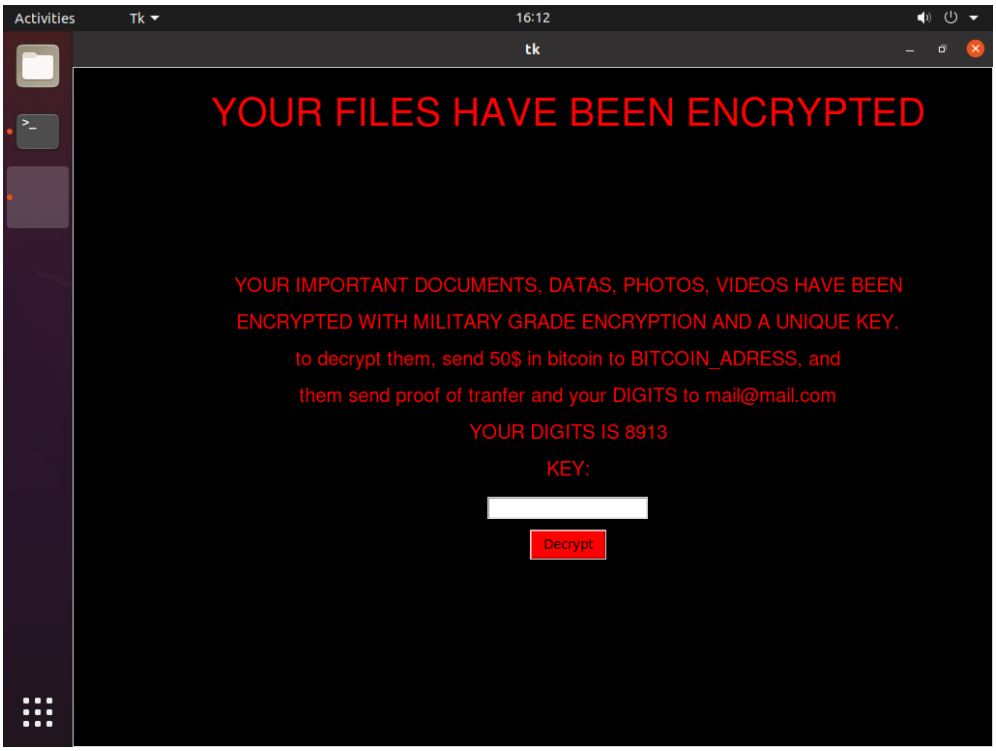

Figura 5.2: Ransom0 ejecutándose en Ubuntu. El *ransomware* muestra una nota informativa indicando que los documentos, fotos y vídeos más importantes han sido cifrados. Para restaurar el acceso a ellos es necesario pagar el rescate.

2. Se modificó la ruta definida en la línea 76 para que, en lugar de intentar cifrar todo el disco, solo cifrara la carpeta de documentos del usuario.

Una vez realizados estos cambios, se crearon los ejecutables para ambos sistemas operativos utilizando `PyInstaller`. A modo de ejemplo, la Figura 5.2 muestra una captura de pantalla tomada justo después de lanzar Ransom0.

5.4.3. Resultados obtenidos con fernet_key_seek

En este apartado se muestra el resultado de ejecutar la herramienta `fernet_key_seek` presentada al principio de esta sección sobre los volcados de memoria creados durante la ejecución de los dos *ransomware* descritos anteriormente. A modo

Ransomware	Volcado	`fernet_key_seek`	`findaes`	`interrogate`
Python-Ransomware	`win_ram`	Sí (7,5 min)	No (3 min)	No (57,5 min)
	`win_proc`	No (10 s)	No (2 s)	No (3,6 min)
	`ubu_ram`	Sí (9,4 min)	No (17 min)	No (-)
	`ubu_proc`	Sí (13 s)	No (5 s)	No (5,2 min)
Ransom0	`win_ram`	Sí (9,3 min)	No (2,7 min)	No (57,7 min)
	`win_proc`	Sí (2 s)	No (2 s)	No (3,3 min)
	`ubu_ram`	Sí (9,7 min)	No (17,7 min)	No (-)
	`ubu_proc`	Sí (2 s)	No (0,4 s)	No (59 s)

Tabla 5.1: Resultado de ejecutar nuestra herramienta (`fernet_key_seek`), `findaes` (versión 1.2) e `interrogate` (*commit* ID `eb5f071`) sobre los volcados de memoria obtenidos durante la ejecución de Python-Ransomware y Ransom0. Un "Sí" indica que dicha herramienta fue capaz de obtener la clave y, un "No", que fue incapaz de encontrarla. El tiempo entre paréntesis muestra cuánto tardó en procesar el volcado y, si no finalizó correctamente, se muestra un "-".

comparativo, también se incluye el resultado de utilizar las herramientas `findaes` e `interrogate`.

Para cada *ransomware*, se realizaron dos ejecuciones distintas en cada sistema operativo. En una se volcó toda la memoria completa y, en la otra, solo se volcó el espacio de memoria asignado al proceso del *ransomware*, tal y como se describió en la Sección 5.3.

La Tabla 5.1 contiene el resultado de procesar los volcados de memoria obtenidos tras la ejecución de ambos *ransomware*. Nuestra herramienta (`fernet_key_seek`) consiguió recuperar la clave de cifrado utilizada en todos los casos excepto en uno. En concreto, cuando solo se volcó el espacio de memoria correspondiente a Python-Ransomware, nuestra herramienta no encontró la clave. Sin embargo, al realizar una comprobación manual, se descubrió que en realidad esta no estaba en el volcado, por lo que no era posible recuperarla.

La herramienta `findaes` no encontró la clave en ninguno de los casos. Sin embargo, cabe destacar que cuando el volcado a procesar era pequeño (el caso de `win_proc` y de `ubu_proc`), `findaes` tardó menos tiempo que `fernet_key_seek`. Por último, la herramienta `interrogate` tampoco encontró la clave en ninguno de los volcados y dio error al procesar algunos de ellos, concretamente `ubu_ram`.

5.5. Estudio del paquete `PyCryptodome`

El paquete `PyCryptodome` proporciona primitivas criptográficas de bajo nivel. En concreto, el *ransomware* utiliza su implementación del cifrado AES para cifrar y descifrar archivos. A diferencia de la implementación Fernet, que combina el cifrado con un código de autenticación de mensajes basado en *hash* (HMAC), en este caso no se puede garantizar que el contenido cifrado no haya sido manipulado.

La implementación del cifrado AES incluida en este paquete admite claves de tamaño 16, 24 o 32 bytes. Normalmente, se utiliza como clave una función que devuelve un resumen criptográfico del tamaño de clave deseado a partir de una cadena de caracteres. Dado que un resumen criptográfico es una cadena de letras y números de tamaño fijo, se puede utilizar una expresión regular para localizar posibles claves. El problema es que, como el cifrado utilizado no incluye autenticación, no hay forma directa ni sencilla de saber si la clave encontrada es la correcta o no. Para resolver este problema, recurrimos a los tipos MIME con el fin de determinar si el fichero descifrado se correspondía con un fichero válido o no.

Con estas ideas en mente, desarrollamos una segunda herramienta, que llamamos `aes_key_seek`, que realiza los siguientes pasos:

1. Recibe dos argumentos: el archivo con el volcado de memoria y uno de los archivos cifrados por el *ransomware*. En la versión actual, el archivo cifrado debe ser un PDF, pero este requisito puede ser cambiado fácilmente.

2. Itera a través de todas las ocurrencias devueltas por la expresión regular del archivo que contiene el volcado de memoria.

3. Para cada ocurrencia, intenta utilizar esa coincidencia como clave. Para ello, descifra el archivo pasado como argumento y luego valida si su tipo MIME corresponde a `application/pdf`. En otras palabras, comprueba si el archivo descifrado es un fichero PDF válido. Si lo es, el programa finaliza su ejecución imprimiendo la clave por pantalla y guardando descifrado el archivo pasado como argumento en un fichero con el nombre `decrypted.pdf`.

Actualmente, `aes_key_seek` admite el cifrado AES en los modos ECB y CBC y solo busca claves de 256 bits, pero puede adaptarse fácilmente para buscar varios tamaños de clave o para probar otros modos de cifrado AES.

Para verificar la eficacia de la herramienta, seleccionamos de GitHub a los dos *ransomware* más populares que hacían uso del paquete `PyCryptodome`: RAASNet y Scrypt.

Al igual que con los dos *ransomware* analizados en la sección anterior, los cuatro ejecutables fueron subidos a VirusTotal [226]. El ejecutable RAASNet para Windows fue detectado por el 26 % de los motores antivirus, mientras que el ejecutable Scrypt solo fue detectado por el 4 %. En el caso de los ejecutables para Linux, el primero fue detectado por el 5 % y el segundo por el 2 %.

Curiosamente, cuando analizamos los resultados devueltos por VirusTotal para el ejecutable de Windows de RAASNet, este fue identificado como Black Kingdom [178]. Este era un *ransomware* que estuvo activo en 2019 y se volvió a ver en 2021 y que, al parecer, estaba basado en el código fuente de RAASNet. Las primeras versiones de Black Kingdom incluso añadían la misma extensión (`.DEMON`) a los archivos cifrados [134]. De los análisis realizados, parece que diferentes funciones, como la de cifrado de los archivos, eran prácticamente idénticas a las de RAASNet [31]. Por supuesto, Black Kingdom incluía algunas funcionalidades adicionales, como la capacidad de explotar vulnerabilidades para elevar privilegios.

En las subsecciones siguientes se incluye una descripción del comportamiento de cada uno de estos *ransomware* así como los resultados obtenidos con la herramienta presentada.

5.5.1. RAASNet

RAASNet [163] es un *ransomware* que apareció por primera vez a finales de julio de 2019. Cuando lo descubrimos por primera vez, el código fuente estaba disponible en GitHub y contaba con más de 300 bifurcaciones y más de 800 estrellas. Sin embargo, a día de hoy, este fue eliminado debido a una violación de las condiciones de servicio de GitHub, aunque aún se puede encontrar una copia del repositorio en `archive.org` [14]. La versión probada aquí se corresponde con el último *commit* disponible (`d3f6129`) cuando el repositorio aún estaba en línea en el 2022, el cual tenía fecha de mediados de diciembre de 2021.

Este *ransomware* es el más completo de todos los analizados. Dispone de una interfaz web (alojada en una dirección .onion) desde la que se puede consultar información

acerca de las víctimas infectadas como, por ejemplo, el sistema operativo, la clave de cifrado utilizada o el nombre del equipo.

Cuando se clona el repositorio y se siguen las instrucciones de instalación, lo primero que aparece es una ventana de inicio de sesión. Si no se ha iniciado sesión, no es posible utilizar la herramienta y, por tanto, no se puede generar el ejecutable que contiene el *ransomware*.

Nuestro objetivo principal era probar el *ransomware* en sí, y no su interfaz de usuario o panel de control. Por lo tanto, dado que todo el código es abierto, en lugar de crear una cuenta y utilizar la interfaz gráfica de usuario para generar la carga útil (*payload* en inglés), nos limitamos a aislar el código necesario para crear el *ransomware*. En concreto, la función principal responsable de generar el *ransomware* es `create_demon`, la cual está implementada en el fichero `create_demon.py` dentro del directorio `src`. Cuando se ejecuta con los argumentos adecuados, esta función genera un archivo llamado `payload.py` que contiene todo el código que se ejecutará en la máquina víctima. Una descripción de alto nivel de este código es la siguiente:

1. Genera una cadena de 64 caracteres aleatorios compuesta por letras mayúsculas y números. A continuación, calcula su resumen criptográfico MD5, el cual será utilizado como clave para cifrar los archivos.

2. Ejecuta el método `platform.system()` para determinar en qué sistema operativo se está ejecutando y el método `platform.node()` para obtener el nombre del equipo.

3. Intenta conectarse al servidor de C&C utilizando la información almacenada en las variables `host` y `port`.

   a) Si la conexión tiene éxito, envía la siguiente información: la IP local del ordenador, el sistema operativo y el nombre del equipo obtenidos en el paso 2, la clave generada en el paso 1 y el nombre del usuario que ejecuta el *ransomware*.

   b) Si la conexión no tuvo éxito, simplemente continúa con la ejecución. Esto implica que aunque se pagara el rescate, el atacante no podría descifrar los archivos porque la clave nunca se envió al servidor de C&C.

4. Para cada uno de los ficheros encontrados dentro de las carpetas definidas en la variable `dirs`, comprueba su extensión para determinar si lo cifra o no según la lista de extensiones definida en la variable `ext`.

5. Para cada archivo a cifrar realiza las siguientes operaciones: lee el contenido, lo cifra con la clave generada en el paso 1, guarda el contenido cifrado en el disco sobrescribiendo el original y le añade la extensión `.DEMON` al nombre del archivo.

6. Para cifrar el contenido, utiliza el cifrado AES en modo CBC. El vector de inicialización se genera aleatoriamente y se añade al principio del archivo.

7. Por cada directorio que atraviesa, crea un archivo `README.txt` con el contenido de la variable `message`, la cual contiene la nota de rescate.

8. Una vez cifrados todos los archivos, crea una ventana que muestra una imagen (almacenada en la variable `photo_code` en `base64`), el mensaje de rescate y una cuenta atrás de 10 horas.

9. Cuando la cuenta atrás llega a cero, el programa simplemente termina su ejecución.

En la Figura 5.3 se puede ver una captura de pantalla tomada segundos después de haber ejecutado RAASNet en Ubuntu. Al mismo tiempo que lanzábamos RAASNet, interceptamos el tráfico de red generado con Wireshark. La Figura 5.4 muestra el tráfico capturado y, en concreto, el paquete que contenía toda la información que mandó el *ransomware* al servidor C&C. Como no cifra la comunicación, se puede ver en texto claro la clave de cifrado utilizada, así como el nombre de usuario, el sistema operativo, la IP y el nombre del equipo.

5.5.2. Scrypt

Scrypt [190] es un *ransomware* cuyo primer *commit* en GitHub es de mediados de 2020. El repositorio original donde se publicó por primera vez aparentemente fue eliminado. La versión probada aquí corresponde a una bifurcación, por lo que se desconoce el número de estrellas o bifurcaciones del repositorio original. Revisando los *commits* de este repositorio, el código no sufrió ningún cambio. En concreto, la versión aquí probada corresponde al último *commit* disponible (`809a587`). Uno de los aspectos que

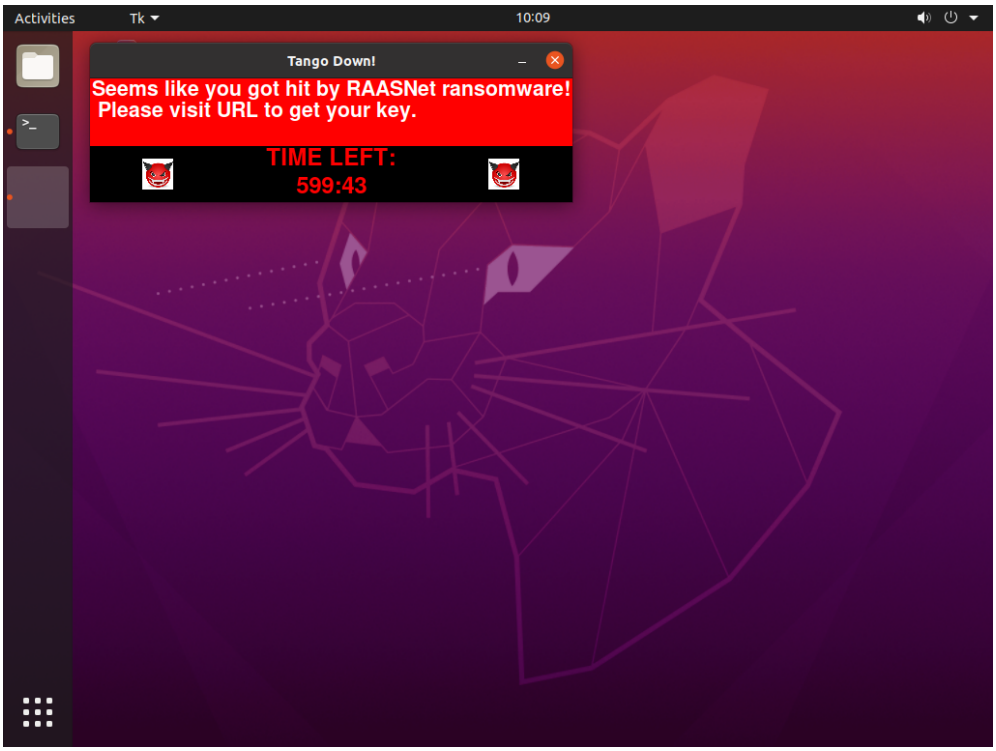

Figura 5.3: RAASNet ejecutándose en Ubuntu. La ventana que creó el *ransomware* es muy sencilla y solo contiene el mensaje de rescate y un temporizador.

diferencia a este *ransomware* de los anteriores es que utiliza la plataforma Discord [67] como servidor de C&C, lo cual puede dificultar su detección a nivel de red.

Una descripción de alto nivel de las acciones realizadas por este *ransomware* es la siguiente:

1. Crea una ventana a pantalla completa con la nota de rescate y un botón con el texto "Continue".

2. Genera una cadena de 12 caracteres compuesta por números y letras mayúsculas y minúsculas para identificar de forma exclusiva a cada víctima.

3. Genera una cadena de 32 caracteres formada por letras mayúsculas y minúsculas. Esta cadena será la clave utilizada para cifrar los archivos.

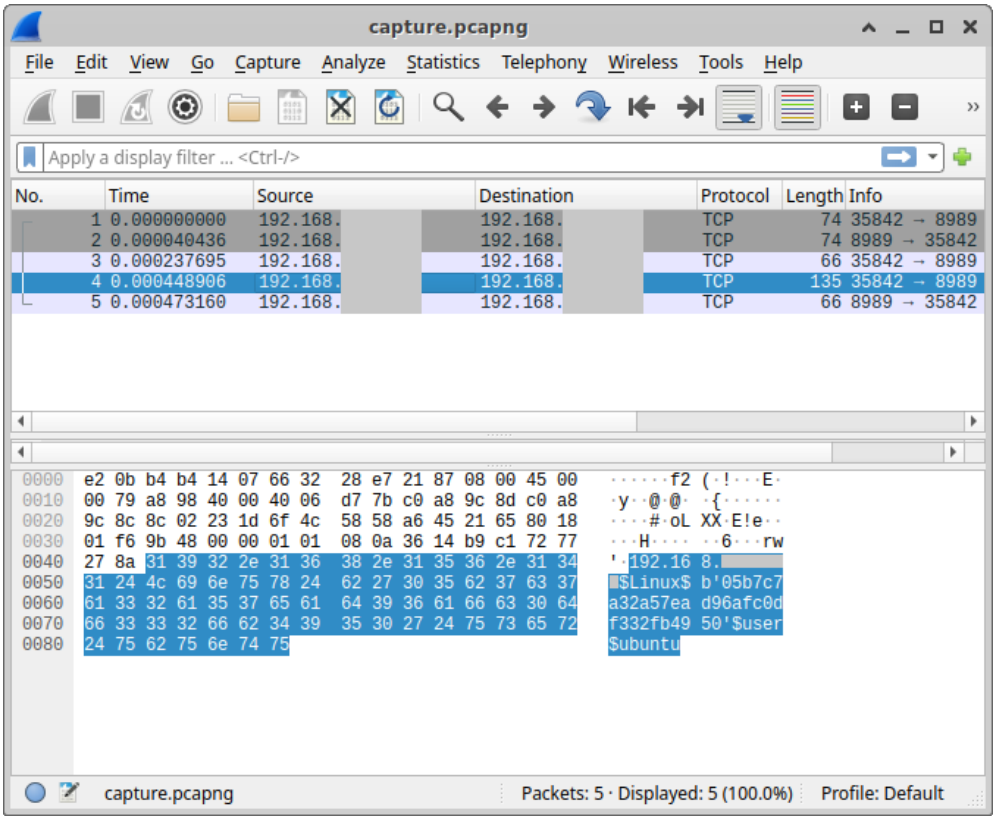

Figura 5.4: Captura del tráfico de red generado tras la ejecución de RAASNet. El paquete seleccionado es el que contiene toda la información esencial sobre la víctima (IP, sistema operativo, clave de cifrado, nombre de usuario y nombre del equipo).

4. Envía un mensaje a Discord a través de un *webhook* con el siguiente contenido: el identificador de la víctima, la clave de cifrado, el nombre de usuario que ejecuta el *ransomware* y la IP de la víctima.

5. Analiza todos los directorios y archivos dentro de la carpeta `C:\Users`. Para cada archivo que encuentra, comprueba si la extensión coincide con una de las definidas en la lista `fileTypes`.

6. Si la extensión de un archivo coincide, entonces crea un nuevo hilo para cifrar ese archivo. Este hilo lee el contenido del fichero, lo cifra utilizando el algoritmo AES

en modo ECB con la clave generada en el paso 3 y lo guarda sobrescribiendo el archivo original.

7. Una vez que ha recorrido todos los ficheros, crea un fichero con el nombre `readme.txt` en el escritorio. En este fichero guarda el contenido de la variable `note`, la cual se supone que contiene el mensaje de rescate.

Antes de ejecutar el *ransomware* fue necesario realizar las siguientes dos modificaciones en el código:

1. Añadir la correspondiente URL del *webhook* de Discord en la línea 17.

2. Modificar la ruta definida en la línea 28 para que, en lugar de que cifrara la carpeta `C:\Users`, cifrara solamente el contenido de la carpeta de documentos.

A modo de ejemplo, la Figura 5.5 muestra lo que observa un usuario cuando ejecuta el *ransomware* Scrypt.

5.5.3. Resultados obtenidos con `aes_key_seek`

En este apartado se muestra el resultado de procesar con la herramienta `aes_key_seek` los volcados de memoria creados durante la ejecución de los *ransomware* presentados anteriormente. A efectos comparativos, y al igual que hicimos en el apartado donde estudiamos el paquete `cryptography`, también incluimos el resultado de procesar los volcados con las herramientas `findaes` e `interrogate`. Para cada *ransomware*, se realizaron dos ejecuciones independientes en cada sistema operativo, tal y como se describió en la Sección 5.3.

La Tabla 5.2 refleja el resultado de procesar todos los volcados de memoria obtenidos tras la ejecución de cada uno de los *ransomware*. Nuestra herramienta (`aes_key_seek`) consiguió recuperar con éxito la clave de cifrado en todos los casos.

La herramienta `findaes` encontró la clave en todos los volcados correspondientes a las ejecuciones en Windows. Sin embargo, cuando el *ransomware* se ejecutó en Ubuntu, solo encontró la clave en el volcado asociado al espacio de memoria de RAASNet. El tiempo de ejecución en los casos en los que sí encontró la clave fue ligeramente inferior al de nuestra herramienta. Cabe destacar que, en las diferentes ejecuciones de la herramienta `findaes`, esta devolvió un buen número de posibles claves que era

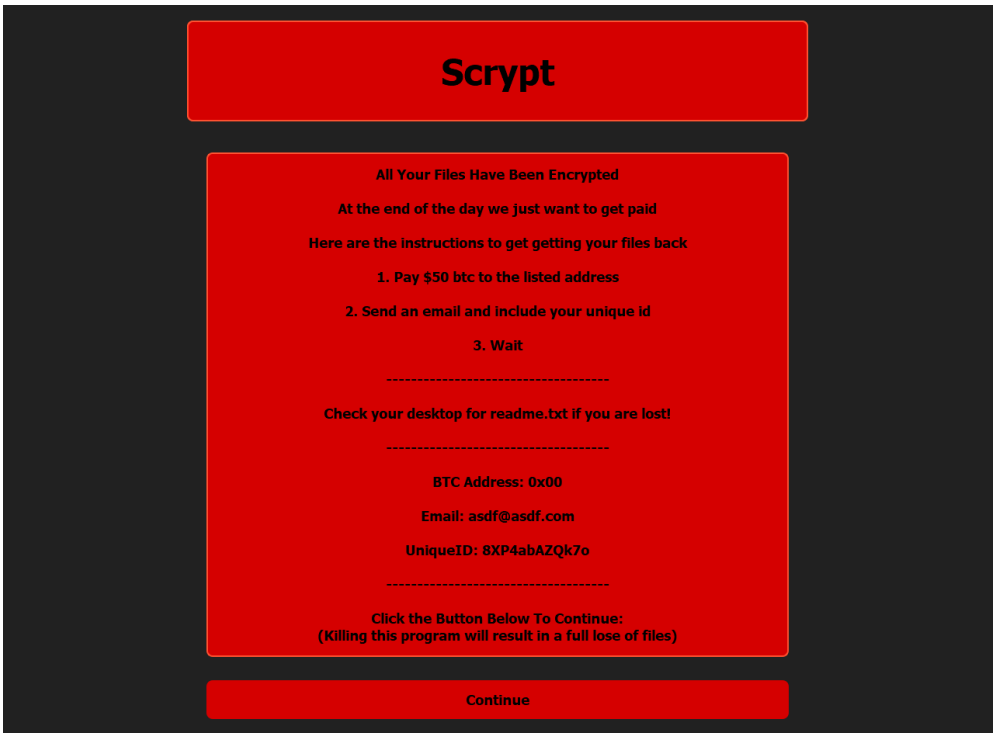

Figura 5.5: Scrypt ejecutándose en Windows. El *ransomware* abrió una ventana a pantalla completa que muestra un mensaje indicando que todos los archivos han sido cifrados. Para recuperarlos es necesario pagar el rescate y esperar a que el atacante envíe la clave de descifrado.

necesario comprobar una a una si era la correcta, mientras que nuestra herramienta devolvió únicamente la clave correcta gracias al uso de tipos MIME.

La herramienta `interrogate` localizó la clave de cifrado utilizada por RAASNet en los casos en los que solo se volcó el espacio de memoria del *ransomware*. En el caso de Scrypt, solo localizó la clave en las ejecuciones de Windows en las que no se volcó la memoria completa. Además, en los casos en los que la herramienta no terminó con un error, tardó mucho más en procesar los archivos que las otras dos.

Ransomware	**Volcado**	`aes_key_seek`	`findaes`	`interrogate`
RAASNet	`win_ram`	Sí (8 min)	Sí (2,8 min)	No (58,3 min)
	`win_proc`	Sí (3 s)	Sí (2 s)	Sí (2,8 min)
	`ubu_ram`	Sí (3,4 min)	No (18,1 min)	No (-)
	`ubu_proc`	Sí (3 s)	Sí (2 s)	Sí (3 min)
Scrypt	`win_ram`	Sí (3,3 min)	Sí (2,8 min)	No (58,3 min)
	`win_proc`	Sí (4 s)	Sí (4 s)	Sí (4,7 min)
	`ubu_ram`	Sí (3,3 min)	No (17,8 min)	No (-)
	`ubu_proc`	Sí (40 s)	No (34 s)	No (-)

Tabla 5.2: Resultado de procesar los volcados de memoria obtenidos durante la ejecución de RAASNet y Scrypt con nuestra herramienta (`fernet_key_seek`), `findaes` (versión 1.2) e `interrogate` (*commit* ID `eb5f071`). Un "Sí" indica que dicha herramienta fue capaz de obtener la clave y, un "No", que fue incapaz de encontrarla. El tiempo entre paréntesis muestra cuánto tardó en procesar el volcado y, si no finalizó correctamente, se muestra un "-".

Las aportaciones más significativas de este capítulo han sido aceptadas para su presentación en la siguiente conferencia:

- Computing Conference 2023, 22-23 junio 2023, Londres (Reino Unido).

Y las actas de la conferencia han sido aceptadas para su publicación en Springer:

- Xosé Fernández-Fuentes, Tomás F. Pena y José C. Cabaleiro, "Recovering from Memory the Encryption Keys used by Ransomware Targeting Windows and Linux Systems", *Proceedings of the Computing Conference 2023*, Springer.
  e-ISSN: 2367-3389.
  Contribución: Conceptualización, Metodología, Investigación, Redacción (borrador original).

Capítulo 6

Conclusiones

Esta tesis se centra, por un lado, en el estudio del nivel de privacidad ofrecido por el modo privado integrado en los navegadores web y, por otro lado, en el estudio del comportamiento del *malware* de rescate o *ransomware*. La primera línea llevó al diseño e implementación de una metodología que permite determinar en qué situaciones estos modos de navegación pueden filtrar información sensible. La segunda línea condujo a un estudio exhaustivo de la forma de actuar de este tipo de *malware* y a la creación de dos nuevas herramientas que ayudan a recuperarse de un ataque de este tipo. Ambas líneas fueron abordadas desde el punto de vista del análisis forense, haciendo uso de diversas técnicas y métodos pertenecientes a esta disciplina.

6.1. Navegación en modo privado

La metodología presentada permite comprobar la eficacia del modo privado incluido en diferentes navegadores web. Esta consiste en realizar un análisis forense exhaustivo tras haber completado una sesión de navegación predefinida utilizando el modo privado en diferentes circunstancias.

La primera aplicación mostrada de esta metodología fue a Mozilla Firefox y a Google Chrome ejecutándose en cuatro entornos Linux diferentes. Se analizó la memoria y el disco duro en busca de cualquier artefacto generado durante las sesiones de navegación. A pesar de ser un análisis dirigido, este muestra la cantidad de información que puede recuperarse de un volcado de memoria completo cuando se realiza un análisis en profundidad. También se mostró que ejecutar Mozilla Firefox o Google

Chrome en una máquina virtual de VMware puede disminuir el nivel de privacidad, permitiendo recuperar información sensible incluso después de reiniciar el ordenador, siendo necesario apagarlo durante un tiempo mínimo para garantizar el borrado de la memoria.

La segunda aplicación fue a los navegadores Google Chrome, Brave, Mozilla Firefox y Tor Browser ejecutándose en tres entornos Android distintos. Todos ellos mantuvieron su palabra de que ninguna actividad realizada en modo privado se almacenaría en el sistema de archivos. Sin embargo, el análisis de la memoria sí permitió recuperar toda o prácticamente toda la sesión de navegación, pudiendo recuperar información sensible (como nombres de usuario, direcciones de correo electrónico y contraseñas) incluso después de reiniciar los dispositivos. Por lo tanto, y al igual que en las versiones de escritorio, la única forma de asegurarse de que no queden artefactos en la memoria es apagar completamente el dispositivo y esperar un tiempo antes de volver a encenderlo. De los cuatro navegadores probados, cabe destacar que Mozilla Firefox y Tor Browser fueron los que menos artefactos dejaron en memoria, siendo Google Chrome el que mayor cantidad de información permitió recuperar. En cuanto a los entornos, el emulador de Android 13 fue del que se pudo obtener un mayor número de artefactos, siendo el emulador de Android 9 el entorno más "privado". Esto también demuestra que el uso de emuladores para realizar análisis forenses a aplicaciones Android puede no ser el mejor entorno, ya que puede dar una falsa sensación de privacidad o, por el contrario, reflejar que se puede obtener mucha más información de la que realmente es posible recuperar en un dispositivo real.

Los análisis también incluyeron la recuperación de archivos previsualizados durante la sesión de navegación, la adquisición del contenido completo de los llaveros integrados en los navegadores, así como diferentes pruebas que revelaron el diferente comportamiento de los propios llaveros.

Como trabajo futuro se puede destacar aplicar la metodología a más navegadores con el fin de tener una visión más amplia de como administran cada uno de ellos el modo privado. También habría que añadir otros hipervisores (como Xen o KVM), más sistemas operativos de escritorio (como Windows, macOS u otras distribuciones Linux) y otros sistemas operativos móviles (como iOS o diferentes ROM de Android). Un área en la que se podría ampliar el alcance de la metodología sería incluyendo el análisis de la memoria de intercambio, lo que implicaría añadir algún entorno con una

cantidad muy limitada de RAM de forma que se forzara el uso de la *swap*.

6.2. *Ransomware*

Durante el transcurso de esta investigación, se desarrollaron dos herramientas para recuperar la clave de cifrado utilizada por diferentes *ransomware* a partir de un volcado de memoria. Ambas herramientas son capaces de procesar volcados de cualquier sistema operativo. La primera de ellas (`fernet_key_seek`) se centra en encontrar claves que hayan sido generadas por la implementación de Fernet proporcionada por el paquete de Python `cryptography`. La segunda (`aes_key_seek`) está diseñada para encontrar la clave de cifrado utilizada por el algoritmo de cifrado AES. En concreto, esta segunda herramienta se ha probado con la implementación proporcionada por el paquete `PyCryptodome`. Ambas herramientas demostraron ser más eficaces y más rápidas a la hora de extraer las claves de cifrado que otras soluciones existentes. Es importante mencionar que, aunque se han probado con las implementaciones proporcionadas por estos dos paquetes de Python, las herramientas deberían ser capaces de encontrar las claves en cualquier volcado de memoria, independientemente del lenguaje de programación utilizado por el software que gestiona las claves. Por ejemplo, si Fernet se utilizara en otro lenguaje de programación, la herramienta desarrollada debería encontrar la clave sin ningún problema, ya que la clave tendría las mismas características. De igual modo ocurriría con el cifrado AES.

Los *ransomware* probados en este trabajo cifran todos los archivos empleando la misma clave. Sin embargo, hay otros *ransomware* más elaborados que cifran cada archivo con una clave diferente. Tal y como están implementadas las herramientas, deberían ser capaces de recuperar cada una de las claves y asociarlas al archivo correspondiente con ligeras o incluso ninguna modificación.

Todas las muestras de *ransomware* analizadas se subieron a VirusTotal [226] para averiguar lo bien que eran detectadas por los motores de antivirus. La sorpresa fue que la muestra más detectada fue RAASNet con solo un 26 %. Y es la más detectada por el simple hecho de que parte de su código se utilizó en una campaña real con el nombre de Black Kingdom. Este bajo nivel de detección facilita que los *script-kiddies* (u otros grupos con más conocimientos) desplieguen fácilmente un ataque de *ransomware* con éxito.

Por supuesto, las herramientas aquí presentadas no sustituyen a ninguna medida de seguridad proactiva contra el *ransomware*. Tampoco están diseñadas para sustituir las buenas prácticas, como la realización de copias de seguridad. Sin embargo, cuando todo lo demás falla y el *ransomware* consigue ejecutarse, hay pocas opciones de recuperación. Por ello, el objetivo de las herramientas desarrolladas es ayudar en este tipo de situaciones.

Como trabajo futuro, se podría destacar analizar otros *ransomware* enfocados a diferentes plataformas como Android o macOS. Otra posible línea de investigación sería desarrollar una medida de protección de código abierto contra el *ransomware* para sistemas Linux que funcionara de forma similar a la incluida en Windows.

Bibliografía

[1] Ahmad Ababneh, Mohammad Abu Awwad y Mohammed I Al-Saleh. «IMO forensics in Android and Windows systems». En: *2017 8th International Conference on Information, Intelligence, Systems & Applications (IISA)*. IEEE. 2017, págs. 1-6.

[2] *Actaeon*. URL: https://www.s3.eurecom.fr/tools/actaeon (Última visita: 30-04-2023).

[3] Oluwademilade Afolabi. *What is Bad Rabbit ransomware?* 2023. URL: https://www.makeuseof.com/what-is-bad-rabbit-ransomware (Última visita: 30-04-2023).

[4] Asmara Afzal et al. «Encrypted network traffic analysis of secure instant messaging application: A case study of Signal messenger app». En: *Applied Sciences* 11.17 (2021), pág. 7789.

[5] Gaurav Aggarwal et al. «An analysis of private browsing modes in modern browsers». En: *19th USENIX Security Symposium (USENIX Security 10)*. 2010.

[6] Nedaa Al Barghouthy y Andrew Marrington. «A comparison of forensic acquisition techniques for Android devices: A case study investigation of Orweb browsing sessions». En: *2014 6th International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE. 2014, págs. 1-4.

[7] Nedaa Al Barghouthy, Andrew Marrington e Ibrahim Baggili. «The forensic investigation of Android private browsing sessions using Orweb». En: *2013 5th International Conference on Computer Science and Information Technology*. IEEE. 2013, págs. 33-37.

[8] Iman M Almomani y Aala Al Khayer. «A comprehensive analysis of the Android permissions system». En: *IEEE Access* 8 (2020), págs. 216671-216688.

[9] Daniel Anderson y Richard von Seck. «The GDPR and its impact on the web». En: *Proceedings of the Seminar Innovative Internet Technologies and Mobile Communications (IITM)*. Ed. por Georg Carle, Stephan Günther y Benedikt Jaeger. Munich: Chair of Network Architectures y Services, 2020, págs. 1-5.

[10] *Android-x86*. URL: https://www.android-x86.org (Última visita: 30-04-2023).

[11] Gabriella Antal. *2022 ransomware statistics & the biggest ransomware attacks*. 2023. URL: https://heimdalsecurity.com/blog/ransomware-statistics/ (Última visita: 30-04-2023).

[12] P Anuradha, T Raj Kumar y NV Sobhana. «Recovering deleted browsing artifacts from web browser log files in Linux environment». En: *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*. IEEE. 2016, págs. 1-4.

[13] Dimitris Apostolopoulos et al. «Discovering authentication credentials in volatile memory of Android mobile devices». En: *Conference on e-Business, e-Services and e-Society*. Springer. 2013, págs. 178-185.

[14] *archive.org copy of the RAASNet GitHub repository*. URL: https://web.archive.org/web/20220408194119/https://github.com/leonv024/RAASNet (Última visita: 30-04-2023).

[15] *Archiveus Trojan*. URL: https://www.knowbe4.com/archiveus-trojan (Última visita: 30-04-2023).

[16] Nitay Artenstein y Gilad Goldman. *Exploiting Android S-Boot: Getting arbitrary code exec in the Samsung Bootloader*. 2017. URL: https://hexdetective.blogspot.com/2017/02/exploiting-android-s-boot-getting.html (Última visita: 30-04-2023).

[17] Muhammad Asim et al. «AndroKit: A toolkit for forensics analysis of web browsers on Android platform». En: *Future Generation Computer Systems* 94 (2019), págs. 781-794.

[18] *Autopsy - Digital Forensics*. URL: https://www.autopsy.com (Última visita: 30-04-2023).

[19] *AVML (Acquire Volatile Memory for Linux)*. URL: https://github.com/microsoft/avml (Última visita: 30-04-2023).

[20] SungHa Baek et al. «SSD-Insider: Internal defense of solid-state drive against ransomware with perfect data recovery». En: *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2018, págs. 875-884.

[21] Pranshu Bajpai y Richard Enbody. «Memory forensics against ransomware». En: *2020 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*. IEEE. 2020, págs. 1-8.

[22] Jim Bates. «Trojan horse: AIDS information introductory diskette version 2.0». En: *Virus Bulletin* 6 (1990), págs. 1143-1148.

[23] Johannes Bauer, Michael Gruhn y Felix C Freiling. «Lest we forget: Cold-boot attacks on scrambled DDR3 Memory». En: *Digital Investigation* 16 (2016), S65-S74.

[24] Derik Belair. *DarkSide ransomware*. URL: https://www.augmentt.com/security/threats/ransomware/dark-side-ransomware (Última visita: 30-04-2023).

[25] Ivan Belcic. *What is Petya ransomware, and why is it so dangerous?* 2019. URL: https://www.avast.com/c-petya (Última visita: 30-04-2023).

[26] Hal Berghel. «Equifax and the latest round of identity theft roulette». En: *Computer* 50.12 (2017), págs. 72-76.

[27] Daniel J Bernstein. «Salsa20 specification». En: *eSTREAM Project algorithm description, http://www. ecrypt. eu. org/stream/salsa20pf. html* (2005).

[28] Daniel J Bernstein. «ChaCha, a variant of Salsa20». En: *Workshop record of SASC*. Vol. 8. 1. Citeseer. 2008, págs. 3-5.

[29] Jonathan Berr. *WannaCry ransomware attack losses could reach $4 billion*. 2017. URL: https://www.cbsnews.com/news/wannacry-ransomware-attacks-wannacry-virus-losses (Última visita: 30-04-2023).

[30] *Bitcoin Wiki*. URL: https://en.bitcoin.it/wiki/Bitcoin (Última visita: 30-04-2023).

[31] *Black Kingdom ransomware*. 2021. URL: https://cyberint.com/blog/research/black-kingdom-ransomware/ (Última visita: 30-04-2023).

[32] Christopher Boyd. *SamSam ransomware: What you need to know*. 2018. URL: https://www.malwarebytes.com/blog/news/2018/05/samsam-ransomware-need-know (Última visita: 30-04-2023).

[33] Bernardo Breve et al. «Enhancing user awareness during Internet browsing». En: *Proceedings of the Fourth Italian Conference on Cyber Security, ITASEC 2020*. Vol. 2597. CEUR Workshop Proceedings. 2020, págs. 71-81.

[34] D Brezinski y T Killalea. «Guidelines for evidence collection and archiving (RFC 3227)». En: *Network Working Group, The Internet Engineering Task Force* (2002).

[35] Martin Brinkmann. *Android 13's upcoming Enhanced Pin Privacy feature explained*. 2023. URL: https://www.ghacks.net/2023/03/31/android-13s-upcoming-enhanced-pin-privacy-feature-explained (Última visita: 30-04-2023).

[36] Michael Buckbee. *Cerber ransomware: What you need to know*. 2022. URL: https://www.varonis.com/blog/cerber-ransomware (Última visita: 30-04-2023).

[37] Carly Burdova. *What is EternalBlue and why is the MS17-010 exploit still relevant?* 2020. URL: https://www.avast.com/c-eternalblue (Última visita: 30-04-2023).

[38] Paul K Burke y Philip Craiger. «Xbox forensics». En: *Journal of Digital Forensic Practice* 1.4 (2007), págs. 275-282.

[39] *CAINE*. URL: https://www.caine-live.net (Última visita: 30-04-2023).

[40] Loredana Caruccio et al. «GDPR compliant information confidentiality preservation in big data processing». En: *IEEE Access* 8 (2020), págs. 205034-205050.

[41] Brad Chacos. *Microsoft issues a rare Windows XP patch to combat a virulent WannaCry-like exploit in older OS versions*. 2019. URL: https://www.pcworld.com/article/397451/microsoft-issues-a-rare-windows-xp-patch-to-combat-a-virulent-wannacry-like-exploit.html (Última visita: 30-04-2023).

[42] Bill Chappell. *WannaCry ransomware: What we know Monday*. 2017. URL: https://www.npr.org/sections/thetwo-way/2017/05/15/528451534/wannacry-ransomware-what-we-know-monday,causing%20major%20disruptions%20worldwide (Última visita: 30-04-2023).

[43] Bill Chappell y Scott Neuman. *U.S. says North Korea 'directly responsible' for WannaCry ransomware attack*. 2017. URL: https://www.npr.org/sections/thetwo-way/2017/12/19/571854614/u-s-says-north-korea-directly-responsible-for-wannacry-ransomware-attack (Última visita: 30-04-2023).

[44] Long Chen y Yue Mao. «Forensic analysis of email on Android volatile memory». En: *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE. 2016, págs. 945-951.

[45] *Chrome-Password-Grabber*. URL: https://github.com/priyankchheda/chrome_password_grabber (Última visita: 30-04-2023).

[46] *Chromium Docs - User data directory*. URL: https://chromium.googlesource.com/chromium/src/+/master/docs/user_data_dir.md (Última visita: 30-04-2023).

[47] Catalin Cimpanu. *Ransomware gang wants to short the stock price of their victims*. 2021. URL: https://therecord.media/ransomware-gang-wants-to-short-the-stock-price-of-their-victims (Última visita: 30-04-2023).

[48] Stefano Cirillo, Domenico Desiato y Bernardo Breve. «CHRAVAT-Chronology awareness visual analytic tool». En: *2019 23rd International Conference Information Visualisation (IV)*. IEEE. 2019, págs. 255-260.

[49] Ben Cohen. *CoronaVirus ransomware*. 2020. URL: https://www.cyberark.com/resources/threat-research-blog/coronavirus-ransomware (Última visita: 30-04-2023).

[50] *Commit where init_on_alloc and init_on_free boot options were introduced*. URL: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=6471384af2a6530696fc0203bafe4de41a23c9ef (Última visita: 30-04-2023).

[51] Lucian Constantin. *REvil ransomware explained: A widespread extortion operation*. 2021. URL: https://www.csoonline.com/article/3597298/revil-ransomware-explained-a-widespread-extortion-operation.html (Última visita: 30-04-2023).

[52] *Conti ransomware (analysis and recovery options).* URL: https://www.provendata.com/conti-ransomware-recovery (Última visita: 30-04-2023).

[53] *Conti ransomware: The history behind one of the world's most aggressive RaaS groups.* 2022. URL: https://flashpoint.io/blog/history-of-conti-ransomware (Última visita: 30-04-2023).

[54] Andrea Continella et al. «ShieldFS: A self-healing, ransomware-aware filesystem». En: *Proceedings of the 32nd annual conference on computer security applications*. 2016, págs. 336-347.

[55] Stephen Cooper. *What is Locky ransomware & how to protect against it?* 2022. URL: https://www.comparitech.com/net-admin/locky-ransomware (Última visita: 30-04-2023).

[56] Stephen Cooper. *What is Maze ransomware & how to protect against it?* 2022. URL: https://www.comparitech.com/net-admin/maze-ransomware (Última visita: 30-04-2023).

[57] Stephen Cooper. *What is NetWalker ransomware & how to protect against it?* 2022. URL: https://www.comparitech.com/net-admin/netwalker-ransomware (Última visita: 30-04-2023).

[58] Stephen Cooper. *What is Jigsaw ransomware & how to protect against it?* 2023. URL: https://www.comparitech.com/net-admin/jigsaw-ransomware (Última visita: 30-04-2023).

[59] Nathan Coppinger. *Netwalker ransomware guide: Everything you need to know.* 2022. URL: https://www.varonis.com/blog/netwalker-ransomware (Última visita: 30-04-2023).

[60] *Cryptography PyPI.* URL: https://pypi.org/project/cryptography (Última visita: 30-04-2023).

[61] *Cryptography web page.* URL: https://cryptography.io/en/latest (Última visita: 30-04-2023).

[62] Joan Daemen y Vincent Rijmen. «Reijndael: The advanced encryption standard.» En: *Dr. Dobb's Journal: Software Tools for the Professional Programmer* 26.3 (2001), págs. 137-139.

[63] Simon R. Davies, Richard Macfarlane y William J. Buchanan. «Evaluation of live forensic techniques in ransomware attack mitigation». En: *Forensic Science International: Digital Investigation* 33 (2020), pág. 300979.

[64] Miguel Lopez Delgado. «Análisis forense digital». En: *Criptored* (2007).

[65] *Digital forensics*. URL: https://www.interpol.int/en/How-we-work/Innovation/Digital-forensics (Última visita: 30-04-2023).

[66] Cezarina Dinu. *Maze ransomware: Origins, operating mode, attacks*. 2021. URL: https://heimdalsecurity.com/blog/maze-ransomware-101 (Última visita: 30-04-2023).

[67] *Discord web page*. URL: https://discord.com (Última visita: 30-04-2023).

[68] Peyton Doyle. *Bernalillo County ransomware attack still felt weeks later*. 2022. URL: https://www.techtarget.com/searchsecurity/news/252512445/Bernalillo-County-ransomware-attack-still-felt-weeks-later (Última visita: 30-04-2023).

[69] Veronica Drake. *The history and evolution of ransomware attacks*. 2022. URL: https://flashpoint.io/blog/the-history-and-evolution-of-ransomware-attacks (Última visita: 30-04-2023).

[70] Jesse Emspaka y Kim Ann Zimmermann. *Internet history timeline: ARPANET to the World Wide Web*. 2022. URL: https://www.livescience.com/20727-internet-history.html (Última visita: 30-04-2023).

[71] Robert E Endeley. «End-to-end encryption in messaging services and national security—case of WhatsApp messenger». En: *Journal of Information Security* 9.01 (2018), págs. 95-99.

[72] *Fernet spec*. URL: https://github.com/fernet/spec/blob/master/Spec.md (Última visita: 30-04-2023).

[73] *findaes source code*. URL: https://sourceforge.net/projects/findaes/files/ (Última visita: 30-04-2023).

[74] Calum Findlay y Petra Leimich. «An assessment of data leakage in Firefox under different conditions». En: *7th International Conference on Cybercrime Forensics Education & Training, CFET 2014*. Canterbury, UK, 2014.

[75] *Firefox Decrypt*. URL: https://github.com/unode/firefox_decrypt (Última visita: 30-04-2023).

[76] David Fitzpatrick y Drew Griffin. *Cyber-extortion losses skyrocket, says FBI*. 2016. URL: https://money.cnn.com/2016/04/15/technology/ransomware-cyber-security (Última visita: 30-04-2023).

[77] *FTK Forensic Toolkit - Exterro*. URL: https://www.exterro.com/forensic-toolkit (Última visita: 30-04-2023).

[78] *FTK Imager - Exterro*. URL: https://www.exterro.com/ftk-imager (Última visita: 30-04-2023).

[79] Steven Furnell y David Emm. «The ABC of ransomware protection». En: *Computer Fraud & Security* 2017.10 (2017), págs. 5-11.

[80] *GandCrab ransomware*. URL: https://www.malwarebytes.com/gandcrab (Última visita: 30-04-2023).

[81] David M Goldschlag, Michael G Reed y Paul F Syverson. «Hiding routing information». En: *Information Hiding: First International Workshop Cambridge, UK, May 30–June 1, 1996 Proceedings*. Springer. 2005, págs. 137-150.

[82] *Google Pushes Mandatory Full-Disk Encryption in Android 6.0*. 2015. URL: https://www.securityweek.com/google-pushes-mandatory-full-disk-encryption-android-60/ (Última visita: 30-04-2023).

[83] *GPcode.AK Ransomware*. URL: https://www.knowbe4.com/gpcodeak-ransomware (Última visita: 30-04-2023).

[84] Mariano Graziano, Andrea Lanzi y Davide Balzarotti. «Hypervisor memory forensics». En: *International Workshop on Recent Advances in Intrusion Detection*. Springer. 2013, págs. 21-40.

[85] Andy Greenberg. *Hackers remotely kill a Jeep on the highway—with me in it*. 2015. URL: https://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway (Última visita: 30-04-2023).

[86] Glenn Greenwald. «NSA collecting phone records of millions of Verizon customers daily». En: *The Guardian* (jun. de 2013). URL: https://www.theguardian.com/world/2013/jun/06/nsa-phone-records-verizon-court-order (Última visita: 30-04-2023).

[87] Glenn Greenwald. «NSA Prism program taps in to user data of Apple, Google and others». En: *The Guardian* (jun. de 2013). URL: https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data (Última visita: 30-04-2023).

[88] Charles Griffiths. *The latest 2023 ransomware statistics*. 2023. URL: https://aag-it.com/the-latest-ransomware-statistics (Última visita: 30-04-2023).

[89] Michael Gruhn y Tilo Müller. «On the practicability of cold boot attacks». En: *2013 International Conference on Availability, Reliability and Security*. IEEE. 2013, págs. 390-397.

[90] Hana Habib et al. «Away from prying eyes: Analyzing usage and understanding of private browsing». En: *Fourteenth symposium on usable privacy and security (SOUPS 2018)*. 2018, págs. 159-175.

[91] *HackBrowserData*. URL: https://github.com/moonD4rk/HackBrowserData (Última visita: 30-04-2023).

[92] Ashley Hansberry, A Lasse y Andrew Tarrh. «Cryptolocker: 2013's most malicious malware». En: *Retrieved February* 9 (2014), pág. 2017.

[93] Abeerah Hashim. *6 times when hackers forced companies to go bankrupt and shut down*. 2023. URL: https://privacysavvy.com/security/business/6-times-hackers-forced-companies-to-go-bankrupt-shut-down (Última visita: 30-04-2023).

[94] Jonathan Hassell. *Cryptolocker: How to avoid getting infected and what to do if you are*. 2013. URL: https://www.computerworld.com/article/2485214/cryptolocker-how-to-avoid-getting-infected-and-what-to-do-if-you-are.html (Última visita: 30-04-2023).

[95] Martin Hellman. «New directions in cryptography». En: *IEEE transactions on Information Theory* 22.6 (1976), págs. 644-654.

[96] *History of cyber security*. URL: https://www.javatpoint.com/history-of-cyber-security (Última visita: 30-04-2023).

[97] Thomas J Holt, Adam M Bossler y Kathryn C Seigfried-Spellar. *Cybercrime and digital forensics: An introduction*. Routledge, 2018.

[98] Graeme Horsman. «A process-level analysis of private browsing behavior: A focus on Google Chromes incognito mode». En: *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE. 2017, págs. 1-6.

[99] Graeme Horsman et al. «A forensic examination of web browser privacy-modes». En: *Forensic Science International: Reports* 1 (2019), pág. 100036.

[100] Matthew Hughes. *A history of ransomware: Where it started & where it's going*. 2016. URL: https://www.makeuseof.com/tag/history-ransomware-russia-reveton (Última visita: 30-04-2023).

[101] Mamoona Humayun et al. «Internet of things and ransomware: Evolution, mitigation and prevention». En: *Egyptian Informatics Journal* 22.1 (2021), págs. 105-117.

[102] Ionut Ilascu. *Ryuk, Sodinokibi ransomware responsible for higher average ransoms*. 2019. URL: https://www.bleepingcomputer.com/news/security/ryuk-sodinokibi-ransomware-responsible-for-higher-average-ransoms (Última visita: 30-04-2023).

[103] Shujaa Imran. *Should you be worried about thieves stealing your iPhone passcode?* 2023. URL: https://www.makeuseof.com/thieves-stealing-iphone-passcode-worry (Última visita: 30-04-2023).

[104] *inotify-tools*. URL: https://github.com/inotify-tools/inotify-tools (Última visita: 30-04-2023).

[105] *inotifywait for Android*. URL: https://github.com/dstmath/inotifywait-for-Android (Última visita: 30-04-2023).

[106] *Internet crime report 2021*. 2022. URL: https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf (Última visita: 30-04-2023).

[107] Abid Khan Jadoon et al. «Forensic analysis of Tor Browser: A case study for privacy and anonymity on the web». En: *Forensic Science International* 299 (2019), págs. 59-73.

[108] *January 2023's most wanted malware: Infostealer Vidar makes a return while Earth Bogle njRAT malware campaign strikes*. 2023. URL: https://blog.checkpoint.com/2023/02/13/january-2023s-most-wanted-malware-infostealer-vidar-makes-a-return-while-earth-bogle-njrat-malware-campaign-strikes (Última visita: 30-04-2023).

[109] Keith Jarvis. «Cryptolocker ransomware». En: *Viitattu* 20 (2013), pág. 2014.

[110] Gerard Johansen. *Digital forensics and incident response*. Packt Publishing Ltd, 2017.

[111] Sina Keshvadi, Mehdi Karamollahi y Carey Williamson. «Traffic characterization of instant messaging apps: A campus-level view». En: *2020 IEEE 45th Conference on Local Computer Networks (LCN)*. IEEE. 2020, págs. 225-232.

[112] Linas Kiguolis. *Jigsaw ransomware virus. 48 variants listed. 2021 update*. 2021. URL: https://www.2-spyware.com/remove-jigsaw-ransomware-virus.html (Última visita: 30-04-2023).

[113] Jeremy Kirk. *A new Android trojan steals your banking info and holds your files ransom*. 2016. URL: https://www.pcworld.com/article/419689/a-new-android-banking-trojan-is-also-ransomware.html (Última visita: 30-04-2023).

[114] Jeremy Kirk. *Apple shuts down first-ever ransomware attack against Mac users*. 2016. URL: https://www.pcworld.com/article/419940/apple-shuts-down-first-ever-ransomware-attack-against-mac-users.html (Última visita: 30-04-2023).

[115] Muhammad Ubale Kiru y Aman B Jantan. «The age of ransomware: Understanding ransomware and its countermeasures.» En: *Artificial Intelligence and Security Challenges in Emerging Networks*. IGI Global, 2019, págs. 1-37.

[116] Ivor Kollár. «Forensic RAM dump image analyzer». Tesis de maestría. Charles University in Prague, Faculty of Mathematics y Physics, 2010.

[117] Konrad Kollnig et al. «Goodbye tracking? Impact of iOS app tracking transparency and privacy labels». En: *arXiv preprint arXiv:2204.03556* (2022).

[118] Eugene Kolodenker et al. «Paybreak: Defense against cryptographic ransomware». En: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. 2017, págs. 599-611.

[119] Edward Kost. *How does Netwalker ransomware work?* 2023. URL: https://www.upguard.com/blog/what-is-netwalker-ransomware (Última visita: 30-04-2023).

[120] Cary Kostka. *Archiveus Trojan: A part of the history of modern ransomware*. 2022. URL: https://ransomware.org/blog/archiveus-trojan-a-part-of-the-history-of-modern-ransomware (Última visita: 30-04-2023).

[121] Brian Krebs. *Who's behind the GandCrab ransomware?* 2019. URL: https://krebsonsecurity.com/2019/07/whos-behind-the-gandcrab-ransomware (Última visita: 30-04-2023).

[122] Brian Krebs. *A closer look at the DarkSide ransomware gang*. 2021. URL: https://krebsonsecurity.com/2021/05/a-closer-look-at-the-darkside-ransomware-gang (Última visita: 30-04-2023).

[123] Michael Kretschmer, Jan Pennekamp y Klaus Wehrle. «Cookie banners and privacy policies: Measuring the impact of the GDPR on the web». En: *ACM Transactions on the Web* 15.4 (2021), págs. 1-42.

[124] Vamsi Krishna. *Jigsaw ransomware and how to combat it*. 2022. URL: https://www.manageengine.com/log-management/cyber-security/jigsaw-ransomware-and-how-to-combat-it.html (Última visita: 30-04-2023).

[125] Sundar Krishnan, Bing Zhou y Min Kyung An. «Smartphone forensic challenges». En: *International Journal of Computer Science and Security* 13.5 (2019), págs. 183-200.

[126] *La historia de los navegadores web*. URL: https://www.mozilla.org/es-ES/firefox/browsers/browser-history (Última visita: 30-04-2023).

[127] *Let's Encrypt stats. Percentage of web pages loaded by Firefox using HTTPS (data source: Firefox Telemetry)*. URL: https://letsencrypt.org/stats/#percent-pageloads (Última visita: 30-04-2023).

[128] John Leyden. *Russian cops cuff 10 ransomware Trojan suspects*. 2010. URL: https://www.theregister.com/2010/09/01/ransomware_trojan_suspects_cuffed (Última visita: 30-04-2023).

[129] *Locky and friends*. URL: https://ransomware.org/what-is-ransomware/the-history-of-ransomware/#locky-and-friends/ (Última visita: 30-04-2023).

[130] *Locky ransomware*. URL: https://www.knowbe4.com/locky-ransomware (Última visita: 30-04-2023).

[131] Ben Lutkevich. *What is a script kiddie?* 2021. URL: https://www.techtarget.com/searchsecurity/definition/script-kiddy-or-script-kiddie (Última visita: 30-04-2023).

[132] Carsten Maartmann-Moe, Steffen E. Thorkildsen y André Årnes. «The persistence of memory: Forensic identification and extraction of cryptographic keys». En: *digital investigation* 6 (2009), S132-S140. URL: https://github.com/carmaa/interrogate (Última visita: 30-04-2023).

[133] Damon McCoy et al. «Shining light in dark places: Understanding the Tor network». En: *Privacy Enhancing Technologies: 8th International Symposium, PETS 2008 Leuven, Belgium, July 23-25, 2008 Proceedings 8*. Springer. 2008, págs. 63-76.

[134] Tomas Meskauskas. *How to remove Black Kingdom/GAmmAWare ransomware from the system*. 2021. URL: https://www.pcrisk.com/removal-guides/17885-black-kingdom-ransomware (Última visita: 30-04-2023).

[135] Kathleen Miles. *Glenn Greenwald on why privacy is vital, even if you 'have nothing to hide'*. 2014. URL: https://www.huffpost.com/entry/glenn-greenwald-privacy_n_5509704 (Última visita: 30-04-2023).

[136] Reza Montasari y Pekka Peltola. «Computer forensic analysis of private browsing modes». En: *International Conference on Global Security, Safety, and Sustainability*. Springer. 2015, págs. 96-109.

[137] *Morris worm*. URL: https://www.hypr.com/security-encyclopedia/morris-worm (Última visita: 30-04-2023).

[138] Matt Muir, Petra Leimich y William J Buchanan. «A forensic audit of the Tor Browser Bundle». En: *Digital Investigation* 29 (2019), págs. 118-128.

[139] Damir Mujezinovic. *AIDS trojan: The story behind the first ever ransomware attack*. 2021. URL: https://www.makeuseof.com/aids-trojan-the-first-ransomware-attack-in-history (Última visita: 30-04-2023).

[140] Joe Mullin. *"I have secrets": Ross Ulbricht's private journal shows Silk Road's birth*. 2015. URL: https://arstechnica.com/tech-policy/2015/01/silk-road-trial-fbi-reveals-whats-on-ross-ulbrichts-computer-in-open-court (Última visita: 30-04-2023).

[141] Dakota Murphey. *A history of information security*. 2019. URL: https://www.ifsecglobal.com/cyber-security/a-history-of-information-security (Última visita: 30-04-2023).

[142] Rebecca Nelson, Atul Shukla y Cory Smith. «Web browser forensics in Google Chrome, Mozilla Firefox, and the Tor Browser Bundle». En: *Digital Forensic Education*. Springer, 2020, págs. 219-241.

[143] Lily Hay Newman. *How an accidental 'kill switch' slowed Friday's massive ransomware attack*. 2017. URL: https://www.wired.com/2017/05/accidental-kill-switch-slowed-fridays-massive-ransomware-attack (Última visita: 30-04-2023).

[144] Philip O'Kane, Sakir Sezer y Domhnall Carlin. «Evolution of ransomware». En: *Iet Networks* 7.5 (2018), págs. 321-327.

[145] *OpenText EnCase Forensic*. URL: https://www.opentext.com/products/encase-forensic (Última visita: 30-04-2023).

[146] Harun Oz et al. «A survey on ransomware: Evolution, taxonomy, and defense solutions». En: *ACM Computing Surveys (CSUR)* 54.11s (2022), págs. 1-37.

[147] *PALADIN EDGE*. URL: https://sumuri.com/product/paladin-edge-64-bit (Última visita: 30-04-2023).

[148] Yogesh Patil. *DarkSide ransomware*. 2021. URL: https://blog.qualys.com/vulnerabilities-threat-research/2021/06/09/darkside-ransomware (Última visita: 30-04-2023).

[149] Darren Pauli. *Ransomware scum find the sweet spot to coin it without copping it*. 2015. URL: https://www.theregister.com/2015/04/30/ransomware_scum_stay_feet_to_beat_the_heat_rsa_ciso_says (Última visita: 30-04-2023).

[150] Alex Perekalin. *Bad Rabbit: A new ransomware epidemic is on the rise*. 2017. URL: https://www.kaspersky.com/blog/bad-rabbit-ransomware/19887 (Última visita: 30-04-2023).

[151] Nicole Perlroth y Scott Shane. *In Baltimore and beyond, a stolen N.S.A. tool wreaks havoc*. 2019. URL: https://www.nytimes.com/2019/05/25/us/nsa-hacking-tool-baltimore.html (Última visita: 30-04-2023).

[152] Alina Georgiana Petcu. *SamSam ransomware 101: How it works and how to avoid it*. 2020. URL: https://heimdalsecurity.com/blog/samsam-ransomware (Última visita: 30-04-2023).

[153] Alina Georgiana Petcu. *What is Netwalker ransomware?* 2022. URL: https://heimdalsecurity.com/blog/netwalker-ransomware-explained (Última visita: 30-04-2023).

[154] *Petya ransomware*. URL: https://www.knowbe4.com/petya-ransomware (Última visita: 30-04-2023).

[155] *Private Browsing - Use Firefox without saving history*. URL: https://support.mozilla.org/en-US/kb/private-browsing-use-firefox-without-history (Última visita: 30-04-2023).

[156] *ProcDump for Linux*. URL: https://github.com/Sysinternals/ProcDump-for-Linux (Última visita: 30-04-2023).

[157] *Profiling Team Snatch: Cybercriminal group publishes five new breaches*. 2019. URL: https://www.zerofox.com/blog/team-snatch-data-breach (Última visita: 30-04-2023).

[158] *PyCryptodome web page*. URL: https://www.pycryptodome.org/en/latest (Última visita: 30-04-2023).

[159] *PyInstaller web page*. URL: https://pyinstaller.org/en/stable (Última visita: 30-04-2023).

[160] *PyPI stats*. URL: https://pypistats.org/top/ (Última visita: 30-04-2023).

[161] *Python Wheels*. URL: https://pythonwheels.com (Última visita: 30-04-2023).

[162] *Python-Ransomware GitHub repository*. URL: https://github.com/ncorbuk/Python-Ransomware (Última visita: 30-04-2023).

[163] *RAASNet GitHub repository*. URL: https://github.com/leonv024/RAASNet (Última visita: 30-04-2023).

[164] Steve Ragan. *SamSam explained: Everything you need to know about this opportunistic group of threat actors*. 2018. URL: https://www.csoonline.com/article/3263777/samsam-explained-everything-you-need-to-know-about-this-opportunistic-group-of-threat-actors.html (Última visita: 30-04-2023).

[165] Kapil Raina. *Zero Trust security explained: Principles of the Zero Trust model*. 2023. URL: https://www.crowdstrike.com/cybersecurity-101/zero-trust-security (Última visita: 30-04-2023).

[166] *Ransom0 GitHub repository*. URL: https://github.com/HugoLB0/Ransom0 (Última visita: 30-04-2023).

[167] *Ransomware history timeline*. URL: https://www.knowbe4.com/ransomware#ransomwaretimeline (Última visita: 30-04-2023).

[168] *Ransomware knowledgebase - GandCrab ransomware*. URL: https://www.knowbe4.com/gandcrab-ransomware (Última visita: 30-04-2023).

[169] *Ransomware knowledgebase - WannaCry ransomware*. URL: https://www.knowbe4.com/wannacry-ransomware (Última visita: 30-04-2023).

[170] *Ransomware protection in Windows Security*. URL: https://support.microsoft.com/en-us/topic/ransomware-protection-in-windows-security-445039d6-537a-488a-ad53-48906f346363 (Última visita: 30-04-2023).

[171] *Ransomware revenue down as more victims refuse to pay*. URL: https://blog.chainalysis.com/reports/crypto-ransomware-revenue-down-as-victims-refuse-to-pay/ (Última visita: 30-04-2023).

[172] *Ransomware spotlight: REvil*. URL: https://www.trendmicro.com/vinfo/gb/security/news/ransomware-spotlight/ransomware-spotlight-revil (Última visita: 30-04-2023).

[173] Khushboo Rathi et al. «Forensic analysis of encrypted instant messaging applications on Android». En: *2018 6th international symposium on digital forensic and security (ISDFS)*. IEEE. 2018, págs. 1-6.

[174] Teresa Reidt. *Android x86: How to choose the right Android OS for x86 systems*. 2022. URL: https://emteria.com/blog/android-x86 (Última visita: 30-04-2023).

[175] *Reveton worm ransomware*. URL: https://www.knowbe4.com/reveton-worm (Última visita: 30-04-2023).

[176] Ronny Richardson y Max M North. «Ransomware: Evolution, mitigation and prevention». En: *International Management Review* 13.1 (2017), pág. 10.

[177] Bander A. S. Al-rimy, Mohd A. Maarof y Syed Z. M. Shaid. «Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions». En: *Computers & Security* 74 (2018), págs. 144-166.

[178] Marc Rivero. *Black Kingdom ransomware*. 2021. URL: https://securelist.com/black-kingdom-ransomware/102873/ (Última visita: 30-04-2023).

[179] Ronald L Rivest, Adi Shamir y Leonard Adleman. «A method for obtaining digital signatures and public-key cryptosystems». En: *Communications of the ACM* 21.2 (1978), págs. 120-126.

[180] Dorit Ron y Adi Shamir. «Quantitative analysis of the full bitcoin transaction graph». En: *Financial Cryptography and Data Security: 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers 17*. Springer. 2013, págs. 6-24.

[181] *Run apps on the Android Emulator*. URL: https://developer.android.com/studio/run/emulator (Última visita: 30-04-2023).

[182] Mark Russinovich. *Process Monitor - Sysinternals*. URL: https://learn.microsoft.com/en-us/sysinternals/downloads/procmon (Última visita: 30-04-2023).

[183] Anmol Sachdeva. *Rewritten GandCrab ransomware targets SMB vulnerabilities to attack faster*. 2018. URL: https://fossbytes.com/gandcrab-ransomware-smb-vulnerabilities-attack-faster (Última visita: 30-04-2023).

[184] *Samsung Upload Client*. URL: https://github.com/bkerler/sboot_dump (Última visita: 30-04-2023).

[185] Gandeva B Satrya, Philip T Daely y Soo Young Shin. «Android forensics analysis: Private chat on social messenger». En: *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE. 2016, págs. 430-435.

[186] Kevin Savage, Peter Coogan y Hon Lau. *The evolution of ransomware*. 2015. URL: https://docs.broadcom.com/doc/the-evolution-of-ransomware-15-en (Última visita: 30-04-2023).

[187] Patrick Sawer et al. *NHS cyber chaos hits thousands of patients*. 2017. URL: https://www.telegraph.co.uk/news/2017/05/13/nhs-cyber-chaos-hits-thousands-patients (Última visita: 30-04-2023).

[188] *sboot_dump*. URL: https://github.com/nitayart/sboot_dump (Última visita: 30-04-2023).

[189] David Schütz. *Accidental $70k Google Pixel lock screen bypass*. 2022. URL: https://bugs.xdavidhu.me/google/2022/11/10/accidental-70k-google-pixel-lock-screen-bypass (Última visita: 30-04-2023).

[190] *Scrypt GitHub repository*. URL: https://github.com/REVENGE977/ScryptRansomware (Última visita: 30-04-2023).

[191] Jérôme Segura. *Citadel: A cyber-criminal's ultimate weapon?* 2012. URL: https://www.malwarebytes.com/blog/news/2012/11/citadel-a-cyber-criminals-ultimate-weapon (Última visita: 30-04-2023).

[192] Adam Shortall y MA Hannan Bin Azhar. «Forensic acquisitions of WhatsApp data on popular mobile platforms». En: *2015 Sixth International Conference on Emerging Security Technologies (EST)*. IEEE. 2015, págs. 13-17.

[193] *SIFT Workstation — SANS Institute*. URL: https://www.sans.org/tools/sift-workstation (Última visita: 30-04-2023).

[194] Aidan Simister. *What is DarkSide ransomware*. 2023. URL: https://www.lepide.com/blog/what-is-darkside-ransomware (Última visita: 30-04-2023).

[195] Gary Sims. *32-bits is dead: Here's what it means for Android, Apple, and more*. 2021. URL: https://www.androidauthority.com/arm-32-vs-64-bit-explained-1232065 (Última visita: 30-04-2023).

[196] Fedor Sinitsyn, Nikita Galimov y Vladimir Kuskov. *Life of Maze ransomware*. 2020. URL: https://securelist.com/maze-ransomware/99137 (Última visita: 30-04-2023).

[197] Stu Sjouwerman. *NotPetya is a cyber weapon, not ransomware*. URL: https://blog.knowbe4.com/notpetya-is-a-cyber-weapon-not-ransomware (Última visita: 30-04-2023).

[198] *Source code of the* `generate_key` *function of the Fernet implementation provided by the* `cryptography` *package*. URL: https://github.com/pyca/cryptography/blob/main/src/cryptography/fernet.py#L47 (Última visita: 30-04-2023).

[199] *Source code of the GetEncryptionKey function of the Chromium project*. URL: https://source.chromium.org/chromium/chromium/src/+/main:components/os_crypt/os_crypt_posix.cc;l=44-45;drc=c497cafe09858c478aa9daa3061e345f0683e61d (Última visita: 30-04-2023).

[200] *Statcounter global stats*. URL: https://gs.statcounter.com (Última visita: 30-04-2023).

[201] Cliff Stoll. *The cuckoo's egg: Tracking a spy through the maze of computer espionage*. Simon y Schuster, 2005.

[202] MAK Sudozai et al. «Forensics study of IMO call and chat app». En: *Digital investigation* 25 (2018), págs. 5-23.

[203] *Support schedule for Android OS*. URL: https://endoflife.date/android (Última visita: 30-04-2023).

[204] Joe Sylve. «LiME-Linux memory extractor». En: *Proceedings of the 7th ShmooCon Conference*. 2012. URL: https://github.com/504ensicsLabs/LiME (Última visita: 30-04-2023).

[205] Joe Sylve et al. «Acquisition and analysis of volatile memory from Android devices». En: *Digital Investigation* 8.3-4 (2012), págs. 175-184.

[206] *Tackling the CoronaVirus ransomware attack*. URL: https://www.manageengine.com/log-management/detect-and-mitigate-coronavirus-ransomware-attack.html (Última visita: 30-04-2023).

[207] *Telefonica, other Spanish firms hit in ransomware attack*. URL: https://www.reuters.com/article/us-spain-cyber-idUKKBN1881TJ (Última visita: 30-04-2023).

[208] Szu-Yuan Teng y Che-Yen Wen. «A Forensic Examination of Anonymous Browsing Activities». En: *Forensic Science Journal* 17.1 (2018), págs. 1-8.

[209] *TeslaCrypt ransomware*. URL: https://www.knowbe4.com/teslacrypt-ransomware (Última visita: 30-04-2023).

[210] *The history of ransomware*. URL: https://ransomware.org/what-is-ransomware/the-history-of-ransomware (Última visita: 30-04-2023).

[211] *The Sleuth Kit*. URL: https://www.sleuthkit.org/sleuthkit (Última visita: 30-04-2023).

[212] *The state of ransomware in the US: Report and statistics 2019*. 2019. URL: https://www.emsisoft.com/en/blog/34822/the-state-of-ransomware-in-the-us-report-and-statistics-2019 (Última visita: 30-04-2023).

[213] Warren Thompson. *A forensic analysis of Android mobile private browsing artifacts*. White Paper. SANS Institute, feb. de 2022. URL: https://www.sans.org/white-papers/a-forensic-analysis-of-android-mobile-private-browsing-artifacts (Última visita: 30-04-2023).

[214] Catherine Thorbecke. *Facebook agrees to pay UK fine over Cambridge Analytica scandal while admitting no liability*. 2023. URL: https://abcnews.go.com/Business/facebook-agrees-pay-uk-fine-cambridge-analytica-scandal/story?id=66635145 (Última visita: 30-04-2023).

[215] Joe Tidy. *Mysterious 'Robin Hood' hackers donating stolen money*. 2020. URL: https://www.bbc.com/news/technology-54591761 (Última visita: 30-04-2023).

[216] Muchamad Kukuh Tri, Imam Riadi y Yudi Prayudi. «Forensics acquisition and analysis method of IMO messenger». En: *International Journal of Computer Applications* 179.47 (2018), págs. 9-14.

[217] *Trojan.Dropper*. URL: https://www.malwarebytes.com/blog/detections/trojan-dropper (Última visita: 30-04-2023).

[218] Eran Tromer. *Cryptanalysis of the Gpcode.ak ransomware virus*. 2008. URL: https://rump2008.cr.yp.to/6b53f0dad2c752ac2fd7cb80e8714a90.pdf (Última visita: 30-04-2023).

[219] Nikolaos Tsalis et al. «Exploring the protection of private browsing in desktop browsers». En: *Computers & Security* 67 (2017), págs. 181-197.

[220] Dora Tudor. *All about Conti ransomware*. 2022. URL: https://heimdalsecurity.com/blog/what-is-conti-ransomware (Última visita: 30-04-2023).

[221] Dora Tudor. *DarkSide ransomware 101*. 2022. URL: https://heimdalsecurity.com/blog/darkside-ransomware-101 (Última visita: 30-04-2023).

[222] Rusydi Umar, Imam Riadi y Bashor Fauzan Muthohirin. «Live forensics of tools on Android devices for email forensics». En: *TELKOMNIKA (Telecommunication Computing Electronics and Control)* 17.4 (2019), págs. 1803-1809.

[223] Vladimir Unterfingher. *Ryuk ransomware: Origins, operation mode, mitigation*. 2021. URL: https://heimdalsecurity.com/blog/ryuk-ransomware (Última visita: 30-04-2023).

[224] *Vidar (malware family)*. URL: https://any.run/malware-trends/vidar (Última visita: 30-04-2023).

[225] Timothy Vidas, Chengye Zhang y Nicolas Christin. «Toward a general collection methodology for Android devices». En: *digital investigation* 8 (2011), S14-S24.

[226] *VirusTotal web page*. URL: https://www.virustotal.com (Última visita: 30-04-2023).

[227] *Volatility Framework*. URL: https://github.com/volatilityfoundation/volatility (Última visita: 30-04-2023).

[228] Jules Wang. *Google explains the Pixel 7 is 64-bit only, says 32-bit apps still have roles to play*. 2022. URL: https://www.androidpolice.com/google-pixel-7-pro-64-bit-info (Última visita: 30-04-2023).

[229] Peiying Wang et al. «MimosaFTL: Adding secure and practical ransomware defense strategy to flash translation layer». En: *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*. 2019, págs. 327-338.

[230] Keith D Watson. «The Tor network: A global inquiry into the legal status of anonymity networks». En: *Wash. U. Global Stud. L. Rev.* 11 (2012), pág. 715.

[231] *WCry ransomware analysis*. 2017. URL: https://www.secureworks.com/research/wcry-ransomware-analysis (Última visita: 30-04-2023).

[232] *What is an exploit kit?* URL: https://www.paloaltonetworks.com/cyberpedia/what-is-an-exploit-kit (Última visita: 30-04-2023).

[233] *What is Bad Rabbit ransomware?* 2021. URL: https://www.sitelock.com/blog/what-is-bad-rabbit-ransomware (Última visita: 30-04-2023).

[234] *What is Bad Rabbit ransomware?* URL: https://www.proofpoint.com/us/threat-reference/bad-rabbit (Última visita: 30-04-2023).

[235] *What is Cerber ransomware?* URL: https://www.proofpoint.com/us/threat-reference/cerber-ransomware (Última visita: 30-04-2023).

[236] *What is Conti ransomware?* URL: https://www.veeam.com/glossary/conti-ransomware.html (Última visita: 30-04-2023).

[237] *What is digital forensics?* URL: https://www.eccouncil.org/cybersecurity/what-is-digital-forensics (Última visita: 30-04-2023).

[238] *What is Maze ransomware? Definition and explanation*. URL: https://www.kaspersky.com/resource-center/definitions/what-is-maze-ransomware (Última visita: 30-04-2023).

[239] *What is Petya ransomware?* URL: https://www.proofpoint.com/us/threat-reference/petya (Última visita: 30-04-2023).

[240] *What is RYUK ransomware?* URL: https://www.trendmicro.com/en_us/what-is/ransomware/ryuk-ransomware.html (Última visita: 30-04-2023).

[241] *What we know about the DarkSide ransomware and the US pipeline attack*. 2021. URL: https://www.trendmicro.com/en_us/research/21/e/what-we-know-about-darkside-ransomware-and-the-us-pipeline-attac.html (Última visita: 30-04-2023).

[242] Michael E Whitman y Herbert J Mattord. *Principles of information security*. Cengage learning, 2021.

[243] Primal Wijesekera et al. «The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences». En: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, págs. 1077-1093.

[244] *WinLock ransomware*. URL: https://www.knowbe4.com/winlock-ransomware (Última visita: 30-04-2023).

[245] Songyang Wu et al. «Forensic analysis of WeChat on Android smartphones». En: *Digital investigation* 21 (2017), págs. 3-10.

[246] Songyang Wu et al. «Forensics on Twitter and WeChat using a customised Android emulator». En: *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*. IEEE. 2018, págs. 602-608.

[247] *wxHexEditor hexadecimal editor*. URL: https://www.wxhexeditor.org (Última visita: 30-04-2023).

[248] *X-Ways*. URL: https://www.x-ways.net (Última visita: 30-04-2023).

[249] Haiyu Yang et al. «A tool for volatile memory acquisition from Android devices». En: *IFIP International Conference on Digital Forensics*. Springer. 2016, págs. 365-378.

[250] Seung Jei Yang et al. «Live acquisition of main memory data from Android smartphones and smartwatches». En: *Digital Investigation* 23 (2017), págs. 50-62.

[251] Salessawi Ferede Yitbarek et al. «Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors». En: *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE. 2017, págs. 313-324.

[252] Lojin Bani Younis, Safa Sweda y Ahmad Alzu'bi. «Forensics analysis of private web browsing using Android memory acquisition». En: *2021 12th International Conference on Information and Communication Systems (ICICS)*. IEEE. 2021, págs. 273-278.

[253] Fan Zhou et al. «Dump and analysis of Android volatile memory on Wechat». En: *2015 IEEE International Conference on Communications (ICC)*. IEEE. 2015, págs. 7151-7156.

Índice de figuras

Índice de tablas

Apéndice A

Publicaciones y autorizaciones

- Xosé Fernández-Fuentes, Tomás F. Pena y José C. Cabaleiro, "Digital forensic analysis methodology for private browsing: Firefox and Chrome on Linux as a case study", *Computers & Security*, Volume 115, April 2022, 102626. ISSN: 0167-4048.
  DOI: https://doi.org/10.1016/j.cose.2022.102626.
  Factor de impacto (JCR 2021): 5,105. Q2.
  Categoría: COMPUTER SCIENCE, INFORMATION SYSTEMS. Rango: 42/164.
  Factor de impacto (SJR 2021): 1,726. Q1.
  Categoría: COMPUTER SCIENCE (MISCELLANEOUS). Rango: 23/332.
  Categoría: LAW. Rango: 13/874.
  Contribución: Conceptualización, Metodología, Investigación, Redacción (borrador original).

Digital forensic analysis methodology for private browsing: Firefox and Chrome on Linux as a case study

Author:
Xosé Fernández-Fuentes,Tomás F . Pena,José C. Cabaleiro

Publication: Computers & Security

Publisher: Elsevier

Date: April 2022

Publicaciones aceptadas pendientes de publicación:

- Xosé Fernández-Fuentes, Tomás F. Pena y José C. Cabaleiro, "Recovering from Memory the Encryption Keys used by Ransomware Targeting Windows and Linux Systems", *Proceedings of the Computing Conference 2023*, Springer, e-ISSN: 2367-3389.
  Contribución: Conceptualización, Metodología, Investigación, Redacción (borrador original).

SPRINGER NATURE
Author's Reuse Rights:

1. The Publisher acknowledges that the Author retains the ability to copy, distribute or otherwise reuse the Contribution, without the requirement to seek specific prior written permission from the Publisher, ("Reuse") subject to and in accordance with the following provisions:
   (a) Reuse of the Contribution or any part of it is permitted in a new edition of the Work or in a new monograph or new textbook written by the same Author provided that in each case the new work is published by the Publisher under a publishing agreement with the Publisher; and
   (b) **Reuse of the Version of Record (as defined below) of the Contribution or any part of it is permitted in a thesis written by the same Author, and the Author is entitled to make a copy of the thesis containing content of the Contribution available in a repository of the Author's academic institution**; and
   (c) any other Reuse of the Contribution in a new book, book chapter, proceedings or journal article, whether published by the Publisher or by any third party, is limited to three figures (including tables) or single text extracts of less than 400 words; and
   (d) any further Reuse of the Contribution is permitted only to the extent and in so far as is reasonably necessary: (i) to share the Contribution as a whole to no more than 10 research colleagues engaged by the same institution or employer as the Author for each colleague's personal and private use only; (ii) for classroom teaching use by the Author in their respective academic institution provided that this does not permit inclusion of any of the

Esta tesis aplica técnicas de análisis forense a dos importantes problemas relacionados con la privacidad y la seguridad: la eficacia del modo de navegación privada y los ataques de ransomware.

Por un lado, se desarrolló una metodología para evaluar el correcto funcionamiento del modo privado integrado en la mayoría de los navegadores web actuales. Las pruebas realizadas con Linux y con Android muestran que es posible recuperar información sensible incluso después de reiniciar los dispositivos.

Por otro lado, se estudió una las amenazas más importantes de los últimos años: el ransomware. Un tipo de malware que bloquea el acceso a los ficheros de un equipo hasta que se paga un rescate. Tras analizar diferentes muestras, se desarrollaron dos herramientas que permiten recuperar la clave de cifrado directamente de memoria.