

# Seguridad en Raspberry Pi

Guía de bastionado para entornos empresariales

The logo of the Universitat Oberta de Catalunya (UOC), consisting of the letters 'UOC' in a stylized, bold, blue font.

**Sergi Vaghi García**

Trabajo Final de Máster  
Seguridad empresarial

**Nombre Tutor/a de TF**

Rafael Páez Reyes

**Profesor/a responsable de  
la asignatura**

Víctor García Font

**13/06/2023**

Universitat Oberta  
de Catalunya

---



Esta obra está sujeta a una licencia de  
Reconocimiento-NoComercial-

SinObraDerivada [3.0 España de Creative  
Commons](https://creativecommons.org/licenses/by-nc-nd/3.0/es/)

## FICHA DEL TRABAJO FINAL

<b>Título del trabajo:</b>	<i>Seguridad en Raspberry Pi. Guía de bastionado para entornos empresariales</i>
<b>Nombre del autor:</b>	Sergi Vaghi García
<b>Nombre del consultor/a:</b>	Rafael Páez Reyes
<b>Nombre del PRA:</b>	Víctor García Font
<b>Fecha de entrega (mm/aaaa):</b>	06/2023
<b>Titulación o programa:</b>	Máster Universitario en Ciberseguridad y Privacidad
<b>Área del Trabajo Final:</b>	Seguridad Empresarial
<b>Idioma del trabajo:</b>	Castellano
<b>Palabras clave</b>	Bastionado, Raspberry, <i>Hardening</i>

### Resumen del Trabajo

El trabajo final de máster ha consistido en la elaboración de una guía de bastionado de Raspberry Pi y su aplicación en un entorno empresarial, utilizando como ejemplo la securización de una Raspberry Pi 4 como servidor web.

El proyecto se divide en dos partes: la primera parte, centrada en la elaboración de la guía de bastionado de Raspberry Pi, desde un enfoque de defensa en profundidad. Y la segunda parte, consistente en la aplicación de la guía y de medidas concretas para bastionar la Raspberry Pi en un entorno empresarial y para su uso como servidor web.

Finalmente, se han logrado conseguir los objetivos planteados en un inicio, obteniendo como resultado una completa guía teórico-práctica sobre bastionado de Raspberry Pi, y que puede ser aplicada incluso en entornos profesionales y empresariales para la protección de este tipo de sistemas.

## **Abstract**

The master's thesis consisted of developing a guide for securing a Raspberry Pi and its application in a business environment, using a Raspberry Pi 4 as an example for setting up a secure web server.

The Project was divided into two parts: the first one focused on creating the guide for securing a Raspberry Pi using a defense-in-depth approach. The second one involved applying the guide and specific measures for securing the Raspberry Pi in a business environment and using it as a web server.

The Project was successfully achieved its objectives, resulting in a comprehensive theoretical and practical guide for Raspberry Pi hardening that can be applied even in professional and business settings to protect these types of systems.

# Índice de contenidos

<b>1. Introducción.....</b>	<b>1</b>
<b>1.1. Problema a resolver.....</b>	<b>1</b>
<b>1.2. Objetivos del trabajo .....</b>	<b>1</b>
<b>1.3. Metodología.....</b>	<b>2</b>
<b>1.4. Planificación de tareas .....</b>	<b>4</b>
<b>1.5. Planificación temporal.....</b>	<b>5</b>
<b>1.6. Análisis de riesgos .....</b>	<b>6</b>
1.6.1. Objetivos del trabajo desmedidos.....	6
1.6.2. Objetivos del trabajo insuficientes .....	7
1.6.3. Tiempo insuficiente para realizar las distintas entregas.....	8
1.6.4. Pérdida de archivos y versiones del trabajo .....	8
1.6.5. Fallos de <i>hardware</i> durante el proyecto.....	9
1.6.6. Fallos de <i>software</i> durante el proyecto.....	10
1.6.7. Errores en la citación de recursos utilizados.....	10
<b>1.7. Recursos y presupuesto del proyecto.....</b>	<b>11</b>
<b>1.8. Estado del arte .....</b>	<b>12</b>
1.8.1. Bastionado de sistemas basados en Raspberry Pi.....	12
1.8.2. Herramientas para la defensa en profundidad.....	13
<b>1.9. Impacto ético, social y ambiental.....</b>	<b>14</b>
<b>1.10. Estructura del trabajo.....</b>	<b>15</b>
<b>2. Seguridad en sistemas Raspberry.....</b>	<b>17</b>
<b>2.1. Mecanismos de prevención .....</b>	<b>17</b>
2.1.1. Gobierno de TI y ciberseguridad .....	17
2.1.2. Políticas de seguridad .....	19
2.1.3. Concienciación de la seguridad.....	20
<b>2.2. Defensa en profundidad.....</b>	<b>21</b>
2.2.1. Seguridad física .....	22
2.2.1.1. <i>Configuración básica inicial</i> .....	24
2.2.1.2. <i>Configuración de arranque</i> .....	24

2.2.1.3. Bloqueo de puertos y conexiones externas .....	27
2.2.1.4. Bloqueo y cierre de sesiones.....	32
2.2.1.5. Protección y cifrado de la microSD.....	33
2.2.2. Seguridad perimetral .....	42
2.2.2.1. Configuración de un cortafuegos.....	43
2.2.2.2. Fail2ban como soporte a UFW.....	46
2.2.2.3. Securización de SSH.....	47
2.2.2.4. Servidor VPN.....	49
2.2.3. Seguridad de red.....	52
2.2.3.1. Protección contra suplantación de identidad.....	52
2.2.3.2. Raspberry Pi como VPN-client con WireGuard.....	55
2.2.3.3. Redes de área local virtuales (VLAN) .....	58
2.2.3.4. Instalación de un IDS en la red.....	59
2.2.4. Seguridad del dispositivo.....	63
2.2.4.1. ClamAV como solución EPP.....	64
2.2.4.2. OSSEC como solución HIDS.....	67
2.2.4.3. OSSEC+ como solución EDR.....	72
2.2.4.4. Atomic OSSEC como solución XDR .....	73
2.2.4.5. Cortafuegos de dispositivo .....	74
2.2.5. Seguridad de la capa de aplicación .....	75
2.2.5.1. Gestión de permisos y atributos .....	75
2.2.5.2. Gestión de identidades y accesos .....	79
2.2.5.3. Cortafuegos a nivel de aplicación (WAF) .....	82
2.2.5.4. Application Armor (AppArmor) .....	83
2.2.6. Seguridad de los datos.....	84
2.2.6.1. Clasificación de la información .....	84
2.2.6.2. Cifrado de los datos.....	86
2.2.6.3. Medidas de recuperación de la información.....	89
<b>3. Bastionado de un entorno profesional.....</b>	<b>94</b>
<b>3.1. Entorno profesional: servidor LEMP.....</b>	<b>94</b>
<b>3.2. Configuración segura de la Raspberry Pi.....</b>	<b>95</b>
3.2.1. Seguridad física del servidor .....	95
3.2.2. Seguridad perimetral y de red .....	96
3.2.3. Seguridad del dispositivo.....	97
3.2.4. Seguridad de la capa de aplicación.....	101
3.2.5. Seguridad de los datos.....	101

<b>3.3. Bastionado de Nginx .....</b>	<b>105</b>
3.3.1. Aspectos generales del servidor .....	105
3.3.1.1. <i>Desactivar módulos de Nginx no utilizados</i> .....	105
3.3.1.2. <i>Configuración de server blocks</i> .....	106
3.3.1.3. <i>Eliminar información del servidor</i> .....	107
3.3.1.4. <i>Cabeceras de seguridad</i> .....	108
3.3.1.5. <i>Limitación de los métodos HTTP</i> .....	109
3.3.1.6. <i>Protección contra DoS y Buffer Overflow</i> .....	109
3.3.1.7. <i>Registros de acceso y errores del servidor</i> .....	110
3.3.1.8. <i>Otras recomendaciones</i> .....	111
3.3.2. Protocolo HTTPS .....	112
3.3.3. <i>Web Application Firewall (WAF)</i> .....	115
<b>3.4. Configuración segura de MariaDB (MySQL) .....</b>	<b>121</b>
<b>3.5. Configuración segura de PHP .....</b>	<b>124</b>
3.5.1. Usar versiones actualizadas de PHP en Raspberry Pi OS .....	124
3.5.2. Configuración segura de PHP .....	125
<b>4. Reflexiones finales .....</b>	<b>128</b>
<b>4.1. Conclusiones .....</b>	<b>128</b>
<b>4.2. Futuras líneas de trabajo e investigación.....</b>	<b>129</b>
<b>5. Glosario.....</b>	<b>131</b>
<b>6. Referencias .....</b>	<b>135</b>

# Índice de figuras

Figura 1. Diagrama de Gantt.....	5
Figura 2. Esquema de la defensa en profundidad.....	22
Figura 3. Selección de opciones avanzadas (raspi-config).....	25
Figura 4. Selección de orden de arranque (raspi-config).....	26
Figura 5. Selección del orden de arranque de microSD o red (raspi-config)....	26
Figura 6. Configuración de bashrc incluyendo TMOUT.....	32
Figura 7. Terminal bloqueada por vlock.....	33
Figura 8. Módulos de initramfs.....	36
Figura 9. Initramfs montado correctamente.....	37
Figura 10. Configuración de config.txt con ititramfs.....	37
Figura 11. Configuración de fstab con la nueva ubicación.....	38
Figura 12. Configuración de crypttab.....	38
Figura 13. Ejemplo de respuesta del benchmark de cryptsetup.....	39
Figura 14. Reducción del tamaño de la partición root.....	39
Figura 15. Cálculo de la suma de control.....	40
Figura 16. Petición de contraseña para desbloqueo de SD.....	42
Figura 17. Configuración básica de jail.conf (fail2ban).....	47
Figura 18. Jaula para el servicio SSH.....	47
Figura 19. Estado activo del túnel VPN con WireGuard.....	51
Figura 20. Configuración de wg0.conf (Servidor).....	58
Figura 21. Ubicación de un NIDS en una posible red corporativa.....	60
Figura 22. Configuración por defecto de suricata.yaml (Suricata).....	61
Figura 23. Oinkmaster.conf modificado para cargar reglas concretas.....	62
Figura 24. Ejemplo de escaneo automatizado con ClamAV y crontab.....	97
Figura 25. Resultado de un escaneo con LMD.....	100
Figura 26. Ejemplo de escaneo automatizado con LMD y crontab.....	100
Figura 27. Subida de backup a Google Drive con Rclone.....	104
Figura 28. Redundancia de backups en Google Drive.....	105
Figura 29. Versión del servidor Nginx expuesta.....	107
Figura 30. Directiva server_tokens deshabilitada.....	107
Figura 31. Directiva para deshabilitar Nginx Server Signature.....	108
Figura 32. Cabeceras de seguridad.....	108



Figura 33. Limitación de métodos HTTP .....	109
Figura 34. Directivas para limitar el tamaño del búfer .....	109
Figura 35. Directivas para el control de Timeouts .....	110
Figura 36. Registro de errores iguales o superiores a warn .....	110
Figura 37. Dominio configurado en server_name.....	112
Figura 38. Certificado SSL/TLS válido .....	114
Figura 39. SSL Settings en nginx.conf .....	115
Figura 40. Carga del módulo ModSecurity en nginx.conf.....	118
Figura 41. Inicialización del motor de reglas .....	120
Figura 42. Contenido de main.conf .....	120
Figura 43. Configuración de server block con ModSecurity activo .....	120
Figura 44. Configuración de MariaDB para evitar SQLi .....	124

# 1. Introducció

## 1.1. Problema a resolver

Raspberry Pi nació como un ambicioso proyecto de creación de una placa que albergase todo lo necesario para funcionar como lo hace un ordenador convencional y que, a su vez, fuese lo más barato posible. El objetivo tras ello fue hacer llegar de forma fácil y barata el conocimiento y la experimentación de ordenadores a los jóvenes [1]. Sin embargo, Raspberry Pi ha sido un éxito rotundo en todos los sentidos. Tanto es así, que hoy en día no son pocas las empresas que adoptan alguno de los modelos de Raspberry Pi para distintas aplicaciones.

Por lo anterior, cabría esperar la existencia de multitud de guías de securización de sistemas basados en Raspberry Pi. La realidad, no obstante, es que existe un gran vacío documental en lo que a bastionado de sistemas con arquitectura RISC se refiere.

## 1.2. Objetivos del trabajo

El presente trabajo final de máster tiene por objetivo general la redacción de una guía completa sobre bastionado de sistemas que utilizan como base una Raspberry Pi. De esta forma, el producto final pretende cumplir las veces de marco referencial en lo que se refiere a la securización de este tipo de sistemas.

El anterior objetivo general se divide a su vez en varios objetivos específicos, expuestos a continuación:

- I. La investigación de todas las herramientas existentes para el bastionado de sistemas informáticos desde la perspectiva de la defensa en profundidad.
- II. La adaptación de tales herramientas al caso concreto de sistemas que utilizan Raspberry Pi OS.
- III. La búsqueda, enumeración y guía de configuración de las soluciones de seguridad más adecuadas para el bastionado de sistemas RISC.
- IV. La confección y construcción de un entorno real de prácticas mediante el uso de hardware dedicado, en el que poder realizar las configuraciones y pruebas necesarias.

- V. La puesta en práctica de la guía en determinados escenarios que pueden darse en entornos laborales, como son el uso de servidores que corren en sistemas Raspberry Pi.
- VI. Hacer fácil y práctico el uso seguro de sistemas RISC en entornos empresariales e industriales.
- VII. El cumplimiento de las exigencias académicas inherentes a la asignatura, basado en las entregas parciales y entrega final de la memoria y bajo los estándares de calidad de la UOC.
- VIII. La correcta realización de la video-presentación sobre el producto final del trabajo realizado, así como de la defensa ante el tribunal de evaluación.

Se pretende mediante este trabajo, por tanto, cubrir la necesidad de una guía detallada sobre el bastionado de sistemas que puedan utilizar como hardware una Raspberry Pi o similares. Pudiendo incluso ser éstos utilizados en entornos empresariales con las máximas garantías de seguridad posibles.

### 1.3. Metodología

En su inicio, la estrategia planteada para la realización del trabajo final de máster estaba enfocada en el *hardening* de un servidor NAS basado en Raspberry Pi, como podría encontrarse en un entorno empresarial de una PYME. Sin embargo, y tras analizar las posibilidades del caso anterior, finalmente se ha decidido por abrir el foco de trabajo. De esta forma, la estrategia finalmente planteada es más ambiciosa, ya que busca la construcción de una guía completa y general, que pueda servir para cualquier escenario en el que se use una Raspberry Pi o similares. Asimismo, se ha decidido incluir también casos de aplicaciones prácticas, como la securización de un servidor web en una empresa mediante la propia guía.

Así, el trabajo puede diferenciarse en dos grandes vertientes. La primera de ellas, enfocada plenamente en el bastionado de sistemas de bajo costo (de tipo SBC) como es el caso de una Raspberry Pi 4. Y la segunda, en cambio, dedicada a la puesta en práctica de todo lo recogido en la primera parte en un entorno empresarial plausible.

Enfocándonos en cada una de las vertientes anteriormente comentadas, nos encontramos que el trabajo combina, en su primera vertiente (la guía), aspectos teóricos y técnicos con muestras de código y pasos a seguir para herramientas y controles concretos. Respecto a la segunda vertiente, ésta cuenta con un enfoque totalmente práctico, dado que se centra en reproducir lo recogido por la guía, pero en entornos empresariales asimilables a la realidad. Al igual que en la primera vertiente, en esta segunda también podemos encontrar demostraciones de configuración de herramientas y controles. A lo anterior, se pueden sumar algunas pruebas de concepto, cuyo objetivo principal es la verificación del correcto funcionamiento de determinadas herramientas y controles anteriormente configurados.

Con tal de cumplir los objetivos de investigación anteriormente planteados, las principales fuentes de recursos previstas son:

- Documentos científicos, como *papers*, tesis doctorales, u otros trabajos de fin de máster o trabajos de fin de grado.
- Documentación oficial de las herramientas y soluciones utilizadas, como son Raspberry Pi, Raspberry Pi OS, Firewalls, EDR, HIDS, o NIDS, entre otros.
- Repositorios de GitHub que puedan contener información útil y relevante para los objetivos del trabajo.
- Literatura de interés, como guías de bastionado de sistemas basados en GNU/Linux, guías de administración de servidores, u otros que puedan servir como inspiración.
- Páginas web de recursos como Gartner u otras.
- Recursos de aprendizaje propios de la UOC.

## 1.4. Planificación de tareas

Núm.	Tarea	Inicio	Fin	Dedicación
<b>1</b>	<b>Plan de Trabajo</b>			
1.1	Problema a resolver	03/03	04/03	6
1.2	Objetivos del trabajo	04/03	05/03	5
1.3	Metodología	04/03	05/03	3
1.4	Tareas a realizar	05/03	06/03	6
1.5	Diagrama de Gantt	05/03	05/03	4
1.6	Análisis de Riesgos	06/03	07/03	5
1.7	Aproximación al estado del arte	07/03	09/03	10
1.8	Impacto ético, social y ambiental	09/03	10/03	6
1.9	Recursos y presupuesto del proyecto	10/03	10/03	3
1.10	Próximas líneas de trabajo e investigación	10/03	10/03	2
<b>1.11</b>	<b>Entrega PEC 1</b>	<b>01/03</b>	<b>14/03</b>	<b>Entrega</b>
<b>2</b>	<b>Desarrollo general de la guía</b>			
2.1	Gobierno TI/Ciberseguridad	15/03	17/03	10
2.2	Políticas de seguridad	18/03	20/03	10
2.3	Concienciación	21/03	22/03	5
2.4	Defensa en profundidad	23/03	10/04	50
2.4.1	Seguridad física			
2.4.2	Seguridad perimetral			
2.4.3	Seguridad de red			
2.4.4	Seguridad del dispositivo			
2.4.5	Seguridad de software			
2.4.6	Seguridad de datos			
<b>2.5</b>	<b>Entrega PEC 2</b>	<b>15/03</b>	<b>11/04</b>	<b>Entrega</b>
<b>3</b>	<b>Demostración en entornos empresariales</b>			
3.1	Servidor LEMP	12/04	08/05	75
3.1.1	Preparación del laboratorio	12/04	13/04	15
3.1.2	Bastionado en profundidad	14/04	28/04	40
3.1.3	Configuraciones concretas para servidores web	28/04	08/05	30
<b>3.2</b>	<b>Entrega PEC 3</b>	<b>12/04</b>	<b>09/05</b>	<b>Entrega</b>
<b>4</b>	<b>Memoria final</b>			
4.1	Revisión y corrección de apartados	10/05	24/05	30
4.2	Redacción de conclusiones	25/05	01/06	25
4.3	Revisión final del documento	02/06	12/06	20
<b>4.4</b>	<b>Entrega PEC 4</b>	<b>10/05</b>	<b>13/06</b>	<b>Entrega</b>
<b>5</b>	<b>Presentación y defensa</b>			
5.1	Grabación de video-presentación	14/06	19/06	10
<b>5.2</b>	<b>Entrega video-presentación</b>	<b>14/06</b>	<b>20/06</b>	<b>Entrega</b>
5.3	Preparación de la defensa	21/06	29/06	15
<b>5.4</b>	<b>Defensa del TFM</b>	<b>26/06</b>	<b>30/06</b>	<b>Entrega</b>

# 1.5. Planificación temporal

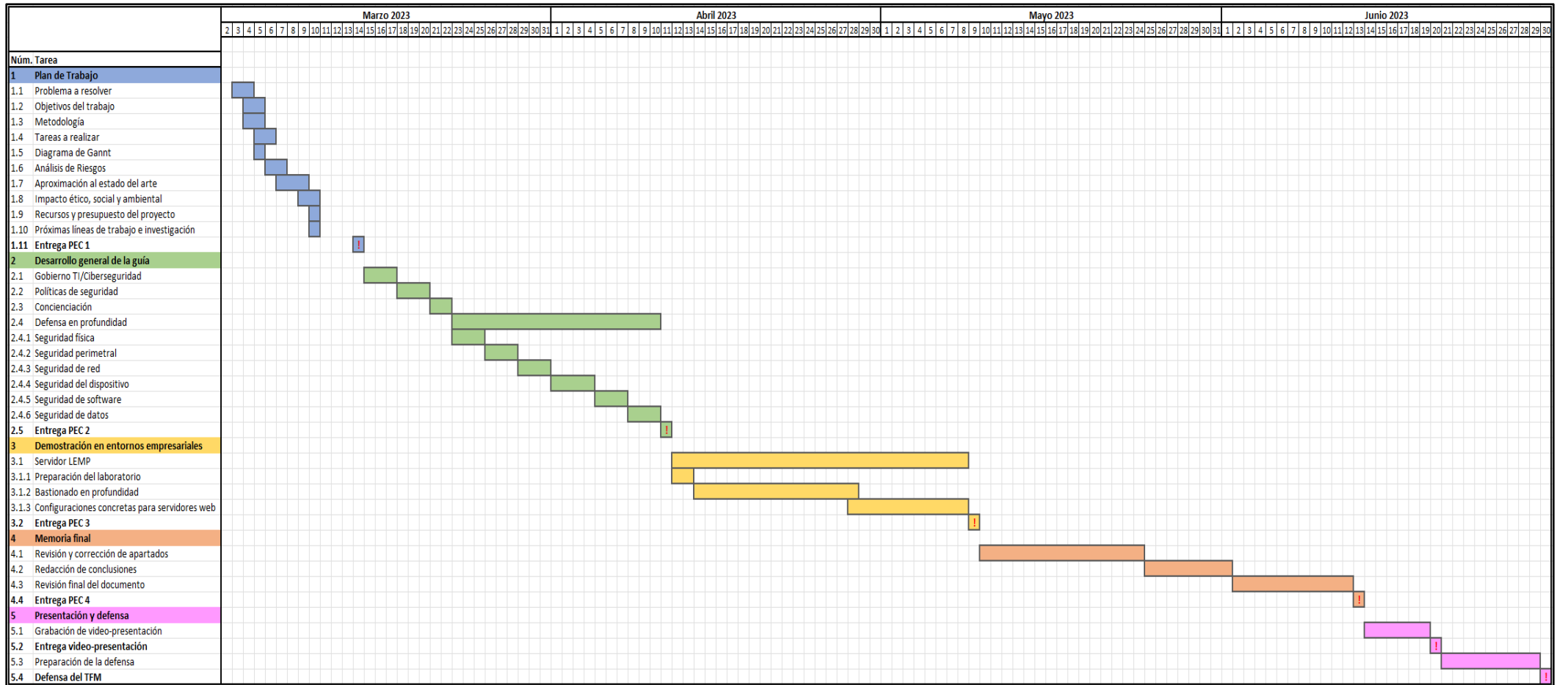


Figura 1. Diagrama de Gantt

## 1.6. Análisis de riesgos

El presente apartado tiene por objetivo el análisis y posterior gestión de todos aquellos riesgos que puedan ser considerados, cuya materialización podrían afectar parcial o totalmente al éxito del trabajo. Para ello, se han considerado un total de siete riesgos, analizados individualmente mediante la descripción de cada uno. Asimismo, se presentan para cada riesgo un seguido de contramedidas o controles, cuyo objetivo final es la salvaguarda del trabajo frente a tales amenazas.

A continuación, se exponen las principales amenazas identificadas en una tabla resumen, en la que también podemos ver la probabilidad de cada riesgo (baja, media o alta), así como el impacto que podría suponer su materialización (también bajo, medio o alto). A partir de los dos valores, se han extraído las puntuaciones de cada riesgo. Para este cálculo, se ha asumido que no ha habido tratamiento de riesgos, por lo que la vulnerabilidad del proyecto respecto de cada riesgo es máxima y se ha obviado en el cálculo de la tabla.

<b>Amenazas</b>	<b>Probabilidad</b>	<b>Impacto</b>	<b>Riesgo</b>
Objetivos desmedidos	Baja	Medio	<b>Medio</b>
Objetivos insuficientes	Baja	Medio	<b>Medio</b>
Tiempo insuficiente	Alta	Alto	<b>Alto</b>
Pérdida de archivos y de versiones	Media	Alto	<b>Alto</b>
Fallos del <i>hardware</i>	Baja	Alto	<b>Medio</b>
Fallos del <i>software</i>	Alta	Alto	<b>Alto</b>
Error de citación	Baja	Alto	<b>Medio</b>

### 1.6.1. Objetivos del trabajo desmedidos

#### Descripción:

Cuando se trata de realizar un trabajo académico de cierta importancia y magnitud, uno de los grandes riesgos que puede aparecer es la excesiva ambición en los objetivos marcados en la fase inicial. Unos objetivos

inabarcables en tiempo y forma pueden suponer un producto final incompleto y carente de la calidad necesaria.

Controles:

La planificación es el control por excelencia de cara a mitigar de forma efectiva el riesgo anteriormente descrito. Con una correcta planificación de objetivos y planificación temporal se reducen en gran medida las probabilidades de materialización del riesgo.

De igual forma, es interesante considerar la revisión de bibliografía y trabajos equivalentes, con el objetivo de calcular de forma más acertada el tiempo requerido para completar cada hito. Finalmente, las reuniones y consultas al tutor se vuelven un factor clave en la limitación del riesgo descrito.

1.6.2. Objetivos del trabajo insuficientes

Descripción:

En la línea del anterior riesgo, la planificación de objetivos insuficientes o poco ambiciosos en las fases iniciales del proyecto causan que, de forma directa, la calidad del producto final se reduzca notablemente. La falta de objetivos podría suponer no alcanzar los mínimos exigidos por el plan docente para la superación de la asignatura.

Controles:

Al igual que el anterior riesgo, la planificación vuelve a ser el control predeterminado para la reducción del riesgo por objetivos insuficientes. La viabilidad del proyecto depende en gran medida de la fase inicial del mismo, en el que la planificación tanto temporal como de tareas es vital.

En este aspecto, el punto de vista experto del tutor gana más fuerza si cabe que en el caso anterior, ya que éste cuenta con el bagaje suficiente como para recomendar cambios en los objetivos del proyecto. De nuevo, las reuniones y consultas con el tutor, especialmente en la fase inicial, se vuelven un factor determinante.



### 1.6.3. Tiempo insuficiente para realizar las distintas entregas

#### Descripción:

El tiempo para realizar las distintas entregas es ajustado y bien medido. Esto hace que la incorrecta planificación o no cumplir con una planificación bien hecha pueda converger en la falta de tiempo y, por tanto, la entrega de material incompleto o carente de contenidos. Lo anterior afecta gravemente la calidad del trabajo y puede conllevar la pérdida de puntuación, tanto de las entregas parciales como de la nota final del trabajo. En el peor de los casos, podría causar la entrega fuera de plazo de alguna de las PEC y/o la no superación de la asignatura.

#### Controles:

Si bien la planificación temporal y de tareas juega un papel importante, no es el único factor para tener en cuenta. Como se ha expuesto anteriormente, aun existiendo una correcta planificación, el riesgo puede acabar materializándose si ésta no se cumple. Por ello, la organización y planificación de objetivos diarios se constituye como el control con mayor relevancia para gestionar el riesgo. Por descontado, dicha organización diaria debe atender en todo momento a la planificación realizada en la fase inicial del proyecto.

### 1.6.4. Pérdida de archivos y versiones del trabajo

#### Descripción:

La pérdida de archivos recopilados a lo largo del proyecto, así como de archivos que contienen versiones del trabajo, es uno de los riesgos que con mayor frecuencia se tiene presente. Ello puede causar un daño al trabajo realizado tal que podría significar la no entrega del proyecto y por ende la no superación de la asignatura.

#### Controles:

La salvaguarda seleccionada para gestionar el riesgo es la política de copias de seguridad. Concretamente, se utilizará la famosa regla del 3-2-1, que consiste en distribuir de la forma más inteligente y eficiente las copias de seguridad del documento principal en el que se desarrolla el proyecto. Así, todo el directorio correspondiente al proyecto (y que contiene absolutamente todos los archivos

de éste) será copiado un total de tres veces una vez por semana. Estas tres copias serán guardadas en dos medios diferentes: dos copias guardadas en almacenamiento *in situ* (ordenador personal y disco duro externo); y una copia guardada en la nube Google Drive de Google Inc, (dentro de la licencia ofrecida por la UOC).

Por otro lado, es importante destacar que la política de copias de seguridad no conlleva únicamente la copia de directorios o archivos bajo la regla 3-2-1. La política incluye tanto el proceso anteriormente descrito, como la revisión de copias y viabilidad del proceso de restauración.

Respecto a lo almacenado en la microSD de la Raspberry Pi, se deberán realizar copias de seguridad periódicas mediante herramientas dedicadas a ello, como es el caso de Win32 Disk Imager en Windows o ApplePi-Baker en macOS.

#### 1.6.5. Fallos de *hardware* durante el proyecto

##### Descripción:

Para el proyecto está previsto el uso de *hardware*, en especial para el montaje y puesta en funcionamiento del laboratorio de pruebas. Es en esta fase del proyecto en la que los fallos en los dispositivos físicos pueden causar retrasos e inconvenientes, pudiendo llegar a suponer la pérdida de alguno de los dispositivos. Lo anterior pone en peligro una de las vertientes fundamentales del trabajo.

##### Controles:

Aunque el riesgo pone en peligro una de las partes fundamentales del trabajo (la parte práctica de implementación), existen pocas posibilidades en lo que a medidas de control se refiere. Para la realización del trabajo se cuenta con una única unidad de cada elemento utilizado.

Si bien existe redundancia en lo que a ordenadores personales se refiere (pues se cuenta con un ordenador de sobremesa y un ordenador portátil), no es el caso para el resto de los componentes. Esto es porque, debido a los recientes problemas de manufactura de microchips, el precio de los dispositivos como la Raspberry Pi se han visto incrementados en muchos casos en un 200%. Por ello,

se descarta desde un inicio la existencia de redundancia de los dispositivos de *hardware* dedicados al laboratorio.

#### 1.6.6. Fallos de *software* durante el proyecto

##### Descripción:

El riesgo comprende tanto fallos u errores producidos por cualquier *software* utilizado durante el proyecto (en muchos casos esperables para los que las TIC son nuestra pasión), como incompatibilidades entre distintos *software* o herramientas instaladas en el dispositivo o red.

##### Controles:

Hoy en día existen muchas herramientas que buscan realizar funciones similares y con distintos grados de compatibilidad. Por ello, es imprescindible que exista una fase de documentación que permita identificar las distintas herramientas existentes para cada cometido, y poder seleccionar aquellas que se ajustan a las necesidades del trabajo.

Respecto a los fallos u errores producidos por el *software* utilizado, el control más eficaz es la correcta gestión de errores, en la que la búsqueda de éstos en los repositorios y sitios web de documentación oficiales debe ser la primera opción a tener en cuenta.

#### 1.6.7. Errores en la citación de recursos utilizados

##### Descripción:

Los errores en la citación de recursos utilizados para la consulta y redacción del trabajo comprenden dos tipos. Por un lado, errores formales a la hora de citar fuentes, que afectan a la valoración de los criterios formales. Y, por otro lado, la existencia de textos de terceros cuya cita no ha sido introducida por olvido u error humano, lo cual constituye plagio académico y, por ende, la no superación de la asignatura (además de otras consecuencias derivadas).

##### Controles:

En lo que respecta a la mitigación del riesgo, es imprescindible que exista, durante todas las fases del proyecto, una correcta gestión de la bibliografía consultada y, en especial, de la bibliografía utilizada. Para ello, es de gran ayuda

herramientas propias de gestión de bibliografía de editores de texto como Microsoft Word. De igual forma, pueden utilizarse otras herramientas de soporte de terceros, como es el caso de Mendeley. Las herramientas mencionadas son efectivas tanto a la hora de gestionar las referencias (y evitar no introducir referencias utilizadas), como a la hora de citar sin errores formales.

### 1.7. Recursos y presupuesto del proyecto

En este apartado se presenta el listado de recursos, principalmente de *hardware*, que serán necesarios para la ejecución de la fase práctica del proyecto. La siguiente tabla muestra los elementos necesarios, así como el precio de cada recurso y el total del presupuesto para todo el proyecto.

En cuanto a los recursos y el presupuesto para *software*, no existe un presupuesto previsto para ello. Esto es porque la intención inicial del proyecto es la de bastionar un sistema Raspberry Pi con el mínimo costo posible. Por ello, las principales herramientas previstas para alcanzar tal cometido son de código abierto y, por tanto, libres en cuanto a modificación y distribución se refiere.

Descripción	Precio	Imagen
Raspberry Pi 4 Modelo B (4GB RAM)	185 €	
SSD Samsung 860 EVO (500GB)	60 €	
MicroSD Samsung EVO Select (64GB)	10,69 €	

Memoria USB SanDisk Ultra (128 GB)	13,99 €	
Ordenador portátil personal	500 €	
<b>Presupuesto total</b>	<b>769, 68 €</b>	

### 1.8. Estado del arte

Tras haber realizado una primera aproximación de búsqueda de literatura científica existente en relación a la securización de Raspberry Pi, el presente apartado tiene por misión esbozar la situación actual del área de estudio.

Para ello, el apartado se divide en dos subapartados. El primero, enfocado en explorar todos aquellos recursos que puedan llegar a tratar –de forma directa o indirecta– el bastionado de Raspberry Pi. El segundo, en cambio, dedicado a aportar un primer listado de herramientas que puedan ser de gran utilidad para la estrategia escogida: la defensa en profundidad.

No obstante, cabe indicar que ello no deja de ser una aproximación en fase inicial del proyecto. Cabe esperar, por tanto, desarrollo y cambio de lo expuesto en este apartado a lo largo del proyecto.

#### 1.8.1. Bastionado de sistemas basados en Raspberry Pi

En lo que a seguridad de sistemas Raspberry se refiere, ciertamente existe muy poca literatura o artículos científicos que aborden la problemática de forma exclusiva o concreta. Como se ha indicado al inicio de esta primera fase, uno de los objetivos del proyecto es cubrir ese agujero documental.

### Materiales enfocados en Raspberry:










Uno de los materiales de mayor valor que aborda la seguridad en Raspberry es la documentación de la propia *Raspberry Pi Foundation*. En ella podemos encontrar desde los pasos necesarios para poder cambiar la contraseña por defecto del dispositivo, hasta consejos y pasos para mejorar la seguridad de SSH [2]. Si bien la documentación está muy bien explicada y es fácil de seguir, sólo se centra en ciertos aspectos de seguridad básica y otros de seguridad perimetral, como SSH o Firewalls. Esto hace que sea un buen punto de inicio, pero está lejos de ser suficiente para los objetivos del proyecto. Otro de los pocos materiales que trabajan la securización de sistemas Raspberry Pi es el *paper* “*Assessment and Hardening of IoT Development Boards*” de Alfandi, Hasan y Balbahaith [3]. La conferencia trata la gestión de la seguridad en dispositivos IoT y usando como ejemplo una Raspberry Pi con el sistema operativo Raspberry Pi OS. Por ello, se posiciona como el segundo de los materiales de referencia para este proyecto.

### Materiales extrapolables a la seguridad en Debian/Raspberry:

Por lo anterior, la mayor cantidad de recursos académicos en lo que al problema a tratar se refiere son aquellos que, si bien tratan la seguridad en otros tipos de sistemas, muchas de las herramientas y prácticas son extrapolables a la seguridad en Raspberry y Raspberry Pi OS. Dentro de este grupo de recursos, los de mayor importancia son las guías desarrolladas por el Centro Criptológico Nacional de configuración segura para CentOS 8 y de securización de sistemas basados en Debian [4], [5]. Por otro lado, podemos encontrar guías de securización de sistemas Linux en general tanto en formato libro [6], como en formato repositorio de GitHub [7] y vídeo [8].

#### 1.8.2. Herramientas para la defensa en profundidad

A continuación, se listan algunas de las herramientas que podrían llegar a ser de gran utilidad en el desarrollo del trabajo. Tal y como se ha indicado anteriormente, están clasificadas en las distintas capas diferenciadas por la defensa en profundidad.

Seguridad perimetral	
	
Seguridad de red	
	
Seguridad del dispositivo	
	
Seguridad de software	
	 Open Source Web Application Firewall
Seguridad de datos	
 Linux Unified Key Setup	

### 1.9. Impacto ético, social y ambiental

Uno de los principales objetivos del proyecto es hacer divulgación sobre la posibilidad de securizar sistemas basados en Raspberry Pi o similares. Como se ha indicado en apartados anteriores, los mini PC como Raspberry Pi tienen grandes ventajas tanto en su coste de producción (y precio final) como en su consumo. Por ello, todo lo que este proyecto recoge va en consonancia con los Objetivos de Desarrollo Sostenible y, en especial, con los Objetivos nº 9 (industria, innovación e infraestructura) y nº 12 (producción y consumo responsables).

Respecto al impacto derivado de la realización de este trabajo final de máster, se considera que el hecho de facilitar la adopción de sistemas basados en Raspberry (mediante la guía de bastionado) puede generar un impacto muy positivo en lo que a sostenibilidad se refiere. Recordemos que, tal y como se ha indicado en el párrafo anterior, los sistemas Raspberry Pi o similares cuentan con un gasto energético muy inferior a otros sistemas convencionales. Por tanto, el trabajo final busca también que las empresas puedan adoptar modelos de sistemas más sostenibles.

Por otro lado, y con respecto a la dimensión de la diversidad de género, raza, religión, orientación sexual, funcional, etnia o ideología, se considera que el presente trabajo no tiene impacto alguno. Tras considerar el alcance y los objetivos del trabajo, así como su resultado, que es puramente técnico, se ha llegado a la conclusión de que éste no cuenta con ningún aspecto que pueda acarrear impacto alguno.

Sin embargo, cabe destacar que, desde sus inicios, Raspberry ha sido un proyecto enfocado a hacer llegar la tecnología y su conocimiento a la mayor cantidad de personas posible. Manteniendo este cometido social, el proyecto también puede servir como guía a personas que, aun teniendo recursos limitados, puedan reproducir todo lo que este trabajo contiene, haciendo más accesible el aprendizaje en el ámbito de la seguridad de sistemas informáticos, y teniendo un impacto positivo en lo que a derechos humanos e igualdad de oportunidades se refiere.

Finalmente, en cuanto a los costes derivados del proyecto, desde su inicio se ha intentado reducir al máximo tanto los costes en hardware como en especial, los costes en software utilizado. Un ejemplo de ello es la priorización por herramientas de código abierto.

#### 1.10. Estructura del trabajo

El proyecto se divide en cuatro grandes apartados: Introducción; Seguridad en sistemas con arquitectura RISC; Aplicaciones prácticas en entornos empresariales; y Conclusiones y futuras líneas de trabajo e investigación. En las siguientes líneas se resume el contenido de cada apartado.



El primer apartado incluye a la introducción del proyecto. En ella, se encuentra: el problema planteado y los objetivos de trabajo; la metodología utilizada; la planificación temporal y de tareas; el análisis de los riesgos que afectan al trabajo; una aproximación al estado del arte; el análisis del impacto ético, social y ambiental; y el presupuesto y recursos necesarios.

El segundo apartado corresponde al desarrollo completo de la guía de securización de Raspberry Pi. En ésta, se tratan todos aquellos aspectos clave en la seguridad: el buen gobierno de TI; las políticas de seguridad; la concienciación en seguridad; y finalmente, la defensa en profundidad, que incluye configuraciones y herramientas para cada nivel de seguridad.

Posterior a la guía de securización encontramos el apartado más práctico: las aplicaciones de todo lo recogido por la guía en entornos empresariales. En el apartado encontramos varios ejemplos de uso real de un sistema basado en Raspberry bastionado con las herramientas más adecuadas para cada caso.

Finalmente, el último apartado corresponde por un lado a las conclusiones alcanzadas tras realizar el proyecto y, por otro lado, al esbozo de futuras líneas de trabajo e investigación una vez finalizado el trabajo.

## 2. Seguridad en sistemas Raspberry

### 2.1. Mecanismos de prevención

Previo a comenzar con el bastionado como tal de nuestra Raspberry Pi (en realidad, de cualquier sistema informático), existe un paso previo que siempre se debe considerar, especialmente cuando hablamos de entornos empresariales. En concreto, hablamos del diseño y construcción de mecanismos de prevención a nivel organizativo. La implementación de este tipo de medidas podría entenderse como una capa más de la defensa en profundidad (previa a la seguridad física y perimetral, por ejemplo). Sin embargo, en esta guía se ha optado por diferenciarlo de forma clara, con el fin de distinguir las medidas organizativas o preventivas, de las medidas técnicas o de defensa.

#### 2.1.1. Gobierno de TI y ciberseguridad

Hablamos de gobierno de tecnologías de la información (y también de ciberseguridad) al diseño y construcción de un conjunto de mandatos, reglas, estructuras organizativas y otras medidas gerenciales dentro de una organización o empresa [9]. Es el sistema por el cual se gestiona de forma acertada las tecnologías dentro de una empresa para, entre otras cosas, alinear los sistemas de información y tecnologías de la información (y la seguridad de ambas) con la estrategia de negocio. El buen gobierno de TI y de su seguridad consigue una sincronía en la que: por un lado, la organización aporta los recursos que el área de TI y el de ciberseguridad necesitan para la ejecución de sus objetivos; y por otro, el área de TI configura sus objetivos con el fin último de ayudar a la organización a alcanzar su misión y visión.

Por tanto, una empresa que cuente con un buen gobierno de TI y de ciberseguridad será capaz de alcanzar tanto sus objetivos como organización, como sus objetivos de tecnología y seguridad.

En lo que a construcción del Gobierno de TI se refiere, podemos destacar varios marcos de referencia basados en modelos de gobierno concretos. Éstos no deben seguirse de principio a fin, ya que son asimilables a guías de buenas prácticas. Sin embargo, es muy recomendable que los actores responsables del Gobierno (véase *Chief Information Officer* o *Data Protection Officer*) analicen y

escojan con sabiduría aquel marco referencial que más se ajuste a la realidad de la organización.

En cuanto a los marcos de referencia de gobierno de TI, los más conocidos son los siguientes [10]:

**COBIT:** desarrollado por ISACA, es posiblemente el marco de gobierno y gestión de TI en la empresa más utilizado del mundo. Su última versión a fecha de realización de este trabajo es COBIT 2019, y su objetivo es la recomendación de buenas prácticas sobre el porqué adoptar la tecnología, cómo hacerlo de forma correcta y quién dentro de la organización debe liderar la gestión de TI. COBIT es una poderosa herramienta de consultoría, pero tal y como recomienda Gartner, los mejores resultados se logran en combinación con otros marcos de referencia como ITIL o el CMMI, explicados a continuación. Otro aspecto a tener en cuenta es que COBIT no es certificable, como sí lo es –en parte– ITIL.

**ITIL:** la *Information Technology Infrastructure Library*, como su nombre indica, es una librería de buenas prácticas de gestión de las TIC. Más allá de la gestión, ITIL también ayuda en lo que a buen gobierno de TI se refiere, facilitando en gran medida la toma de decisiones en dicha materia. La última versión de ITIL hoy en día es ITIL V4, y es certificable de forma parcial en su vertiente de sistemas de gestión de seguridad de la información (práctica nº9 de la categoría de gestión general).

**CMMI:** el *Capability Maturity Model Integration*, si bien no es un modelo concreto de gestión de TI (como si lo son COBIT e ITIL), se ha adoptado ampliamente tanto en la gestión y gobierno de TI como en los sistemas de gestión de seguridad de la información. La gran ventaja del CMMI es que nos permite analizar y conocer el grado de madurez de todos aquellos procesos que derivan del gobierno de TI, con el objetivo de identificar aquellos que requieren de mayor atención y dedicación. De nuevo, puede ser combinado con otros marcos como COBIT e ITIL, ya que en ningún caso el uso de cualquiera de los marcos excluye a otros.

### 2.1.2. Políticas de seguridad

Otro de los aspectos a tener en cuenta en lo que a medidas preventivas se refiere es el diseño y la implementación de políticas de seguridad dentro de la organización. Mediante la redacción e implementación de las políticas de seguridad, entendidas como procedimientos internos de seguridad informática y de la información y normas sobre el buen uso de los sistemas de información, se puede mejorar en gran medida la capacidad de prevención de la organización ante los ciberataques más comunes.

Es importante entender que, aunque las políticas de seguridad se enfocan en nuestro caso en los sistemas de información, actualmente dichos sistemas se utilizan en casi todas (por no decir, todas) las áreas de un negocio. Por lo que las políticas deben aplicarse a todas las áreas de negocio, y no sólo al área de TI. En la misma línea, es importante recordar que las políticas de seguridad deben ser promocionadas e implementadas desde los estratos más altos de la organización. Es muy improbable que una dirección que no cree en sus políticas de seguridad y que, ni las promociona ni las cumple, pueda lograr que sus trabajadores sí lo hagan. Las políticas deben afectar a la dirección, al personal técnico y al empleado y, en este sentido, el papel de liderazgo que la dirección debe cumplir, como veremos en la concienciación de seguridad, es un factor clave para el éxito.

Si bien cada empresa es un mundo, y las políticas de seguridad deben ser diseñadas y moldeadas a las realidades de cada organización, se pueden dibujar algunas políticas que, en general, afectan a la gran mayoría de organizaciones. Un recurso muy valioso en relación con las políticas de seguridad en las empresas es el que ofrece INCIBE a través de su sitio web: <https://www.incibe.es/protege-tu-empresa/herramientas/politicas>. Se trata de un servicio gratuito que ofrece un catálogo completo sobre políticas de seguridad para empresarios, personal técnico y empleados. Son de carácter genérico, por lo que pueden ser adaptadas e implementadas en empresas muy diferentes. No obstante, el catálogo de INCIBE debe ser usado como punto de partida o guía inicial. En todo caso, debe ser la dirección de los sistemas de información la encargada de desarrollar y ampliar las políticas de seguridad a la realidad de la organización.

### 2.1.3. Concienciación de la seguridad

El objetivo de la concienciación de la seguridad es la reducción del riesgo generado por los usuarios mediante la introducción de programas de formación y ayuda a usuarios, trabajadores y directivos de la empresa en materia de seguridad.

La idea tras la concienciación de la seguridad no es la de formar a todas las personas que pertenecen a la organización como profesionales de ciberseguridad. La concienciación está enfocada en mostrar al trabajador la forma en la que los ciberdelincuentes buscan aprovecharse del usuario para alcanzar sus objetivos (a un nivel básico), y trabajar en buenas prácticas aplicadas a su trabajo diario para reducir el riesgo de ciberataque.

Debemos tener presente que el usuario o el trabajador está muy ocupado en sus labores diarias, por lo que añadir formación en seguridad informática hará que tenga menos tiempo para alcanzar sus objetivos. Por ello, la efectividad de la concienciación se da cuando ésta se suministra en pequeñas dosis y a lo largo del tiempo [9], sin que interfiera en el día a día del trabajador.

Al igual que en el caso de las políticas de seguridad, la concienciación de la seguridad no sólo debe recaer sobre los trabajadores. Ésta deberá iniciarse en los cargos directivos de la organización, que serán los encargados de transmitir su importancia y necesidad a través del liderazgo. Sin lo anterior, el éxito de los programas de concienciación de la seguridad informática puede verse afectado.

Así, algunos de los ejercicios de formación más comunes y que pueden conformar el programa de concienciación son:

- Los talleres de formación en los que teoría y práctica entran en juego;
- La concienciación a través de carteles, anuncios o trípticos informativos;
- Simulacros esporádicos, en los que los trabajadores se enfrentan a intentos de phishing simulados que, en caso de caer en ellos, se les informa de que se trata de un simulacro de formación.

Y algunos de los conocimientos que más importancia tienen de cara a la gestión del riesgo inherente al usuario son:

- El conocimiento de los vectores de ataque más comunes, como el phishing vía correo electrónico y SMS, o la incorrecta gestión de las credenciales.
- La formación en la gestión de credenciales y contraseñas.
- Concienciación sobre la importancia de la información y los datos de carácter sensible (tanto de la empresa como de los clientes y usuarios).

Aunque parezca una medida secundaria en comparación con otras medidas técnicas, la concienciación de la seguridad es uno de los pilares fundamentales en lo que a resiliencia y seguridad informática se refiere. Recordemos que uno de los eslabones más débiles de la cadena que conforma la ciberseguridad en una empresa es el usuario, y la ausencia de medidas que empoderen al usuario frente a las amenazas hace que el resto de las medidas técnicas queden absolutamente comprometidas.

## 2.2. Defensa en profundidad

En el presente apartado se desarrollan las distintas medidas técnicas y herramientas de seguridad para la protección de las distintas capas que conforman la seguridad de un sistema basado en Raspberry Pi. Para su aplicación práctica y demostración, se ha utilizado una Raspberry Pi 4 modelo B con el sistema operativo Raspberry Pi OS Lite (64-bit). Y para ello, lo haremos desde el enfoque de la defensa en profundidad, cuyo principal objetivo es aplicar contramedidas en las distintas capas de un sistema informático, a fin de retrasar al máximo un posible ciberataque [9]. Concretamente, la defensa en profundidad divide un sistema en seis capas o niveles, para los cuales fundamenta la defensa en la prevención, la aplicación de contramedidas, y la monitorización y respuesta en cada uno de los niveles. La siguiente figura muestra de forma gráfica el modelo de defensa en profundidad [9, p. 6].

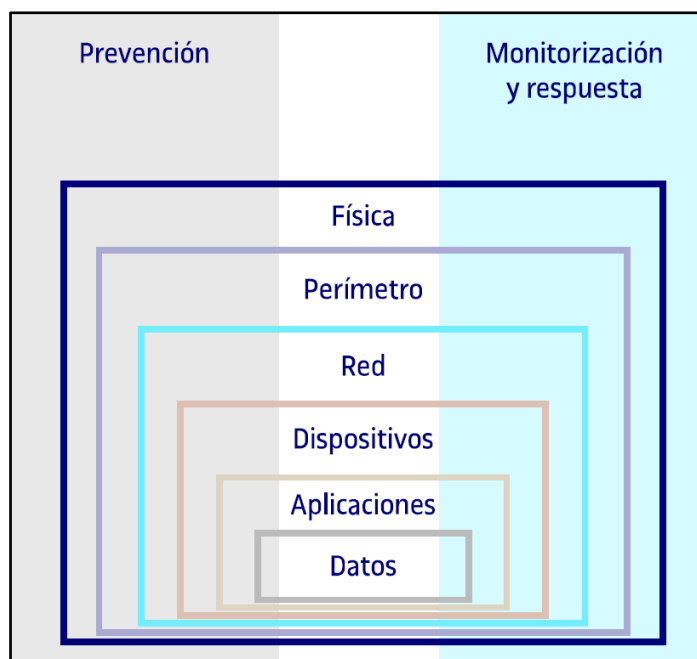


Figura 2. Esquema de la defensa en profundidad

Antes de entrar en materia, y referente a la filosofía de implementación de la defensa en profundidad, se debe tener presente en todo momento la siguiente idea: el fin de la seguridad existe para el beneficio del negocio en el que se implementa y de su misión y objetivos. Y, en ningún caso, deberá interferir más de lo necesario en su operativa diaria. Podríamos centrarnos únicamente en configurar el sistema con el bastionado más potente posible, pero de nada nos serviría si por ello, éste fuese completamente inutilizable para el negocio en el que se encuentra. Por ello, debemos trabajar para alcanzar el equilibrio ideal entre seguridad y usabilidad del sistema objeto del bastionado.

### 2.2.1. Seguridad física

La seguridad física es, junto con la seguridad perimetral, la primera línea de defensa ante ataques malintencionados. Ésta cuenta con dos vertientes: la seguridad entendida como todas aquellas medidas de seguridad enfocadas a la protección del medio físico, como obstáculos físicos, controles de acceso y recepciones, personal de seguridad privada, o videovigilancia. Y, por otro lado, la seguridad física lograda a partir de mecanismos y configuraciones que permiten proteger a los sistemas de información de ataques que conlleven la manipulación física.

En este apartado se exponen en primer lugar aquellas medidas de seguridad dedicadas a la protección del medio físico y, en segundo lugar, aquellos mecanismos enfocados a poner más difícil el trabajo de un atacante que pudiera tener acceso físico a la Raspberry Pi.

No entraremos en profundidad en lo que a protección del medio físico se refiere. Sin embargo, es importante tener en consideración aquellas medidas básicas y fundamentales, exigidas en muchos casos por frameworks o estándares de calidad, como podría ser el dominio de Seguridad Física y Ambiental de un Sistema de Gestión de la Seguridad de la Información (correspondiente al estándar ISO/IEC 27001:2022).

Con lo anterior, las medidas para la protección del medio físico más esenciales son:

- 1) El uso de obstáculos físicos que dificulten el paso, como cercas, muros, barreras o puertas;
- 2) La vigilancia de las instalaciones de la organización, a través de un circuito cerrado de televisión (o CCTV) y mediante personal de seguridad privada (vigilantes de seguridad);
- 3) La implementación de mecanismos de autenticación física que combinen tecnologías distintas y que se fundamente en varios factores de autenticación a la vez (el conocimiento de un PIN, la posesión de una credencial y la biometría);
- 4) Implantación de políticas de control de accesos y gestión de permisos a personal autorizado, con apoyo de controles de acceso físico y recepción;
- 5) El diseño e implantación de políticas de seguridad en oficinas, despachos y cualquier área sensible en materia de ciberseguridad;
- 6) Y, por último, los mecanismos de prevención y protección contra desastres naturales y amenazas (intencionales o accidentales) provocadas por las personas.

Contando con los puntos anteriores, estaremos –hasta cierto punto– protegiendo las instalaciones y la organización en la que se encuentran los sistemas de información contra intrusiones físicas indeseadas.



Habiendo tratado la seguridad del medio físico, toca enfocarnos en aquellos aspectos que podemos configurar de nuestra Raspberry Pi 4 para mejorar la seguridad ante situaciones en las que el atacante tenga acceso físico al sistema.

#### 2.2.1.1. Configuración básica inicial

Una de las primeras cosas que se debe hacer una vez instalado Raspberry Pi OS en la microSD es la configuración del usuario y contraseña. En anteriores versiones del OS, el usuario por defecto era “pi” y su contraseña “raspberrypi”. Sin embargo, esto ha cambiado en las nuevas versiones, en las que desde un inicio nos obliga a crear un usuario y contraseña. Para el usuario se recomienda escapar de nombres que puedan identificar directamente al dispositivo, como “pi”, “raspberrypi” y demás variantes. En cuanto a la contraseña, se recomienda usar una contraseña larga y que combine letras mayúsculas y minúsculas, números y símbolos.

Continuando con la configuración inicial, es fundamental mantener el sistema actualizado a las últimas versiones. Lo haremos mediante:

```
$ sudo apt update
$ sudo apt full-upgrade
```

Por último, en Raspberry Pi OS podemos ejecutar comandos como superusuario mediante *sudo* sin que nos pida una contraseña. Ello puede conllevar que un tercero con acceso al dispositivo pueda hacer y deshacer a su antojo. Para evitar esto, debemos configurar una contraseña para *sudo* con el siguiente comando:

```
$ sudo visudo /etc/sudoers.d/010_pi-nopasswd
```

Y modificar la línea correspondiente al usuario para que quede como se muestra:

```
sergi ALL=(ALL) PASSWD: ALL
```

#### 2.2.1.2. Configuración de arranque

La Raspberry Pi es un dispositivo que no cuenta con sistemas BIOS o UEFI, y tampoco es compatible con gestores de arranque como GRUB o GRUB2. Por lo que su configuración de seguridad básica no es algo que nos deba preocupar. No obstante, sí que existe un fichero que reemplaza la BIOS en la Raspberry Pi, conocido como *config.txt*, fichero de configuración que es leído por la GPU antes de que lo haga la CPU ARM, y que se ubica en */boot/config.txt* [11].

A parte del fichero *config.txt*, debemos configurar de forma segura los medios de arranque del dispositivo. En el caso de que únicamente utilicemos el puerto de microSD como medio de arranque, debemos deshabilitar la posibilidad de hacerlo desde otros medios como USB o Ethernet. Respecto a este último, no es recomendable deshabilitarlo, dado que en determinados supuestos es de gran utilidad el arranque del dispositivo a través del puerto Ethernet (como en el caso de servidores). Volviendo al arranque a través de los puertos USB, ésta viene por defecto activada en la Raspberry Pi 4 [12]. Dada la imposibilidad de encontrar la manera de desactivar la opción en *config.txt*, la forma más rápida y efectiva de solucionarlo es mediante el comando *raspi-config*, con el que podremos cambiar el orden de arranque a una opción que no contemple el arranque por USB. Para ello, basta con entrar en la configuración de la Raspberry:

```
$ sudo raspi-config
```

Una vez dentro del asistente, seleccionaremos la opción 6 (opciones avanzadas):

```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options      Configure system settings
2 Display Options    Configure display settings
3 Interface Options  Configure connections to peripherals
4 Performance Options Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options   Configure advanced settings
8 Update             Update this tool to the latest version
9 About raspi-config Information about this configuration tool

<Select>                                <Finish>
```

Figura 3. Selección de opciones avanzadas (*raspi-config*)

Ahora, seleccionaremos la opción “*Boot Order*”, identificada por el código A6:

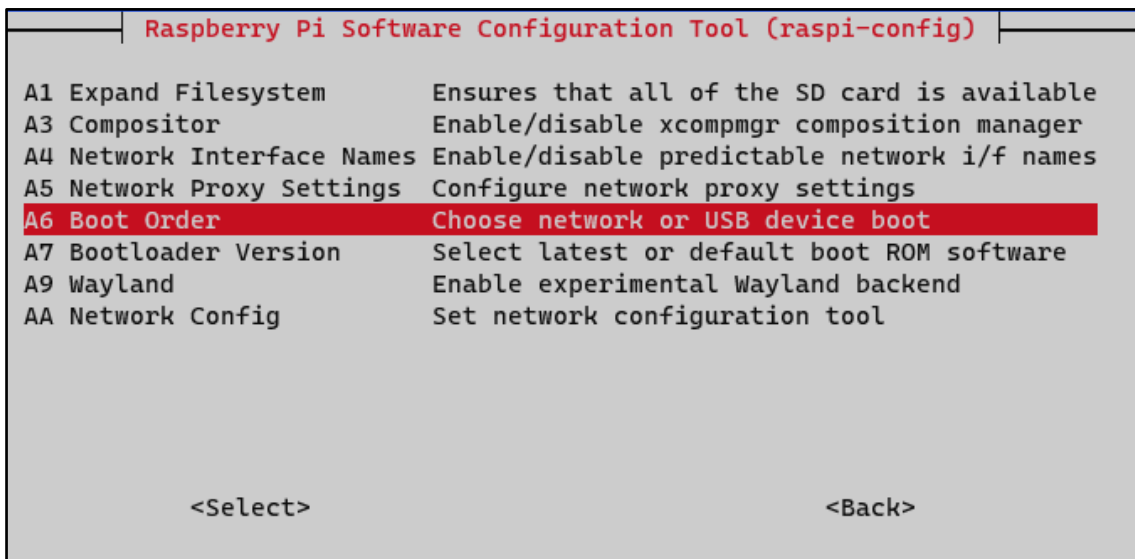


Figura 4. Selección de orden de arranque (raspi-config)

Y finalmente, seleccionamos la opción B1, es decir, “*Network Boot*”:

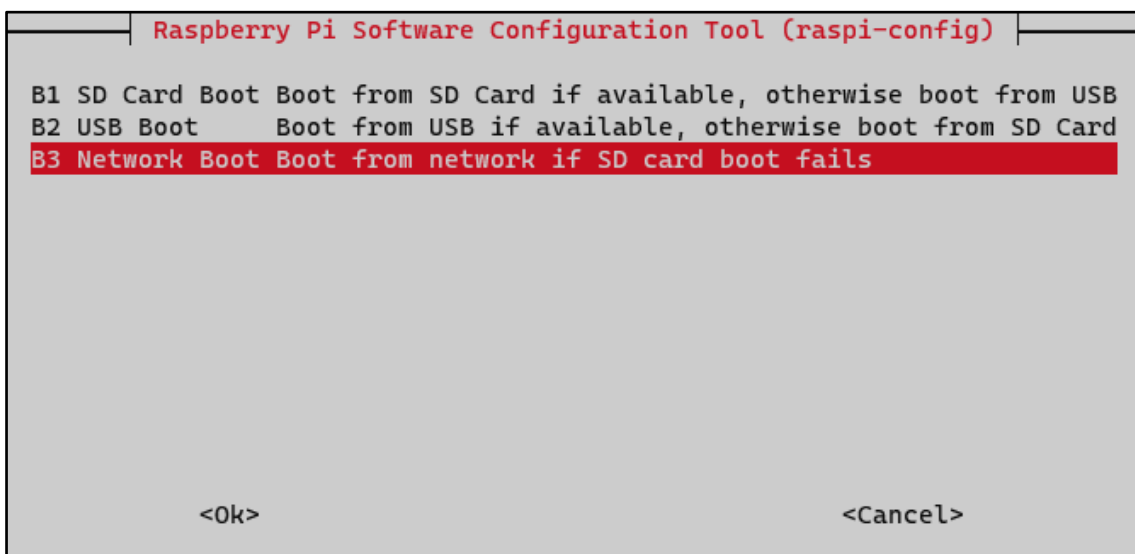


Figura 5. Selección del orden de arranque de microSD o red (raspi-config)

Esto hará que la Raspberry Pi priorice el arranque por la microSD en primer lugar, y si éste falla, lo intente a través del arranque de red (mediante el puerto Ethernet). De esta forma, aunque tengamos conectado un USB con un OS instalado, la Raspberry Pi nunca lo utilizará como medio de arranque.

Por otro lado, deberemos mantener en todo momento la opción “Auto Login” desconectada. Por el contrario, la opción de arranque preferente es aquella que, a través de consola, requiera usuario y contraseña.

### 2.2.1.3. Bloqueo de puertos y conexiones externas

La Raspberry Pi 4 cuenta con varios puertos y antenas con la capacidad de transmitir paquetes de datos: cuatro puertos USB tipo A (dos puertos 2.0 y dos puertos 3.0), un puerto Gigabit Ethernet, un módulo Wi-Fi 802.11ac, y un módulo Bluetooth 5.0. Esto hace que las posibles vías de entrada con las que cuenta un atacante con acceso físico al dispositivo sean elevadas. Por ello, es necesario conocer qué puertos son verdaderamente indispensables para el cometido de la Raspberry Pi, para posteriormente proceder a deshabilitar el resto de los puertos. Respecto a esto, lo más común es que únicamente se utilice la conexión vía Ethernet, por lo que a continuación se exponen las distintas formas de deshabilitar de forma efectiva los módulos Wi-Fi y Bluetooth, así como los cuatro puertos USB.

Para el bloqueo de los puertos USB, existen varias posibilidades. Una de ellas consiste en el corte del suministro eléctrico hacia los módulos. Sin embargo, queda totalmente descartada, ya que, en el caso de la Raspberry Pi 4, tanto los USB Tipo A como el módulo Gigabit Ethernet son alimentados por el mismo circuito, lo que hace imposible cortar el suministro de los puertos USB sin afectar al puerto Gigabit Ethernet. Por suerte, existen otras formas de hacerlo a nivel de software.

La forma expuesta a continuación utiliza una herramienta no incluida en Raspberry Pi OS llamada “uhubctl” (que podemos encontrar en el repositorio <https://github.com/mvp/uhubctl>). También necesitaremos descargar la utilidad “git”, no incluida en Raspberry Pi OS y que nos permite clonar los archivos de un repositorio concreto para luego proceder a su instalación en el sistema.

Primero de todo, instalamos “git” y “libusb”, librería que nos permitirá acceder de forma lógica a los puertos USB:

```
$ sudo apt install git
$ sudo apt install libusb-1.0-0-dev
```

Instalamos la herramienta “uhubctl”:

```
$ sudo git clone https://github.com/mvp/uhubctl
$ cd uhubctl
```

```
$ sudo make
$ sudo make install
```

Una vez tenemos todo lo necesario instalado, podemos deshabilitar los puertos USB a mediante “uhubctl”, pero sin comprometer el puerto Ethernet. Como se ha expuesto anteriormente, la Raspberry Pi 4 Modelo B cuenta con un total de cuatro puertos USB. De cara a la inhabilitación de cada uno de los puertos, debemos conocer el identificador numérico que el sistema asigna a cada puerto. En nuestro caso, los puertos USB 1 y 2 corresponden a los USB 3.0, y los puertos 3 y 4 corresponden a los USB 2.0 y 2.1. Raspberry Pi OS divide ambas parejas en dos hubs distintos: el hub 2, correspondiente a los dos puertos USB 3.0; y el hub 1-1, correspondiente a los dos puertos USB 2.0 y 2.1.

Sabiendo el identificador de cada puerto USB, el bloqueo de los distintos puertos USB puede realizarse mediante el siguiente comando (nótese que el bloqueo debe hacerse de forma individual para cada puerto, aunque en nuestro caso el ejemplo se presenta en un único *one liner*):

```
$ sudo uhubctl -l 1-1 -p 1 -a off && uhubctl -l 1-1 -p 2 -a off &&
uhubctl -l 1-1 -p 3 -a off && uhubctl -l 1-1 -p 4 -a off

Current status for hub 2 [1d6b:0003 Linux 6.1.19-v8+ xhci-hcd xHCI
Host Controller 0000:01:00.0, USB 3.00, 4 ports, ppps]

  Port 1: 0080 off

Sent power off request

New status for hub 2 [1d6b:0003 Linux 6.1.19-v8+ xhci-hcd xHCI Host
Controller 0000:01:00.0, USB 3.00, 4 ports, ppps]

  Port 1: 0080 off

Current status for hub 1-1 [2109:3431 USB2.0 Hub, USB 2.10, 4 ports,
ppps]

  Port 1: 0000 off

Sent power off request

New status for hub 1-1 [2109:3431 USB2.0 Hub, USB 2.10, 4 ports, ppps]

  Port 1: 0000 off

There were permission problems while accessing USB.
```

```
Follow https://git.io/JIB2Z for a fix!
```

```
No compatible devices detected at location 1-1!
```

```
Run with -h to get usage info.
```

Como puede verse en el ejemplo anterior, el comando ya nos brinda información sobre los puertos que han sido bloqueados. Sin embargo, podemos obtener una información más legible mediante el comando “sudo uhubctl”. Así, comprobamos que los puertos USB han sido efectivamente deshabilitados:

```
$ sudo uhubctl
Current status for hub 2 [1d6b:0003 Linux 6.1.19-v8+ xhci-hcd xHCI
Host Controller 0000:01:00.0, USB 3.00, 4 ports, ppps]
  Port 1: 0080 off
  Port 2: 0080 off
  Port 3: 0080 off
  Port 4: 0080 off
Current status for hub 1-1 [2109:3431 USB2.0 Hub, USB 2.10, 4 ports,
ppps]
  Port 1: 0000 off
  Port 2: 0000 off
  Port 3: 0000 off
  Port 4: 0000 off
Current status for hub 1 [1d6b:0002 Linux 6.1.19-v8+ xhci-hcd xHCI
Host Controller 0000:01:00.0, USB 2.00, 1 ports, ppps]
  Port 1: 0507 power highspeed suspend enable connect [2109:3431
USB2.0 Hub, USB 2.10, 4 ports, ppps]
```

Sin embargo, la configuración de uhubctl no es persistente, por lo que durará hasta el siguiente reinicio. Si deseamos que la configuración sea duradera en el tiempo, necesitamos que lo anterior se ejecute en cada reinicio. Para ello, basta con automatizarlo mediante el fichero “*rc.local*”. Éste es un binario (escrito en Bash) que se lee al iniciar el sistema, y su misión es realizar las tareas que contiene. Para automatizar el bloqueo de USB, basta con crear un script que contenga el comando, y llamar a su ejecución desde *rc.local*.

Para crear un script escrito en Bash, basta con movernos al directorio */bin/*, y allí creamos el archivo. En nuestro caso, se llama “*startconfig*”:

```
$ cd /bin/  
$ sudo nano startconfig
```

En él, añadimos las siguientes líneas, incluyendo el comando que queremos que se ejecute. Por ejemplo:

```
#!/bin/bash -e  
  
sudo uhubctl -l 1-1 -p 1 -a off && uhubctl -l 1-1 -p 2 -a off &&  
uhubctl -l 1-1 -p 3 -a off && uhubctl -l 1-1 -p 4 -a off  
  
echo "Todos los puertos USB han sido deshabilitados"  
  
exit
```

Una vez creado, le damos permisos de ejecución:

```
$ sudo chmod 744 startconfig
```

Ahora que tenemos creado el binario, debemos modificar el fichero *rc.local* para que ejecute el binario “*startconfig*”:

```
$ sudo nano /etc/rc.local
```

E incluir las siguientes líneas al final del documento, con la instrucción de ejecución del binario:

```
# Disable all USB  
/bin/startconfig  
  
exit 0
```

Respecto a *rc.local*, es importante que los distintos comandos o binarios que incluyamos se encuentren entre la primera línea (“*#!/bin/bash*”) y la última línea (“*exit 0*”).

Ahora sí, el binario que incluye la instrucción de bloqueo de los puertos USB se ejecutará en cada reinicio a través de *rc.local*.

Pasando ahora a los módulos Wi-Fi y Bluetooth, éstos pueden ser deshabilitados de varias formas distintas [13]: de forma temporal o en cada reinicio del sistema

mediante `crontab`; añadiendo los módulos a la lista negra de `modprobe`; o editando el archivo `config.txt`.

No obstante, se expone a continuación una de las formas más sencillas para desactivar el Wi-Fi y el Bluetooth de forma persistente. Para ello, haremos uso de “*RFKill*”, un subsistema que incluye el kernel de Linux mediante el cual podemos activar o desactivar los radiotransmisores de la Raspberry Pi. Los pasos a seguir se exponen a continuación.

Mediante el siguiente comando comprobamos el estado actual del Wi-Fi y Bluetooth:

```
$ sudo rfkill
ID TYPE      DEVICE      SOFT      HARD
 0 wlan      phy0       unblocked unblocked
 1 bluetooth hci0       unblocked unblocked
```

Podemos comprobar que, tanto a nivel de software como a nivel de hardware, ambas interfaces se encuentran bloqueadas. El comando para deshabilitar el módulo Wi-Fi y módulo Bluetooth es el siguiente:

```
$ sudo rfkill block wifi && sudo rfkill block bluetooth
```

Una vez ejecutado el comando, podemos volver a comprobar el estado del Wi-Fi y Bluetooth tras el bloqueo. En el siguiente ejemplo podemos ver que, efectivamente, ambas interfaces de red han sido bloqueadas a nivel de software:

```
$ sudo rfkill
ID TYPE      DEVICE      SOFT      HARD
 0 wlan      phy0       blocked  unblocked
 1 bluetooth hci0       blocked  unblocked
```

Con lo anterior, se han reducido en gran medida las posibles vías de entrada con las que un atacante con acceso físico podría introducirse en el sistema. En caso de necesitar alguno de los módulos o puertos bloqueados, bastaría con eliminarlos de los comandos de bloqueo o bien volverlos a habilitar. Para el caso de los USB sería:



```
$ sudo uhubctl -l 1-1 -p 1 -a on
$ sudo uhubctl -l 1-1 -p 2 -a on
$ sudo uhubctl -l 1-1 -p 3 -a on
$ sudo uhubctl -l 1-1 -p 4 -a on
```

Mientras que para las interfaces de red Wi-Fi o Bluetooth sería:

```
$ sudo rfkill unblock wifi
$ sudo rfkill unblock bluetooth
```

#### 2.2.1.4. Bloqueo y cierre de sesiones

Otra de las medidas de seguridad física que podemos configurar en la Raspberry Pi es el cierre automático de sesiones, con el objetivo de reducir el riesgo por dispositivos desatendidos. Para ello, se ha usado la variable TMOU de Bash.

Para poder hacerlo de forma persistente en el sistema, debemos modificar el archivo *“bashrc”*, ubicada en el directorio de inicio de cada usuario. Utilizamos *“nano”* para editar el archivo en cuestión:

```
$ sudo nano ~/.bashrc
```

Y una vez dentro, añadimos al final del archivo la siguiente línea:

```
# auto log-out with TMOU
export TMOU=600
```

Figura 6. Configuración de *bashrc* incluyendo TMOU

Con lo anterior, el sistema cerrará la sesión de forma automática si el usuario no realiza ninguna acción en 600 segundos (es decir, 10 minutos). Esto es sólo un ejemplo. Cada administrador de sistemas deberá asignar un tiempo máximo de inactividad atendiendo a las necesidades de los usuarios.

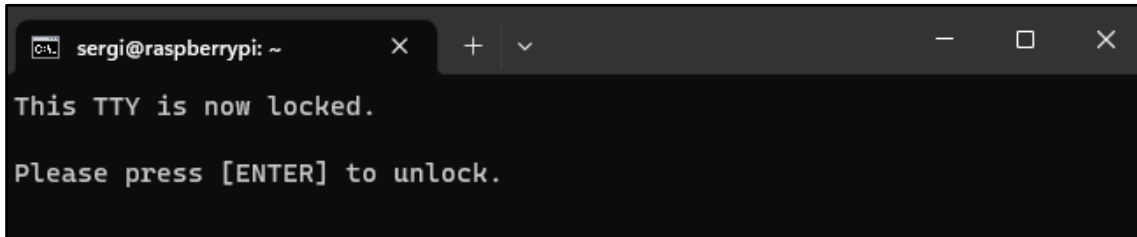
Por último, podemos realizar bloqueos del Terminal de Raspberry Pi OS. Para ello, se pueden usar herramientas como *“vlock”*, disponible en la mayoría de los repositorios. Podemos instalarla mediante:

```
$ sudo apt install vlock
```

Su uso es muy sencillo, pero depende del usuario realizar las distintas opciones requeridas para bloquear el Terminal. Por ejemplo, para bloquear la Terminal del propio usuario basta con utilizar:

```
$ vlock --current
```

Y el terminal se bloqueará, exigiendo de nuevo la contraseña del usuario para poder utilizarlo:



*Figura 7. Terminal bloqueada por vlock*

Si por el contrario deseamos bloquear todas las sesiones, podemos utilizar:

```
$ vlock -all
```

#### 2.2.1.5. Protección y cifrado de la microSD

Uno de los mayores riesgos para la seguridad física de la Raspberry Pi es la facilidad con la que un atacante puede extraer la microSD, ya que ésta se encuentra totalmente accesible de forma física. Recordemos que las configuraciones de seguridad que podemos hacer sobre nuestro sistema se perderán al momento en que se extraiga la microSD. Siendo conscientes de este problema, deberemos mantener la Raspberry Pi en una ubicación segura, cuyo acceso físico sea difícil. Para ello, existen en el mercado multitud de opciones de cajas que encapsulan la Raspberry Pi y no permiten extraer la microSD sin antes desmontar la propia caja.

Sumado a lo anterior, otra de las medidas más eficaces es el cifrado de la microSD. Si bien esto puede hacerse de forma relativamente sencilla en un sistema convencional (como un ordenador portátil), la Raspberry Pi 4 no cuenta con la mayoría de las características necesarias para llevar a cabo el proceso de cifrado. Por ejemplo, no cuenta con aceleración de cifrado por hardware (conocida como AES). Por suerte, existen herramientas complementarias que nos permitirán realizar el proceso.

En nuestro caso concreto, utilizaremos la especificación de cifrado “LUKS” (*Linux Unified Key Setup*), que cuenta con un conjunto de herramientas y es la especificación más usada para el cifrado de discos en entornos Linux.

Los pasos para el cifrado de la microSD con LUKS se relatan a continuación, siguiendo la guía realizada por “*rr-developer*” [14].

El primer paso es instalar la herramienta complementaria “*cryptsetup*”:

```
$ sudo apt install busybox cryptsetup initramfs-tools
```

A continuación, verificamos que contamos con “*Adiantum*”, la construcción de cifrado que nos ofrece AES:

```
$ cryptsetup benchmark -c xchacha20,aes-adiantum-plain64

# Tests are approximate using memory only (no storage IO).
#           Algorithm |           Key |           Encryption |           Decryption
xchacha20,aes-adiantum      256b      126.4 MiB/s      138.5 MiB/s
```

Ahora, debemos crear el binario “*initramfs*” mediante:

```
$ sudo nano /etc/kernel/postinst.d/initramfs-rebuild
```

Y en el binario creado (en nuestro caso “*initramfs-rebuild*”) incluimos el siguiente contenido:

```
#!/bin/sh -e

# Rebuild initramfs.gz after kernel upgrade to include new kernel's
modules.

#
https://github.com/Robpol86/robpol86.com/blob/master/docs/\_static/initramfs-rebuild.sh

# Save as (chmod +x): /etc/kernel/postinst.d/initramfs-rebuild
# Remove splash from cmdline.
if grep -q '\bsplash\b' /boot/cmdline.txt; then
    sed -i 's/ \?splash \?/ /' /boot/cmdline.txt
fi
```

```

# Exit if not building kernel for this Raspberry Pi's hardware
version.
version="$1"
current_version="$(uname -r)"
case "${current_version}" in
    *-v7+)
        case "${version}" in
            *-v7+) ;;
            *) exit 0
        esac
    ;;
    *)
        case "${version}" in
            *-v7+) exit 0 ;;
        esac
    ;;
esac

# Exit if rebuild cannot be performed or not needed.
[ -x /usr/sbin/mkinitramfs ] || exit 0
[ -f /boot/initramfs.gz ] || exit 0
lsinitramfs /boot/initramfs.gz |grep -q "/$version$" && exit 0 #
Already in initramfs.

# Rebuild.
mkinitramfs -o /boot/initramfs.gz "$version"

```

Como cualquier otro binario, debemos hacerlo ejecutable:

```
$ sudo chmod +x /etc/kernel/postinst.d/initramfs-rebuild
```

El siguiente paso es especificar los programas que deberán ser incluidos en initramfs. Éstos son: *“resize2fs”*, *“fdisk”* y *“cryptsetup”*. Para ello, debemos crear un nuevo archivo en la siguiente ubicación:

```
$ sudo nano /etc/initramfs-tools/hooks/luks_hooks
```

Y que tenga el siguiente contenido:

```
#!/bin/sh -e
PREREQS=""
case $1 in
    prereqs) echo "${PREREQS}"; exit 0;;
esac

. /usr/share/initramfs-tools/hook-functions

copy_exec /sbin/resize2fs /sbin
copy_exec /sbin/fdisk /sbin
copy_exec /sbin/cryptsetup /sbin
```

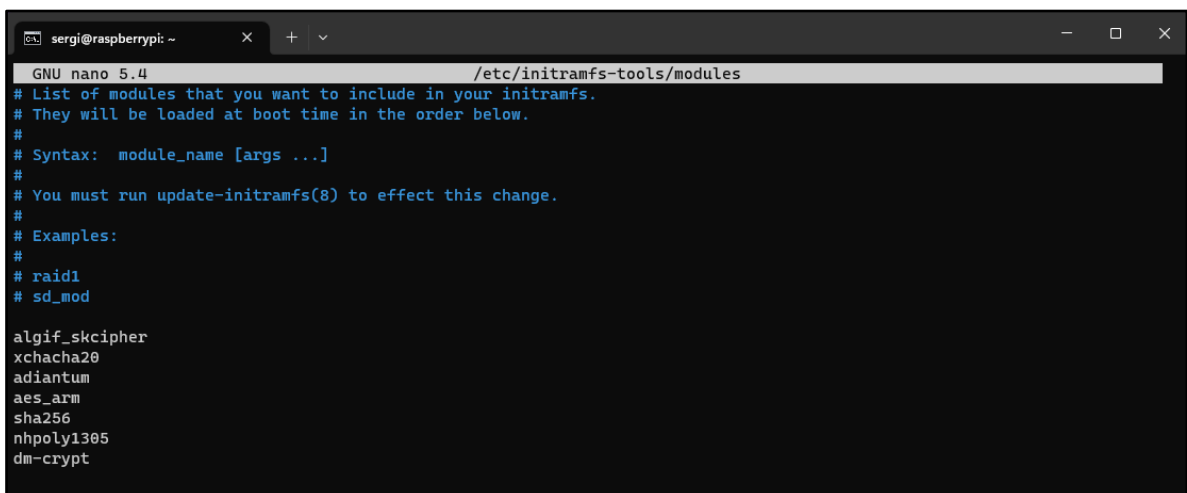
De nuevo, lo hacemos ejecutable:

```
$ sudo chmod +x /etc/initramfs-tools/hooks/luks_hooks
```

De igual forma, necesitamos configurar y añadir módulos del kernel de Linux a initramfs, y que son necesarios para el cifrado con LUKS, pero que no se incluyen por defecto. Lo anterior se hace modificando el archivo *“/etc/initramfs-tools/modules”*:

```
$ sudo nano /etc/initramfs-tools/modules
```

En el que deberemos añadir los módulos del kernel, como se muestra en la siguiente imagen:



```
sergi@raspberrypi: ~
GNU nano 5.4 /etc/initramfs-tools/modules
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax: module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod

algif_skcipher
xchacha20
adiantum
aes_arm
sha256
nhpoly1305
dm-crypt
```

Figura 8. Módulos de initramfs

Para cargar todos los cambios realizados anteriormente, debemos montar de nuevo “initramfs” con el comando:

```
$ sudo -E CRYPTSETUP=y mkinitramfs -o /boot/initramfs.gz
cryptsetup: ERROR: Couldn't resolve device /dev/root
cryptsetup: WARNING: Couldn't determine root device
```

Nos devolverá un error. Sin embargo, podemos comprobar cómo, efectivamente, los programas que se requieren han sido cargados correctamente:

```
sergi@raspberrypi:~$ lsinitramfs /boot/initramfs.gz | grep -P "sbin/(cryptsetup|resize2fs|fdisk)"
usr/sbin/cryptsetup
usr/sbin/fdisk
usr/sbin/resize2fs
sergi@raspberrypi:~$ lsinitramfs /boot/initramfs.gz | grep -P "(algif_skcipher|chacha|adiantum|aes-arm|sha256|nnpoly1305|dm-crypt)"
usr/lib/modules/6.1.19-v8+/kernel/arch/arm64/crypto/aes-arm64.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/arch/arm64/crypto/chacha-neon.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/crypto/adiantum.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/crypto/algif_skcipher.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/crypto/chacha20poly1305.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/crypto/chacha_generic.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/crypto/nnpoly1305.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/drivers/md/dm-crypt.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/lib/crypto/libchacha.ko.xz
usr/lib/modules/6.1.19-v8+/kernel/lib/crypto/libchacha20poly1305.ko.xz
usr/bin/sha256sum
```

Figura 9. Initramfs montado correctamente

Es momento de preparar el inicio del sistema para que utilice un sistema de ficheros de *root* encriptado. El primer archivo a modificar es *config.txt*, incluyendo lo siguiente:

```
$ sudo nano /boot/config.txt
```

```
[all]
# Add initramfs
initramfs initramfs.gz followkernel
```

Figura 10. Configuración de *config.txt* con *ititramfs*

El siguiente archivo a modificar es “*/boot/cmdline.txt*”, con tal de cambiar el valor de *root* de la partición por defecto “*/dev/mmcblk0p2*” a “*dev/mapper/sdcard*”. El contenido del archivo debería quedar tal que así, incluyendo el parámetro de “*cryptdevice=*”:

```
console=serial0,115200 console=tty1 root=/dev/mapper/sdcard
rootfstype=ext4 fsck.repair=yes rootwait
cryptdevice=/dev/mmcblk0p2:sdcard
```

A continuación, modificamos “*/etc/fstab*”, con tal de actualizar a la nueva ubicación de la partición de *root*:

```
sergi@raspberrypi: ~
GNU nano 5.4 /etc/fstab
proc /proc proc defaults 0 0
PARTUUID=6bcf2c18-01 /boot vfat defaults 0 2
/dev/mapper/sdcard / ext4 defaults,noatime 0 1
# a swapfile is not a swap partition, no line here
# use dphys-swapfile swap[on|off] for that
```

Figura 11. Configuración de *fstab* con la nueva ubicación

Finalmente, el último archivo a modificar es *“/etc/crypttab”*, añadiendo al final del archivo una nueva línea con el siguiente contenido:

```
sergi@raspberrypi: ~
GNU nano 5.4 /etc/crypttab
# <target name> <source device> <key file> <options>
sdcard /dev/mmcblk0p2 none luks
```

Figura 12. Configuración de *crypttab*

Una vez tenemos todo configurado, podemos proceder al reinicio de la Raspberry Pi. El resultado esperado debería ser un error, dado que la Raspberry está intentando iniciar con una partición de *root* que todavía no ha sido creada. No obstante, nos aparecerá la Shell de *initramfs*, mediante la cual podremos terminar el cifrado.

Una vez dentro de la Shell de *initramfs*, y previo a cualquier modificación, debemos comprobar si podemos usar *cryptsetup* correctamente, mediante el siguiente test:

```
(initramfs) cryptsetup benchmark -c xchacha20,aes-adiantum-plain64
```

```
(initramfs) cryptsetup benchmark -c xchacha20,aes-adiantum-plain64
# Tests are approximate using memory only (no storage IO).
PBKDF2-sha1      143248 iterations per second for 256-bit key
PBKDF2-sha256   233639 iterations per second for 256-bit key
PBKDF2-sha512   189684 iterations per second for 256-bit key
PBKDF2-ripemd160 121362 iterations per second for 256-bit key
PBKDF2-whirlpool 49423 iterations per second for 256-bit key
argon2id        4 iterations, 298734 memory, 4 parallel threads (CPUs) for 256-bit key (requested 2000 ms time)
argon2id        4 iterations, 210683 memory, 4 parallel threads (CPUs) for 256-bit key (requested 2000 ms time)
#
```

Algorithm	Key	Encryption	Decryption
aes-cbc	128b	32.5 MiB/s	35.8 MiB/s
serpent-cbc	128b	N/A	N/A
twofish-cbc	128b	24.1 MiB/s	25.6 MiB/s
aes-cbc	256b	27.2 MiB/s	27.5 MiB/s
serpent-cbc	256b	N/A	N/A
twofish-cbc	256b	25.4 MiB/s	25.6 MiB/s
aes-xts	256b	34.3 MiB/s	37.8 MiB/s
serpent-xts	256b	N/A	N/A
twofish-xts	256b	24.7 MiB/s	26.3 MiB/s
aes-xts	512b	28.7 MiB/s	29.1 MiB/s
serpent-xts	512b	N/A	N/A
twofish-xts	512b	26.6 MiB/s	26.3 MiB/s

```
(initramfs)
```

Figura 13. Ejemplo de respuesta del benchmark de cryptsetup

Una vez comprobado lo anterior, podemos iniciar el proceso de encriptado de la partición de *root*. Para ello, verificamos que no existen errores en la partición:

```
(initramfs) e2fsck -f /dev/mmcblk0p2
```

Y reducimos su tamaño lo máximo posible mediante “*resize2fs*”:

```
(initramfs) resize2fs -fM -p /dev/mmcblk0p2
```

```
(initramfs) resize2fs -fM -p /dev/mmcblk0p2
resize2fs 1.46.2 (28-Feb-2021)
Resizing the filesystem on /dev/mmcblk0p2 to 701031 (4k) blocks.
Begin pass 2 (max = 2534)
Relocating blocks          XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Begin pass 3 (max = 476)
Scanning inode table      XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
The filesystem on /dev/mmcblk0p2 is now 701031 (4k) blocks long.
(initramfs)
```

Figura 14. Reducción del tamaño de la partición *root*

Es importante en este paso tomar nota del número de bloques de 4k (en el caso del ejemplo, “701031”), ya que deberemos introducirlo en pasos siguientes.

El siguiente paso es calcular la suma de control antes de copiar la partición, cuyo resultado nos servirá para verificar que la información copiada es la misma. Nótese que en “*count=*” debemos introducir el número de bloques de 4k:

```
(initramfs) time dd bs=4k count=701031 if=/dev/mmcblk0p2 | sha1sum
```



```
(initramfs) time dd bs=4k count=701031 if=/dev/mmcblk0p2 | sha1sum
BusyBox v1.30.1 (Debian 1:1.30.1-6+b3) multi-call binary.

Usage: dd [if=FILE] [of=FILE] [ibs=N obs=N/bs=N] [count=N] [skip=N] [seek=N]
[conv=notrunc|noerror|sync|fsync]
[iflag=skip_bytes|fullblock] [oflag=seek_bytes]

Copy a file with converting and formatting

if=FILE      Read from FILE instead of stdin
of=FILE      Write to FILE instead of stdout
bs=N         Read and write N bytes at a time
ibs=N        Read N bytes at a time
obs=N        Write N bytes at a time
count=N      Copy only N input blocks
skip=N       Skip N input blocks
seek=N       Skip N output blocks
conv=notrunc Don't truncate output file
conv=noerror Continue after read errors
conv=sync    Pad blocks with zeros
conv=fsync   Physically write data out before finishing
conv=swab    Swap every pair of bytes
iflag=skip_bytes skip=N is in bytes
iflag=fullblock Read full blocks
oflag=seek_bytes seek=N is in bytes
status=noxfer Suppress rate output
status=none  Suppress all output

N may be suffixed by c (1), u (2), b (512), kB (1000), k (1024), MB, M, GB, G
da39a3ee5e6b4b0d3255bfef95601890afd80709 -
Command exited with non-zero status 1
real 0m 0.06s
user 0m 0.01s
sys 0m 0.04s
(initramfs) [ 420.084937] usb 2-1: USB disconnect, device number 2
[ 423.989266] usb 2-1: new SuperSpeed USB device number 3 using xhci_hcd
[ 424.010884] usb 2-1: New USB device found, idVendor=0781, idProduct=5581, bcdDevice= 1.
[ 424.012745] usb 2-1: New USB device strings: Mfr=1, Product=1, S
```

Figura 15. Cálculo de la suma de control

Si conectamos una memoria USB a la Raspberry Pi, podremos comprobar como aparecen un seguido de mensajes informativos sobre la unidad conectada, y desaparece la Shell de initramfs. Para verificar la ruta del USB, podemos utilizar:

```
fdisk -l /dev/sda
```

Ahora, copiamos todo el sistema de ficheros de *root* que se encuentra en la microSD a la memoria USB (indicando en “*count=*” el número de bloques de 4k):

```
(initramfs) time dd bs=4k count=701031 if=/dev/mmcblk0p2 of=/dev/sda
```

Una vez copiado, podemos comprobar si la copia se ha realizado correctamente con el cálculo de la suma de control:

```
(initramfs) time dd bs=4k count=701031 if=/dev/sda | sha1sum
```

Ahora que ya hemos comprobado que las sumas de control dan el mismo resultado y que, por tanto, la copia se ha realizado correctamente, podemos crear el volumen de LUKS mediante *cryptsetup*. Los parámetros del comando mostrado a continuación pueden ser modificados según convenga (más

información sobre las opciones de cryptsetup puede encontrarse en <https://man7.org/linux/man-pages/man8/cryptsetup.8.html>):

```
(initramfs) cryptsetup --type luks2 --cipher xchacha20,aes-adiantum-plain64 --hash sha256 --iter-time 5000 --pbkdf argon2i luksFormat /dev/mmcblk0p2
```

Mientras se realiza el proceso, cryptsetup nos pedirá a través de consola que indiquemos una frase-contraseña y su confirmación, que servirá posteriormente para descryptar y poder leer la partición.

Una vez creada la partición encriptada (el volumen LUKS), deberemos abrirlo indicando la frase-contraseña anteriormente configurada:

```
(initramfs) cryptsetup luksOpen /dev/mmcblk0p2 sdcard
```

Y copiar el contenido del sistema de ficheros de *root* desde la memoria USB hacia la microSD:

```
(initramfs) time dd bs=4k count=701031 if=/dev/sda of=/dev/mapper/sdcard
```

Volvemos a calcular la suma de control para validar el proceso:

```
(initramfs) time dd bs=4k count=701031 if=/dev/mapper/sdcard | sha1sum
```

Y también validamos el sistema de ficheros del volumen LUKS:

```
(initramfs) e2fsck -f /dev/mapper/sdcard
```

Por último, expandimos el sistema de ficheros copiado a su tamaño original con *resize2fs*:

```
(initramfs) resize2fs -f /dev/mapper/sdcard
```

Ya podemos extraer la memoria USB (que ya ha hecho su función), y salir de la Shell de initramfs mediante el comando “*exit*”. Esto hará que el proceso de inicio continúe, pero vuelva a la Shell de initramfs, dado que el volumen LUKS sigue sin ser accesible para el sistema. Para hacer que sea accesible, basta con abrirlo de nuevo:

```
(initramfs) cryptsetup luksOpen /dev/mmcblk0p2 sdcard
```

Y volver a salir de la Shell de initramfs mediante “*exit*”. Para evitar que cada vez que se inicie la Raspberry Pi, el sistema entre en la Shell de initramfs y tener que abrir el volumen LUKS, debemos montar por última vez initramfs (iniciando sesión con cualquier usuario) y reiniciar el sistema:

```

$ sudo mkinitramfs -o /tmp/initramfs.gz
$ sudo cp /tmp/initramfs.gz /boot/initramfs.gz
$ sudo reboot

```

Una vez reiniciado el sistema, podremos comprobar como durante el proceso de inicio, nos pedirá que aportemos la frase-contraseña para poder descryptar la partición:

```

[ 1.501100] usb 1-1: new USB device found, idVendor=2109, idProduct=3431, bcdDevice= 4.21
[ 1.906547] usb 1-1: New USB device strings: Mfr=0, Product=1, SerialNumber=0
[ 1.908868] usb 1-1: Product: USB2.0 Hub
[ 1.913368] hub 1-1:1.0: USB hub found
[ 1.916034] hub 1-1:1.0: 4 ports detected
[ 2.213585] usb 1-1.2: new full-speed USB device number 3 using xhci_hcd
Starting version 247.3-7+deb11u1
[ 2.322919] usb 1-1.2: New USB device found, idVendor=045e, idProduct=0745, bcdDevice= 6.76
[ 2.325783] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 2.328241] usb 1-1.2: Product: Microsoft® 2.4GHz Transceiver v8.0
[ 2.330762] usb 1-1.2: Manufacturer: Microsoft
[ 2.347398] input: Microsoft Microsoft® 2.4GHz Transceiver v8.0 as /devices/platform/scb/fd500000.p
[ 2.410665] hid-generic 0003:045E:0745.0001: input,hidraw0: USB HID v1.11 Keyboard [Microsoft Micros
[ 2.429775] input: Microsoft Microsoft® 2.4GHz Transceiver v8.0 Mouse as /devices/platform/scb/fd50
[ 2.433282] input: Microsoft Microsoft® 2.4GHz Transceiver v8.0 Consumer Control as /devices/platfo
[ 2.494149] hid-generic 0003:045E:0745.0002: input,hidraw1: USB HID v1.11 Mouse [Microsoft Microsoft
[ 2.510765] input: Microsoft Microsoft® 2.4GHz Transceiver v8.0 Consumer Control as /devices/platfo
[ 2.574113] input: Microsoft Microsoft® 2.4GHz Transceiver v8.0 System Control as /devices/platform
[ 2.578411] hid-generic 0003:045E:0745.0003: input,hiddev96,hidraw2: USB HID v1.11 Device [Microsoft
[ 2.953515] brcmstb-i2c fef04500.i2c: @97500hz registered in polling mode
[ 2.966356] brcmstb-i2c fef09500.i2c: @97500hz registered in polling mode
Begin: Loading essential drivers ... [ 3.119380] NET: Registered PF_ALG protocol family
[ 3.264492] device-mapper: ioctl: 4.47.0-ioctl (2022-07-28) initialised: dm-devel@redhat.com
done.
Begin: Running /scripts/init-premount ... done.
Begin: Mounting root file system ... Begin: Running /scripts/local-top ... Please unlock disk sdcard: _

```

Figura 16. Petición de contraseña para desbloqueo de SD

Con lo anterior, ya tendremos el contenido de la microSD protegida ante cualquiera que desee extraerla y leerla posteriormente en otro equipo. A su vez, también estaremos protegiendo el inicio del sistema, dado que durante el proceso se nos pedirá la frase-contraseña utilizada para cifrar la microSD con tal de descryptarla.

### 2.2.2. Seguridad perimetral

El siguiente escalón en la defensa en profundidad es la seguridad perimetral, entendida como la aplicación de contramedidas en las zonas del sistema (o infraestructura) que se encuentran expuestas al exterior (a Internet).

Los controles y herramientas de seguridad del perímetro de la red pueden ser aplicadas desde distintos enfoques en lo que a uso de recursos se refiere. Por ejemplo, podríamos tener para la protección de nuestra Raspberry Pi 4 (en la que se ejecuta algún proceso crítico de la empresa) un sistema –otra Raspberry Pi o similares– dedicado exclusivamente a hacer funciones de cortafuegos. De

hecho, las soluciones presentadas en este apartado pueden ser configuradas en distintos sistemas individualizados, siendo la opción más recomendable. La realidad presupuestaria de este trabajo final de máster hace que todas las medidas mostradas en este apartado hayan sido puestas a prueba en una única Raspberry Pi 4. No obstante, esto no supone un problema, ya que todas ellas son compatibles con cualquier dispositivo basado en el kernel de Linux.

Con lo anterior, pasamos ahora a tratar las medidas indispensables para la seguridad de nuestro sistema en lo que al perímetro se refiere.

#### *2.2.2.1. Configuración de un cortafuegos*

En sistemas operativos basados en el núcleo de Linux, la solución de firewall de código abierto más usada es Iptables. Iptables es un módulo del kernel de Linux que permite, mediante el uso de reglas, la gestión del tráfico de red entrante y saliente del sistema en el que se configura.

Si bien Iptables es la solución estrella en lo que a sistemas firewall dedicados se refiere, su configuración es laboriosa y requiere de mucho conocimiento y tiempo. Por esta razón, para esta guía se ha optado por el uso de *Uncomplicated Firewall* (UFW en adelante). UFW es un software que, a través de líneas de comandos, permite configurar Iptables de una forma intuitiva y simple. Aunque la herramienta está especialmente diseñada para su uso en dispositivos finales, como es el caso de nuestra Raspberry Pi 4, para la seguridad perimetral nos interesa que sea una segunda máquina o servidor el que pueda realizar las tareas de cortafuegos. La elección del hardware dedicado al firewall perimetral recae sobre el administrador de la red, por lo que no entraremos en más detalle en ese aspecto. De igual forma, también es elección del administrador de red la ubicación del firewall dentro de la red. Sin embargo, si lo que deseamos es lograr que éste proteja el perímetro de la red, deberá ser ubicado entre el dispositivo gateway y el resto de la red (comúnmente conectada al firewall a través de un switch), filtrando de esta forma el tráfico y delimitando el perímetro de la red.

Respecto a la configuración de UFW, no existe una política de cortafuegos o un conjunto de reglas que cubran las necesidades en general. La configuración del firewall de una máquina puede ser muy variada, ya que ésta debe atender a las necesidades y la realidad de la red en la que se ubica. A continuación,

únicamente se muestran los aspectos más elementales de la configuración de UFW para firewall que corra en una máquina con el núcleo de Linux. Es por tanto responsabilidad de cada administrador de sistemas la elección de la política de cortafuegos que mejor se adapta a su infraestructura.

La instalación de UFW en cualquier máquina Linux es muy sencilla. Lo único que debemos hacer es en primer lugar actualizar los repositorios de software disponibles mediante:

```
$ sudo apt update
$ sudo apt upgrade
```

Para posteriormente instalar la herramienta desde los repositorios del sistema operativo:

```
$ sudo apt install ufw
```

Una vez instalado, ya podemos permitir el uso de UFW en el sistema. Pero cuidado, porque si la comunicación con el servidor-firewall la estamos realizando desde otro sistema a través de SSH, antes deberemos permitir el uso de SSH. De lo contrario, al permitir UFW, éste bloqueará por defecto las conexiones por SSH, obligándonos a permitir su uso directamente desde el servidor. El comando para permitir SSH es:

```
$ sudo ufw allow ssh
```

Ahora sí, podemos activar UFW con el siguiente comando:

```
$ sudo ufw enable
```

Para desactivar UFW basta con ejecutar:

```
$ sudo ufw disable
```

Si deseamos conocer el estado de las distintas reglas configuradas en UFW únicamente deberemos ejecutar el comando:

```
$ sudo ufw status verbose
```

Lo anterior nos devuelve el estado actual de UFW, así como un listado de las reglas configuradas. Una de las ventajas de UFW es que, por defecto y a diferencia de Iptables, la política de cortafuegos es permitir el tráfico saliente y denegar el tráfico entrante, salvo excepciones que pueden encontrarse mediante el comando:

```
$ sudo ufw show raw
```

De esta forma, lo ideal es configurar reglas permisivas (de tipo “Allow”) únicamente de aquel tráfico entrante que nos interese mantener. Esto se hace con los siguientes comandos:

```
$ sudo ufw allow <puerto o protocolo>
```

```
$ sudo ufw deny <puerto o protocolo>
```

De hecho, podemos forzar a permitir el uso de UDP o TCP de un puerto en concreto. Por ejemplo:

```
$ sudo ufw allow 443/tcp
```

También podemos permitir o denegar el tráfico proveniente de direcciones IP específicas hacia puertos concretos:

```
$ sudo ufw allow from <dirección IP> to <IP Raspberry> port <nº puerto>
```

```
$ sudo ufw deny from <dirección IP> to <IP Raspberry> port <nº puerto>
```

E incluso combinar puertos concretos con protocolos y direcciones IP específicas:

```
$ sudo ufw allow from <dirección IP> to any port <nº puerto> proto <protocolo>
```

Para eliminar las reglas configuradas, basta con lanzar el mismo comando de la regla en concreto con el prefijo “delete”. Por ejemplo:

```
$ sudo ufw delete allow 80
```

O bien mostrar añadir un número a cada regla mediante:

```
$ sudo ufw status numbered
```

```
Status: active
```

To	Action	From
--	-----	----
[ 1] 22	ALLOW IN	Anywhere
[ 2] 22/tcp	ALLOW IN	Anywhere
[ 3] 80	DENY IN	Anywhere
[ 4] 22 (v6)	ALLOW IN	Anywhere (v6)
[ 5] 22/tcp (v6)	ALLOW IN	Anywhere (v6)
[ 6] 80 (v6)	DENY IN	Anywhere (v6)

Para posteriormente eliminar la regla con el número deseado:

```
$ sudo ufw delete 3
Deleting:
deny 80
Proceed with operation (y|n)? y
Rule deleted
```

Respecto a reglas concretas que ayudan a mejorar la seguridad de nuestra red, debemos mencionar la opción de “Rate Limit” que nos ofrece UFW. Mediante reglas “limit” podemos limitar la cantidad de veces que una única dirección IP puede utilizar alguno de los puertos de nuestro firewall. Por ejemplo, podríamos limitar el uso de SSH mediante:

```
$ sudo ufw limit ssh
```

Así, podremos evitar ataques de fuerza bruta hacia los servicios que están corriendo en la red. Esto puede hacerse con UFW tal y como se ha mostrado, sin embargo, podemos combinar UFW con otra herramienta llamada “fail2ban”, cuyo cometido es la prevención de ataques de fuerza bruta mediante su detección y bloqueo.

#### *2.2.2.2. Fail2ban como soporte a UFW*

La configuración de fail2ban es sencilla. Únicamente deberemos instalar la herramienta [15]:

```
$ sudo apt install fail2ban
```

Una vez instalada, podremos entrar a modificar la configuración de fail2ban, ubicada en “/etc/fail2ban/jail.conf”:

```
$ sudo nano /etc/fail2ban/jail.conf
```

Por defecto, fail2ban creará una regla de UFW cuando detecte que una dirección IP ha fallado el inicio de sesión al menos cinco veces (“maxretry”) en un período de diez minutos (“findtime”), bloqueando la IP durante diez minutos (“bantime”). En este punto sería recomendable incrementar “bantime” al menos a una hora:

```

# "bantime" is the number of seconds that a host is banned.
bantime = 1h

# A host is banned if it has generated "maxretry" during the last "findtime"
# seconds.
findtime = 10m

# "maxretry" is the number of failures before a host get banned.
maxretry = 5

```

Figura 17. Configuración básica de jail.conf (fail2ban)

En cuanto a los servicios que puede monitorizar fail2ban, éstos se llaman “Jails”, y pueden ser consultados y modificados en el propio “jail.conf” (al final del archivo). Por defecto, fail2ban trae jaulas de multitud de servicios, como SSH:

```

[sshd]

# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode = normal
port = ssh
logpath = %(sshd_log)s
backend = %(sshd_backend)s

```

Figura 18. Jaula para el servicio SSH

En el directorio “/etc/fail2ban/filter.d” podemos encontrar archivos de configuración de los servicios que incluye fail2ban por defecto. Pero pueden ser añadidos más servicios siguiendo la misma estructura.

### 2.2.2.3. Securización de SSH

SSH es probablemente el servicio más usado en dispositivos Raspberry Pi, dado que facilita en gran medida la configuración y uso de ésta de forma remota. Y justamente por eso, es uno de los primeros objetivos marcados por los atacantes cuando se encuentran ante un sistema basado en Raspberry Pi. Por ello, resulta indispensable llevar a cabo ciertas acciones con el objetivo de securizar el protocolo.

El archivo que debemos modificar para la configuración de SSH es “/etc/ssh/sshd\_config”. El primer elemento a modificar es el puerto que utiliza SSH. Por defecto, es el puerto 22. Podemos modificar el puerto a cualquier otro que se encuentre libre mediante:

```
$ sudo nano /etc/ssh/sshd_config
```



Y modificando la línea con el valor “port”, por ejemplo:

```
Include /etc/ssh/sshd_config.d/*.conf

Port 62245
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Por otro lado, debemos bloquear el acceso de *root* a través de SSH. Esto es porque SSH exige aportar un nombre de usuario, la dirección IP y la contraseña. Si *root* puede usar SSH, un atacante podría intentar entrar al sistema por SSH mediante “*ssh root@<dirección IP>*”, sin la necesidad de conocer ningún usuario. Esto se hace modificando la línea “*PermitRootLogin*” de la siguiente forma:

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

Respecto a la autenticación, podemos modificar también el período de gracia (“*LoginGraceTime*”), o el número máximo de intentos de autenticación (“*MaxAuthTries*”) simplemente descomentando la línea e incluyendo el valor deseado. Por último, mediante “*AllowUsers*” y “*DenyUsers*” podemos crear una lista blanca (*AllowUsers*) y una lista negra (*DenyUsers*) de usuarios.

Una vez modificado el archivo “*sshd\_config*”, deberemos reiniciar SSH mediante:

```
$ sudo systemctl restart ssh
```

Finalmente, si deseamos incrementar la seguridad de SSH un paso más, podemos configurar la autenticación mediante clave pública SSH. Por defecto, SSH utiliza usuario y contraseña para la autenticación de usuarios. Sin embargo, podemos configurar una clave SSH (mediante RSA). Para ello, deberemos permitir la autenticación tanto por contraseña como por clave pública:

```
PubkeyAuthentication yes
```

```
PasswordAuthentication yes
```

Una vez terminado el proceso, podemos mantener ambas formas o dejar de permitir la autenticación por contraseña.

El siguiente paso es generar el par de claves (pública y privada) en el sistema desde el cual realizamos la conexión. Mediante el intérprete de comandos de Windows, lo podemos hacer de la siguiente forma:

```
PS C:\Users\bluew> ssh-keygen
```

```
Generating public/private rsa key pair.
```

```
Enter file in which to save the key (C:\Users\bluew/.ssh/id_rsa):  
C:\Users\bluew/.ssh/id_rsa
```

Tras aportar una frase-contraseña se habrán generado el par de claves RSA. Ahora, debemos enviar la clave pública a nuestra Raspberry Pi mediante el siguiente comando (en Windows):

```
PS C:\Users\bluew> type $env:USERPROFILE\.ssh\id_rsa.pub | ssh  
192.168.1.42 "cat >> .ssh/authorized_keys"
```

O bien en sistemas Linux, mediante:

```
$ ssh-copy-id -i /Users/<user>/.ssh/id_rsa.pub sergi@192.168.1.42
```

Para indicar al cliente de SSH que utilice la autenticación que acabamos de configurar sin especificar la frase-contraseña o la clave, deberemos crear el archivo "config" en "<usuario>/.ssh/config", e incluir las siguientes líneas:

```
Host <IP de la Raspberry>
```

```
    IdentityFile <Ubicación de la clave privada>
```

#### 2.2.2.4. Servidor VPN

El objetivo de una VPN (*Virtual Private Network* en inglés) es crear un túnel de red sobre Internet por el cual las comunicaciones estén cifradas. Con ello, se consigue proteger la confidencialidad y la integridad de las comunicaciones en redes inseguras.

Para el caso de nuestra Raspberry Pi, la VPN nos servirá para proteger las conexiones de usuarios que accedan al sistema desde el exterior de la red en

caso de utilizarla como servidor. Por ejemplo, para situaciones de teletrabajo en las que no hagamos uso de SSH.

Existen muchas soluciones VPN de código abierto. Podríamos decir que OpenVPN es la solución por excelencia para entornos empresariales y personales. Sin embargo, otras soluciones como WireGuard están ganando peso en los últimos tiempos por su gran rendimiento y simplicidad. Ambas nos aportan la seguridad necesaria para un sistema basado en Raspberry Pi, por lo que es decisión de cada administrador usar una u otra.

Antes de entrar en la configuración, cabe remarcar que la implementación ideal de un servidor VPN deberá ser siempre realizada en una máquina concreta y dedicada a ello. En nuestro caso, y por motivos de optimización del proyecto, utilizaremos como ejemplo de instalación un servidor VPN montado en una Raspberry Pi 4.

Para el caso concreto de Raspberry Pi, se recomienda el uso de PiVPN (<https://pivpn.io/>). PiVPN facilita la instalación y puesta en marcha de un servidor VPN basado en OpenVPN o WireGuard, ya que contiene scripts de instalación de ambas soluciones.

A continuación, se explican los pasos a seguir para montar un servidor con PiVPN y WireGuard. Previo a ello, necesitaremos instalar “curl”:

```
$ sudo apt install curl
```

El siguiente paso es instalar PiVPN mediante el siguiente comando:

```
$ curl -L https://install.pivpn.io | bash
```

Una vez termine el proceso, nos aparecerá el instalador automático de PiVPN, que nos guiará durante el proceso de creación del servidor VPN. El proceso es tan simple como seguir los pasos y configurar a placer el servidor. Al final del proceso, PiVPN creará el par de claves que cifrará las comunicaciones, y terminará el proceso.

Como en otros casos, la configuración de la VPN puede ser distinta en cada sistema. Sin embargo, hay que tener cuidado con un aspecto: de cara a los clientes de la VPN, podemos usar tanto IP Pública como DNS. Si usamos IP

Pública, debemos tener en cuenta que ésta no sea dinámica, dado que generaría conflicto. Se recomienda usar DNS con servicios como DynDNS.

Para comprobar que WireGuard (en nuestro caso) ha sido instalado correctamente, basta con ejecutar:

```
$ sudo wg show
```

Habiendo montado el servidor WireGuard, es momento de configurar los distintos perfiles de cliente para que puedan conectarse al servidor. Esto se hace mediante:

```
$ sudo pivpn add
```

E indicando un nombre de usuario para el perfil de cliente. El listado de clientes puede verse con:

```
$ ls -la configs
```

Con el servidor WireGuard montado y el perfil de cliente creado, sólo falta descargar el cliente de WireGuard en la máquina cliente, y transferir el perfil de cliente (que hemos creado en el paso anterior) de la Raspberry Pi a la máquina cliente (mediante FTP, por ejemplo). Una vez importada en el cliente de WireGuard, podremos activar y desactivar el túnel desde la máquina cliente.

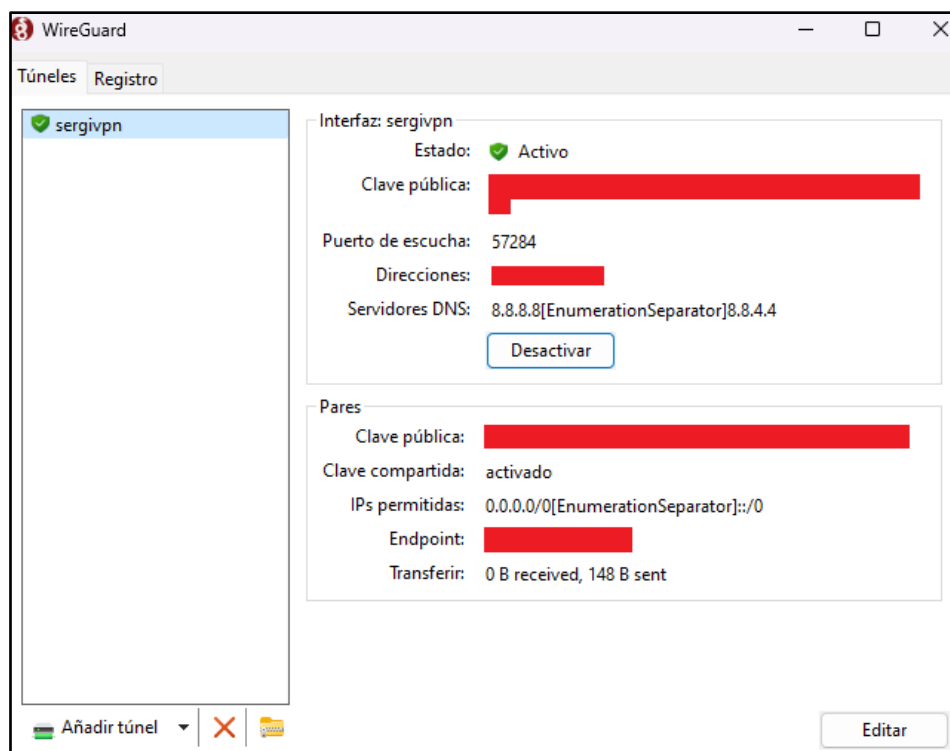


Figura 19. Estado activo del túnel VPN con WireGuard

### 2.2.3. Seguridad de red

En el apartado anterior hemos expuesto medidas de protección del perímetro de la red. Esto hace que la propia red sea más segura. Sin embargo, ello no asegura del todo la protección contra posibles adversarios. De hecho, cualquiera que lograse entrar en la red interna podría llegar a hasta nuestra Raspberry Pi sin más inconvenientes.

Por ello, debemos tomar también medidas de protección de la red interna que, si bien algunas no tienen aplicación o configuración directa en nuestra Raspberry Pi, contribuyen igualmente al bastionado del sistema.

El apartado de seguridad en red se configura de dos partes bien diferenciadas. La primera de ellas va enfocada a la protección de la Raspberry Pi (a nivel de red) en entornos no controlados, desde el punto de vista de la estrategia *Zero Trust*. La segunda parte, en cambio, se enfoca en presentar algunas soluciones ajenas a la Raspberry Pi, pero que, correctamente instaladas o configuradas en la red, contribuyen a la seguridad de nuestro sistema.

Con lo anterior, empezamos con la primera parte de la seguridad de red, que contempla la protección contra la suplantación de identidad y la configuración de la VPN WireGuard como cliente en nuestra Raspberry Pi.

#### 2.2.3.1. Protección contra suplantación de identidad

Cuando un atacante se encuentra en una red interna, uno de los tipos de ataque que con mayor probabilidad podría llevar a cabo es aquel que utiliza la suplantación de identidad como medio para lograr configurar un entorno de hombre en el medio, más conocido como *Man In The Middle*, MITM. Algunas de estas técnicas de suplantación de identidad afectan a las tablas ARP (técnica conocida como ARP Poisoning), o utilizan servidores DHCP (como el DHCP Spoofing), entre otras técnicas. A continuación, se muestran algunas acciones con el objetivo de limitar las posibilidades de establecer un entorno MITM [6].

La primera medida de seguridad se enfoca en la protección contra ARP Poisoning, mediante la cual un atacante es capaz de envenenar las tablas de caché ARP de los endpoints víctimas del ataque, redirigiendo el tráfico saliente y entrante de la víctima hacia y desde la máquina atacante. Esto se puede hacer

mediante el establecimiento de entradas ARP estáticas. Para ello, bastaría con establecerlo en cada arranque del sistema mediante:

```
$ arp -s 192.168.1.42 DC:A6:32:17:45:87
```

Aunque la medida más eficaz es crear un archivo que contenga todas las relaciones de direcciones MAC y direcciones IP de los dispositivos de la red (incluida la de nuestra Raspberry Pi, evidentemente). El fichero sigue el siguiente esquema:

```
DC:A6:32:17:45:87 192.168.1.42
```

```
5C:A6:E6:8E:A7:77 192.168.1.1
```

Una vez creado el archivo (en *"/etc/staticarp"* por ejemplo), éste deberá ser ejecutado en cada inicio del sistema mediante:

```
$ arp -f /etc/staticarp
```

La anterior medida puede ser muy efectiva en entornos de red con pocos sistemas. Sin embargo, puede ser tediosa en entornos de red más grandes. Aun así, se recomienda aplicarlo en todos los entornos. Por otro lado, es una medida que, como ocurría con el bloqueo de puertos USB, no es persistente y debe ser realizada en cada reinicio como se ha indicado. Por ello, se recomienda encarecidamente crear un binario que ejecute la tarea, o incluir el comando anterior al binario *"startconfig"*, creado anteriormente para el bloqueo de USB. Con ello, podremos automatizar la ejecución del binario con *rc.local*. Por ejemplo, el binario *"startconfig"* con esta configuración nos quedaría de la siguiente forma:

```
#!/bin/bash -e

sudo uhubctl -l 1-1 -p 1 -a off && uhubctl -l 1-1 -p 2 -a off &&
uhubctl -l 1-1 -p 3 -a off && uhubctl -l 1-1 -p 4 -a off

echo "Todos los puertos USB han sido deshabilitados"

sudo arp -f /etc/staticarp

exit
```

Como se ha añadido el comando en el mismo binario que ejecuta también el bloqueo de puertos USB, no necesitaríamos modificar *rc.local*, pues éste apunta al binario. Si por el contrario hemos creado un binario únicamente para ARP, basta con añadir la ruta del binario en cuestión a *rc.local*, tal y como se ha indicado en el apartado de bloqueo de puertos y conexiones externas.

Las entradas ARP estáticas son efectivas por si solas, pero no ofrecen información sobre ataques al administrador de la red. Por ello, Álvarez y González recomiendan su uso junto con la monitorización de ARP con la herramienta “*arpwatch*”. En distribuciones Debian (como Raspberry Pi OS), puede ser instalada mediante:

```
$ sudo apt install arpwatch
```

Una vez instalado, y siguiendo las indicaciones del archivo */etc/arpwatch/README*, podemos configurar una dirección de correo electrónico para que nos notifique de eventos de *arpwatch*. Esto se hace con los siguientes comandos:

```
$ sudo su
# echo 'IFACE_ARGS="-m sevaghi@uoc.edu"' > /etc/arpwatch/eth0.iface
```

Como se puede ver, mediante “-m” podemos indicar el correo electrónico al que *arpwatch* enviará los avisos de ataques detectados. De todas formas, para ello necesitaríamos configurar un servidor de correo en nuestra Raspberry Pi, como por ejemplo Postfix.

Respecto a los ataques que utilizan servidores DHCP maliciosos, la única forma viable de evitarlos en endpoints es mediante la configuración del fichero */etc/dhcp/dhclient.conf*, con el cual podemos crear una lista negra de servidores DHCP maliciosos. Esto se hace de la siguiente forma:

```
$ sudo nano /etc/dhcp/dhclient.conf
```

Y añadimos los DHCP maliciosos en el fichero con el siguiente formato:

```
reject <Dirección IP>;
```

Una vez guardados los cambios tendremos configurada la lista negra de servidores DHCP.

Por último, otra de las vías para establecer un entorno MITM es el abuso de ICMP Redirect, mediante el cual se puede llegar a modificar la tabla de ruta de un dispositivo en favor del atacante. Para evitarlo, basta con no aceptar las redirecciones por ICMP. Esto se hace editando el valor del fichero `/proc/sys/net/ipv4/conf/all/accept_redirects` al valor "0", indicando que no se desea aceptar redirecciones por ICMP:

```
$ sudo nano /proc/sys/net/ipv4/conf/all/accept_redirects
```

En las versiones más actuales de Raspberry Pi OS, este archivo ya está configurado por defecto con el valor 0, por lo que la redirección por ICMP no debe ser una preocupación.

#### *2.2.3.2. Raspberry Pi como VPN-client con WireGuard*

En el apartado anterior sobre seguridad perimetral hemos tratado la configuración de un servidor VPN con el objetivo de proteger las comunicaciones con usuarios externos. Ahora, volvemos a retomar la tecnología de las VPN, pero del lado del cliente. La idea es configurar nuestra Raspberry Pi para que haga de cliente de un servidor VPN (por ejemplo, el que hemos creado anteriormente y que se encuentra en nuestra red privada), con el objetivo de proteger las comunicaciones de la Raspberry en redes públicas o redes de dudosa confianza.

Para ello, volveremos a utilizar WireGuard. Sin embargo, la instalación varía a la expuesta anteriormente, ya que ahora no haremos uso de PiVPN.

La instalación de WireGuard como cliente en Raspberry Pi OS se puede realizar desde los repositorios oficiales [16]:

```
$ sudo apt install wireguard
```

Una vez instalado, debemos obtener la clave pública de nuestro servidor VPN. Para servidores VPN de WireGuard, podemos conocer la clave pública con el siguiente comando, que deberá ser ejecutado en la máquina que hace de servidor:



```
$ sudo wg show wg0
interface: wg0
  public key: ZVgEdCidKPJ9MD1IH0istx8dRZZ+TkH8Zv9j8niGQ=
  private key: (hidden)
  listening port: 51820
```

Del lado del cliente, debemos crear ahora el par de claves del cliente. Para ello, es recomendable proteger la clave privada en lo que a permisos se refiere. Lo anterior puede hacerse con *umask*, que permite establecer los permisos por defecto de nuevos archivos. Para este caso, podemos crear la clave privada a la vez que establecemos los permisos con *umask*:

```
$ (umask 077 && wg genkey > wg-private.key)
```

A continuación, creamos la clave pública del cliente:

```
$ wg pubkey < wg-private.key > wg-public.key
```

Para el siguiente paso, necesitaremos conocer la clave privada del cliente, por lo que es buen momento para imprimirla por consola:

```
$ cat wg-private.key
MKiTn+gXr4M6PQqHg+Y6RSakzLhRGWxCs/9GPm5IvUM=
```

El siguiente paso es crear el archivo de configuración del servicio de cliente de WireGuard, que debe encontrarse en la ruta */etc/wireguard/wg0.conf*. Basta con ejecutar:

```
$ sudo nano /etc/wireguard/wg0.conf
```

E incluir la información generada del lado del servidor. Para ello, debemos haber creado un perfil de cliente en el servidor (proceso que se puede ver en el apartado 2.2.2.4). La información que debemos copiar en nuestro archivo de configuración se encuentra en el directorio */etc/wireguard/configs/* de la máquina servidor. Para conocer la información podemos realizar un *cat* (como *root*) al fichero:

```
$ cat /etc/wireguard/configs/Sergi.conf
[Interface]
PrivateKey = uFGcgoBpzNMGJN7q1UZwtc0+r8X04GbUCMbiYN7owGU=
Address = 10.200.178.2/24
DNS = 9.9.9.9, 149.112.112.112
```

```
[Peer]
PublicKey = ZVgEdCidKPJ9MDlIH0istx8dRZEEZ+TkH8Zv9j8niGQ=
PresharedKey = oj6cuf8XDqpL92CQiUqxH8Ix50gdH01/CwNNG/gP5F8=
Endpoint = 79.156.185.255:51820
AllowedIPs = 0.0.0.0/0, ::0/0
```

La anterior información debe ser incluida en el fichero `/etc/wireguard/wg0.conf` de la máquina cliente.

Sumado a lo anterior, debemos crear un dispositivo de red de WireGuard también en la máquina cliente, que inicie el túnel de forma automática al iniciar el sistema, y que utilice una interfaz IPv4 con dirección estática:

```
$ sudo nano /etc/network/interfaces.d/wg0
```

El fichero `/etc/network/interfaces.d/wg0` debe contener las líneas que se muestran a continuación. Únicamente se debe modificar la dirección IP del cliente.

```
# Creación de wg0 al inicio del sistema
auto wg0

# wg0 como interfaz IPv4 con dirección estática
iface wg0 inet static

    # Dirección IP del cliente en la red de WireGuard
    address 10.200.178.2/24

    # Creación del dispositivo con comando de link de ip
    pre-up ip link add $IFACE type wireguard

    # Añade el archivo de configuración del cliente de wg
    pre-up wg setconf $IFACE /etc/wireguard/~IFACE.conf

    # Borra la interfaz wg0 al terminar el servicio
    post-down ip link del $IFACE
```

En la máquina del servidor VPN, si hemos creado el cliente o usuario con PiVPN, nos encontraremos que la configuración del archivo de servicio de WireGuard, que se encuentra en “/etc/wireguard/wg0.conf”, está debidamente configurado. Si por el contrario lo hemos hecho de forma manual con WireGuard, debemos modificar el archivo para incluir un apartado de “Peer”, tal y como se muestra a continuación:

```
[Interface]
PrivateKey = 8LCTb6eoCmWMpgbSVVWkApQyADcTPGqImWsPF5NiYLM=
Address = 10.200.178.1/24
MTU = 1420
ListenPort = 51820
### begin Sergi ###
[Peer]
PublicKey = BG7irqeQCiKdxnd10WyXlNoQ9VfpDupxFVxy3C4UsC0=
PresharedKey = oj6cuf8XDqpL92CQiUqxH8Ix50gdH0L/CwNNG/gP5F8=
AllowedIPs = 10.200.178.2/32
### end Sergi ###
```

Figura 20. Configuración de wg0.conf (Servidor)

Una vez tenemos configurados tanto el cliente como el servidor, podemos activar el túnel del lado del cliente. Esto se hace mediante:

```
$ sudo ifup wg0
```

Con lo anterior, ya tendremos listo el túnel privado que nos permitirá conectarnos de forma segura con nuestra Raspberry Pi en redes públicas.

Ahora, pasemos a explicar las soluciones que podemos implementar y configurar en la red en la que se encuentre nuestra Raspberry Pi, con el objetivo de contribuir a su securización.

### 2.2.3.3. Redes de área local virtuales (VLAN)

En redes pequeñas quizás no es un factor determinante, pero la segmentación de una red física de gran tamaño en varias subredes lógicas independientes es una de las medidas a tomar en lo que a seguridad de red se refiere. Para ello, se suele hacer uso del método VLAN, acrónimo del inglés *virtual local área network*. VLAN permite crear redes lógicas independientes dentro de una misma red física, pudiendo crear redes independientes de distintas áreas del negocio, aislar zonas, o crear zonas restringidas con el objetivo de reducir el impacto de un posible ataque lo máximo posible [6].

A la hora de diseñar la topología de red de área local virtual deseada, ésta puede desplegarse mediante distintos elementos. Sin embargo, la más recomendada cuando se decide segmentar la red por motivos de seguridad es el uso de máquinas Linux para tal cometido [6]. Estas máquinas Linux pueden ofrecer una segmentación de red a la vez que hacen funciones de enrutador y cortafuegos, incrementando notablemente el aislamiento de cada una de las VLAN.

No tendría sentido configurar nuestra Raspberry Pi para la configuración de VLAN. Primero de todo porque lo más seguro es que se trate de un *endpoint*. Y, por otro lado, porque la Raspberry Pi 4 está muy limitada en lo que a interfaces de red se refiere. Por ello, lo ideal es utilizar para tal cometido una máquina Linux que cuente con la capacidad de ampliar las interfaces de red Ethernet.

#### *2.2.3.4. Instalación de un IDS en la red*

Para terminar, nos queda hablar de uno de los elementos con mayor impacto en el control de la red: el IDS. Un *Intrusion Detection System* es un dispositivo cuyo cometido es la supervisión de una red ante actividades maliciosas [9]. Éstos se presentan en dos tipos principales atendiendo al alcance de supervisión: IDS que trabajan a nivel de *hosts* (conocidos como HIDS), e IDS que trabajan a nivel de red (conocidos como NIDS). Evidentemente, para la seguridad de la red, nos interesará utilizar un NIDS.

Existen varios enfoques en lo que a posicionamiento del NIDS se refiere. Sin embargo, para la seguridad en red se recomienda su uso dentro de la red (en lugar de, por ejemplo, el perímetro de la red). A continuación, se muestra una posible configuración de un NIDS en una red.

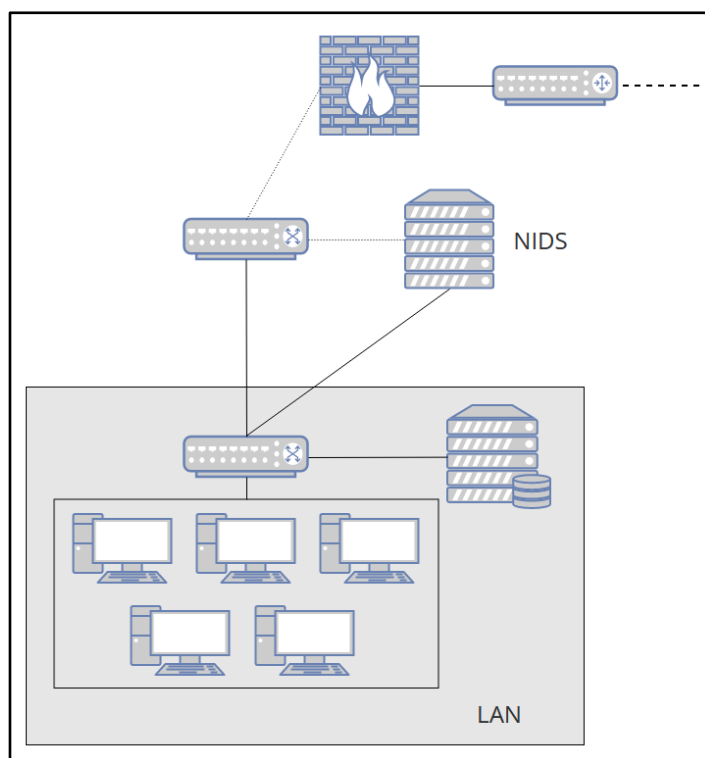


Figura 21. Ubicación de un NIDS en una posible red corporativa

Como ha ocurrido con otras medidas de protección de la red, el NIDS no debe configurarse en nuestra Raspberry Pi 4, objeto final de la securización. No obstante, en este apartado se propone el uso de una segunda Raspberry Pi que haga las veces de NIDS.

Para la configuración de la Raspberry Pi como NIDS pueden utilizarse multitud de herramientas IDS. En el mundo *Open Source*, sin duda alguna las más conocidas son Snort y Suricata. Ambas son más que suficientes para su aplicación en entornos empresariales. Sin embargo, Suricata supera en rendimiento y actualización a Snort, por lo que se recomienda encarecidamente su uso para este cometido. De hecho, Suricata puede hacer funciones tanto de IDS como de IPS (sistema de prevención de intrusiones).

Para la instalación de Suricata en sistemas Linux (en nuestro caso, una Raspberry Pi), se deben seguir los pasos mostrados a continuación, comenzando como siempre por la instalación de paquetes de Suricata:

```
$ sudo apt install suricata
```

Para verificar que se ha instalado correctamente, podemos utilizar el comando:

```
$ service suricata status
```

Que, si todo ha ido bien, nos debería devolver una respuesta como la siguiente:

```
suricata.service - Suricata IDS/IDP daemon
  Loaded: loaded (/lib/systemd/system/suricata.service; enabled;
  vendor preset: enabled)
  Active: active (running) since Thu 2023-04-06 19:49:18 CEST; 16s
  ago
  Docs: man:suricata(8)
        man:suricatasc(8)
        https://suricata-ids.org/docs/
  Process: 1444 ExecStart=/usr/bin/suricata -D --af-packet -c
  /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited,
  status=0/SUCCESS)
  Main PID: 1445 (Suricata-Main)
  Tasks: 10 (limit: 3933)
  CPU: 3.579s
  CGroup: /system.slice/suricata.service
          └─1445 /usr/bin/suricata -D --af-packet -c
  /etc/suricata/suricata.yaml --pidfile /run/suricata.pid
```

Una vez instalado Suricata, podemos ver y modificar la configuración por defecto de Suricata en el archivo `/etc/suricata/suricata.yaml`:

```
$ sudo nano /etc/suricata/suricata.yaml
```

El archivo cuenta con muchas variables que se agrupan en cuatro pasos. Para el inicio, nos interesa configurar “`HOME_NET`”, para indicar a Suricata cual es nuestra red y “`EXTERNAL_NET`”, para indicarle redes externas. Una configuración básica es identificar nuestra red como “`192.168.0.0/24`” y la red externa como cualquiera que no sea `HOME_NET` mediante “`!$HOME_NET`”, tal y como se muestra en la siguiente figura.

```
vars:
# more specific is better for alert accuracy and performance
address-groups:
  HOME_NET: "[192.168.0.0/24]"
  #HOME_NET: "[192.168.0.0/16]"
  #HOME_NET: "[10.0.0.0/8]"
  #HOME_NET: "[172.16.0.0/12]"
  #HOME_NET: "any"

  EXTERNAL_NET: "!$HOME_NET"
  #EXTERNAL_NET: "any"
```

Figura 22. Configuración por defecto de `suricata.yaml` (Suricata)

Sin embargo, la configuración de *HOME\_NET* y *EXTERNAL\_NET* debe atender a la arquitectura de la red en la que se encuentre el NIDS.

Suricata utiliza por defecto un repositorio de reglas para la identificación y detección de posibles intrusiones (recordemos que Suricata es un IDS basado en firmas). Sin embargo, pueden ser cargadas otras reglas, como por ejemplo las del proveedor *Emerging Threats* (<https://rules.emergingthreats.net>). Para ello, basta con modificar el archivo */etc/oinkmaster.conf*, añadiendo la URL que contenga las reglas que nos interesa cargar en nuestro Suricata:

```
# Example for rules from the Emerging Threats site (previously known as Bleeding Snort).
# url = http://www.emergingthreats.net/rules/emerging.rules.tar.gz
# Old url:
# url = http://www.bleedingsnort.com/downloads/bleeding.rules.tar.gz
url = https://rules.emergingthreats.net/open/suricata-5.0/emerging-all.rules.tar.gz
```

Figura 23. *Oinkmaster.conf* modificado para cargar reglas concretas

De igual forma, podemos descomentar las líneas pertenecientes a aquellos sitios que sean de nuestro interés, como las reglas de la comunidad y de Snort.

Respecto a las reglas en funcionamiento del NIDS, éstas pueden ser activadas o desactivadas en el archivo de configuración */etc/suricata/suricata.yaml* (simplemente comentando y descomentando las líneas correspondientes). El uso de unas u otras depende de la red en la que se encuentre el NIDS, por lo que es responsabilidad de cada administrador activar aquellas que mejor se adecuen a la red.

Para poder actualizar las reglas de Suricata, basta con ejecutar el siguiente comando:

```
$ sudo oinkmaster -o /etc/suricata/rules/
```

Por último, debemos configurar el sistema para que el servicio de Suricata arranque junto con el propio sistema. Por defecto en Raspberry Pi OS se encuentra activado como se puede ver a continuación:

```
$ sudo systemctl is-enabled suricata
enabled
```

Pero si no fuera el caso, ello se puede activar con el siguiente comando:

```
$ sudo systemctl enable suricata
```

Con ello, ya tendremos una máquina NIDS basada en Raspberry Pi y Suricata que monitorice la red, notificando de eventos de posibles intrusiones.

El NIDS podría funcionar por sí solo sin necesitar ninguna otra herramienta. Sin embargo, es muy recomendable instalar junto al NIDS una herramienta de monitorización e información de los eventos de seguridad detectados por el NIDS. Esto se conoce como SIEM (*Security Information and Event Manager*). Existen muchas soluciones SIEM, pero la que se propone aquí es el uso de *Elastic (ELK) Stack*, desarrollada por Elastic y que combina las herramientas ElasticSearch, Filebeat y Kibana [17].

Respecto a la instalación y configuración del SIEM, ésta se encuentra fuera del alcance de esta guía de securización. Sin embargo, cabe remarcar varios aspectos en lo que a la instalación del SIEM se refiere. Para el escenario planteado, se debería seguir lo siguiente:

- Raspberry Pi NIDS: debe tener instalado Suricata y la herramienta Filebeat, encargada de analizar el registro *eve.json* de Suricata y enviar los eventos a la máquina que corre ElasticSearch.
- *Host* del SIEM: el *host* del SIEM puede ser la Raspberry Pi 4 objeto de la securización o cualquier otra máquina Linux o Windows. Debe tener instaladas las herramientas ElasticSearch y Kibana, encargadas de la monitorización de los eventos detectados por Suricata y enviados por Filebeat.

#### 2.2.4. Seguridad del dispositivo

Habiendo asegurado la red interna con las medidas anteriores, es el momento de poner el foco en la seguridad de nuestra Raspberry Pi 4 a nivel de dispositivo. Como se podrá ver, muchas de las herramientas o metodologías ya han sido aplicadas en otros apartados. Sin embargo, esta vez se configurarán para la protección del dispositivo.

A continuación, se presentan varias herramientas de securización de la Raspberry Pi. Respecto a ellas, deben ser tomadas como distintas opciones a escoger, ya que el uso de una o varias de estas herramientas en un mismo dispositivo dependerá de las necesidades de seguridad de cada dispositivo.



#### 2.2.4.1. ClamAV como solución EPP

El software antivirus, también conocido como *Endpoint Protection Platform* (EPP) es una de las soluciones de seguridad del dispositivo con mayor trayectoria y la más conocida. Por ello, no podía faltar en la guía de bastionado un ejemplo de antivirus efectivo y de código abierto para nuestra Raspberry Pi.

Como solución EPP se ha decidido utilizar la herramienta “ClamAV” (<https://www.clamav.net/>), antivirus *open source* y multiplataforma que cuenta con un gran prestigio en el sector.

Su instalación es muy sencilla y puede estar configurado en la Raspberry Pi en pocos minutos. Como en otras ocasiones, la instalación puede realizarse desde los repositorios oficiales de Raspberry Pi OS:

```
$ sudo apt update && sudo apt full-upgrade
$ sudo apt install clamav clamav_daemon
```

Tras unos segundos la herramienta (y su demonio) habrá sido instalada correctamente. Entrando ahora con la operativa de la herramienta, ésta es especialmente sencilla. ClamAV nos permite realizar escaneos generales o escaneos concretos de directorios y ficheros, así como eliminar directorios/ficheros infectados o moverlos a zonas de cuarentena para su posterior análisis. A continuación se expone la forma en la que podemos realizar estas opciones básicas.

Para realizar escaneos generales del sistema con ClamAV debemos ejecutar el siguiente comando:

```
$ sudo clamscan -r /
```

La opción “-r” hace que el escaneo sea recursivo. El resultado del escaneo general nos devolverá un resumen del escaneo, como el que puede verse a continuación:

```
----- SCAN SUMMARY -----
Known viruses: 8665533
Engine version: 0.103.8
Scanned directories: 13503
Scanned files: 47052
```

```
Infected files: 0
Total errors: 15818
Data scanned: 2695.39 MB
Data read: 3117.94 MB (ratio 0.86:1)Time: 1575.493 sec (26 m 15 s)
Start Date: 2023:05:05 14:56:31
End Date: 2023:05:05 15:22:47
```

Para realizar escaneos concretos de directorios o ficheros, podemos añadir al anterior comando las distintas opciones. Por ejemplo, si deseamos escanear un fichero concreto y el fichero de historial de Bash, podemos ejecutar:

```
$ sudo clamscan -r /home/sergi/prueba.sh /home/sergi/.bash_history
/home/sergi/prueba.sh: OK
/home/sergi/.bash_history: OK

----- SCAN SUMMARY -----
Known viruses: 8665533
Engine version: 0.103.8
Scanned directories: 0
Scanned files: 2
Infected files: 0
Data scanned: 0.01 MB
Data read: 0.00 MB (ratio 2.00:1)
Time: 55.504 sec (0 m 55 s)
Start Date: 2023:05:05 14:47:48
End Date: 2023:05:05 14:48:44
```

Si detectamos algún fichero o directorio infectado, podemos eliminarlo con ClamAV directamente (aunque éste únicamente lo eliminará si verdaderamente está infectado). Para ello, basta con añadir la opción “*—remove*” al comando de escaneo:

```
$ sudo clamscan -r --remove /home/sergi/
```

Si lo que deseamos es mover el archivo a otra ubicación en lugar de eliminarlo, podemos hacer uso de la opción “*—move=*”, indicando a continuación la nueva ruta:

```
$ sudo clamscan -r --move=/home/sergi/infected /home/sergi/
```

Si queremos guardar el resultado de los escaneos, podemos hacerlo en nuevos archivos en nuestro sistema con la opción “-l” de la siguiente forma:

```
$ sudo clamscan -r /home/sergi -l /home/sergi/scan.txt
```

Otra opción útil es “-i”, que hace que ClamAV únicamente muestre por pantalla los archivos infectados.

A parte de estas opciones, ClamAV cuenta con muchas otras opciones, que pueden ser consultadas con el siguiente comando:

```
$ clamscan --help
```

Para terminar, podría ser una buena práctica realizar escaneos de forma periódica y con asiduidad en aquellos directorios más importantes de nuestra Raspberry Pi. Dado que lo anterior puede ser tedioso si se realiza de forma manual, podemos automatizar la ejecución de los escaneos con crontab. Por ejemplo, si queremos realizar un escaneo cada día, que elimine los archivos infectados y guarde el resultado en un archivo, podemos añadirlo directamente en */tmp/crontab.cmdUPE/crontab*, de la siguiente forma:

```
$ sudo crontab -e
```

Y añadimos al final del documento la siguiente línea:

```
0 0 * * * sudo clamscan -r --remove / -l /home/sergi/scan.txt
```

En el comando anterior, los cinco primeros caracteres hacen referencia a lo siguiente (ordenado de izquierda a derecha):

- 1) Minuto de ejecución: valores de 0 a 59
- 2) Hora de ejecución: valores de 0 a 23
- 3) Día del mes: valores de 1 a 31
- 4) Mes: valores del 1 al 12
- 5) Día de la semana: valores de 0 a 6, siendo 0 el Domingo y 6 el Sábado

Para indicar cualquier valor, se utiliza el asterisco (\*).

Otra forma de automatizar con crontab puede hacerse creando un script en Bash que contenga el comando o instrucción de escaneo con ClamAV, e incluyendo la ejecución de ese script en crontab. Un ejemplo de ello puede verse en el apartado 2.2.6.3, sobre medidas de recuperación de la información.

#### 2.2.4.2. OSSEC como solución HIDS

En el apartado de seguridad de la red ya se ha trabajado con Suricata, un IDS/IPS cuyo objetivo era la protección de la red interna. Si bien Suricata (o en su defecto, Snort) puede ser también usado como IDS a nivel de *host*, su instalación y configuración ya ha sido tratada.

Por lo anterior, se propone utilizar la herramienta OSSEC como solución HIDS para nuestra Raspberry Pi. OSSEC es un HIDS de código abierto desarrollado por Atomicorp, y que incluye las siguientes funcionalidades [18]:

- Detección de intrusiones basada en *Logs*;
- Detección de *Rootkits*;
- Detección de software malicioso;
- Respuesta activa;
- Auditoría de cumplimiento;
- Monitorización de la integridad de los archivos;
- Inventario del sistema.

La instalación de OSSEC en Raspberry Pi OS es sencilla, dado que cuenta con un asistente de instalación que facilita su configuración. Si lo deseamos, también puede instalarse de forma manual.

Antes de instalar OSSEC, debemos instalar las librerías y herramientas necesarias para correr el script. Esto se hace mediante:

```
$ sudo apt install libz-dev libssl-dev libpcrc2-dev build-essential  
libsystemd-dev
```

Para la instalación, basta con hacer un *wget* de OSSEC mediante:

```
$ wget https://github.com/ossec/ossec-hids/archive/3.7.0.tar.gz
```

A continuación, descomprimos el archivo:

```
$ tar -zxvf 3.7.0.tar.gz
```

Nos movemos al directorio de OSSEC:

```
$ cd ossec-hids-3.7.0/
```

Y ya podemos proceder a su instalación mediante:

```
$ sudo ./isntall.sh
```

Se nos abrirá el asistente de instalación, el cual nos pedirá elegir un idioma para el asistente, así como la versión de OSSEC que deseamos instalar. Para nuestro entorno la mejor opción es usar la versión servidor. Sin embargo, en entornos con un número de *hosts* mayor, quizás sea más eficaz combinar la versión servidor con la versión agente.

Por otro lado, el asistente de instalación nos preguntará si deseamos utilizar las distintas funcionalidades que ofrece OSSEC. Lo ideal es habilitar todas ellas para conseguir la máxima eficacia, por lo que basta con indicar en todas las preguntas del asistente que sí deseamos habilitar las funcionalidades.

Una vez configuradas las preferencias, nos pedirá presionar *Enter* para iniciar la instalación. Un ejemplo de configuración sería el siguiente:

```
OSSEC HIDS v3.7.0 Guión de instalación - http://www.ossec.net
```

```
Usted va a comenzar el proceso de instalación de OSSEC HIDS.
```

```
Usted debe tener un compilador de C previamente instalado en el sistema.
```

```
- Sistema: Linux raspberrypi 6.1.19-v8+  
- Usuario: root  
- servidor: raspberrypi
```

```
-- Presione ENTER para continuar ó Ctrl-C para abortar. --
```

```
1- Que tipo de instalación desea (servidor, agente, local ó ayuda)?  
servidor
```

```
- Usted eligió instalación de Servidor.
```

2- Configurando las variables de entorno de la instalación.

- Eliga donde instalar OSSEC HIDS [/var/ossec]:

- La instalación se realizará en /var/ossec.

3- Configurando el sistema OSSEC HIDS.

3.1- Desea recibir notificación por correo electrónico? (s/n) [s]: s

-Cuál es su dirección de correo electrónico? sevaghi@uoc.edu

- Hemos encontrado su servidor de correo (SMTP):  
aspmx.l.google.com.

- Desea usarlo? (s/n) [s]: s

--- Usando el servidor SMTP: aspmx.l.google.com.

3.2- Desea Usted agregar el servidor de integridad del sistema?  
(s/n) [s]: s

- Ejecutando syscheck (servidor de integridad del sistema).

3.3- Desea Usted agregar el sistema de detección de rootkit? (s/n)  
[s]: s

- Ejecutando rootcheck (sistema de detección de rootkit).

3.4- Las respuestas activas le permitirán ejecutar un comando  
específico en base a los eventos recibidos. Por ejemplo,

Usted podra bloquear una dirección IP ó deshabilitar el acceso  
de un usuario específico.

Más información en:

<http://www.ossec.net/docs/docs/manual/ar/index.html>

- Desea habilitar respuesta activa? (s/n) [s]: s

- Respuesta activa habilitada.

- Por omisión podemos habilitar el bloqueo del servicio o el descarte del paquete por medio del Firewall.

El bloqueo del servicio agregará al atacante en el archivo etc/hosts.deny

y el decarte del paquete añadirá la regla en iptables

(si el sistema fuera linux) ó ipfilter (si el sistema fuera Solaris, FreeBSD or NetBSD).

- Las dos repuestas pueden ser utilizadas para detener un escaneo de fuerza bruta contra SSHD, escaneo de puertos y otras formas de ataque. Por ejemplo se podrá agregar a los atacantes de acuerdo a eventos registrados por medio de snort.

- Desea habilitar la respuesta desechar en el Firewall? (s/n) [s]: s

- Respuesta desechar en el Firewall habilitada (local) para niveles >= 6

-

- 192.168.1.1

- Desea Usted agregar más IPs a la lista blanca? (s/n)? [n]: n

```
3.5- Desea Usted habilitar syslog remoto (puerto 514 udp)? (s/n)
[s]: s
```

```
- Syslog remoto habilitado.
```

```
3.6- Estableciendo la configuración para analizar los siguientes
registros:
```

```
-- /var/log/messages
-- /var/log/auth.log
-- /var/log/syslog
-- /var/log/dpkg.log
```

```
- Si desea monitorizar algún otro registro, solo
tendrá que editar el archivo ossec.conf y agregar una
nueva entrada de tipo localfile.
Cualquier otra pregunta de configuración podrá ser
respondida visitandonos en línea en http://www.ossec.net .
```

```
--- Presione ENTER para continuar ---
```

Tras unos minutos, OSSEC habrá sido instalado en nuestra Raspberry Pi 4 en su versión servidor. Para iniciar el HIDS, basta con ejecutar:

```
sergi@raspberrypi:~ $ sudo /var/ossec/bin/ossec-control start
Starting OSSEC HIDS v3.7.0...
Started ossec-maild...
Started ossec-execd...
Started ossec-analysisd...
Started ossec-logcollector...
Started ossec-remoted...
Started ossec-syscheckd...
Started ossec-monitord...
Completed.
```



Y para detener la herramienta, podemos hacerlo con:

```
sergi@raspberrypi:~ $ sudo /var/ossec/bin/ossec-control stop
Deleting PID file '/var/ossec/var/run/ossec-remoted-16496.pid' not
used...
Killing ossec-monitord ..
Killing ossec-logcollector ..
ossec-remoted not running ..
Killing ossec-syscheckd ..
Killing ossec-analysisd ..
Killing ossec-maild ..
Killing ossec-execd ..
OSSEC HIDS v3.7.0 Stopped
```

Finalmente, si lo que deseamos es modificar algún aspecto de la configuración del HIDS, podemos hacerlo en el archivo `/var/ossec/etc/ossec.conf`, al que podremos acceder como *root* mediante:

```
$ nano /var/ossec/etc/ossec.conf
```

#### 2.2.4.3. OSSEC+ como solución EDR

La herramienta básica de OSSEC es más que suficiente si nuestras necesidades de seguridad únicamente requieren de la implementación de un HIDS. Pero, si lo que deseamos es dar un paso más allá en lo que a seguridad se refiere (sin salir del mundo *Open Source*), tenemos que pensar en implementar un EDR.

Para lo anterior, Atomicorp nos ofrece una solución llamada OSSEC+, que no es más que una versión ampliada y mejorada de OSSEC, ofrecida igualmente de forma gratuita. Respecto a OSSEC, la solución EDR añade [18]:

- Aprendizaje automático;
- Cifrado mediante infraestructura de clave pública (PKI);
- Monitorización de OSSEC+ con Elastic (ELK) Stack.

Al igual que ocurre con la implementación de Suricata + Elastic (ELK) Stack, vista anteriormente en el apartado de seguridad de la red interna, OSSEC+ se constituye de un hub, encargado de monitorizar a la herramienta, y un agente, que será el que se instale en la Raspberry Pi 4. Lamentablemente, el hub de

OSSEC+ no tiene soporte para Raspberry Pi OS, ya que sólo puede instalarse en CentOS7, Rocky Linux 8, RedHat Enterprise Linux 7 y 8, y Ubuntu 18 y 20. Por tanto, necesitaremos de una segunda máquina que haga las veces de hub, y que evidentemente cumpla con los requisitos de sistema operativo expuestos.

La instalación del hub de OSSEC+ es muy sencilla. Basta con ejecutar como *root*:

```
$ wget -q -O - https://updates.atomicorp.com/installers/oum | bash
```

Configuramos *oum*:

```
$ oum configure
```

Y finalmente, actualizamos *oum*:

```
$ oum update
```

Respecto a la instalación del agente OSSEC+ en nuestra Raspberry Pi 4, es exactamente la misma instalación que hemos visto en el apartado anterior (OSSEC como solución HIDS), pero en su versión agente en lugar de la versión servidor.

Finalmente, podemos agregar extensiones a OSSEC+ de forma opcional. No entraremos en analizar qué extensiones podemos añadir, ya que Atomicorp ofrece multitud de ellas, y su elección depende de las necesidades de cada sistema e infraestructura. El listado completo de extensiones para OSSEC puede ser revisado en: <https://atomicorp.com/ossec-extensions/>.

#### 2.2.4.4. Atomic OSSEC como solución XDR

En caso de necesitar una solución de seguridad de dispositivos aún más completa que OSSEC en sus modalidades HIDS o EDR, podemos optar por una de las soluciones más completas en lo que a seguridad de *hosts* se refiere: un XDR.

De nuevo, Atomicorp nos ofrece Atomic OSSEC (<https://atomicorp.com/atomic-enterprise-ossec/>), su solución XDR de pago y enfocado para empresas con exigentes requisitos de seguridad.

Respecto a sus homónimos OSSEC y OSSEC+, Atomic OSSEC añade las siguientes funcionalidades [18]:

- Consola de administración basada en Web;
- Gestión de agentes de grupo;
- Herramientas integradas de cumplimiento de CIS y SCAP;
- Más de 5.000 reglas IDS y FIM;
- Supervisión forense de integridad de archivos en tiempo real;
- Integración de proveedores de nube nativos (AWS, Azure y GCP);
- Protección integrada contra software malicioso;
- Escaneo de vulnerabilidades *zero day* basado en *host*;
- Inteligencia de amenazas globales;
- Informes y auditoría de cumplimiento;
- Controles de acceso basado en roles;
- Sistema de gestión de la configuración;
- Sistema de control de Rollback;
- Integración SIEM nativa (Splunk, Arcsight y otros);
- Integración nativa de Elastic (ELK) Stack;
- Integración de Slack, PagerDuty y Jira;
- Integración de Cloudflare;
- Enrutamiento de datos de salida;
- Integraciones de almacenamiento de datos a largo plazo;
- Módulo de inventario de sistemas y aplicaciones;
- Cifrado avanzado (PKI y Noise Socket).

Por razones evidentes, no entraremos en detalle con la implementación de Atomic OSSEC en nuestro sistema y dispositivo. Sin embargo, se recomienda encarecidamente tener en cuenta la herramienta, especialmente en aquellas situaciones en las que las necesidades de seguridad sean elevadas.

#### *2.2.4.5. Cortafuegos de dispositivo*

Tal y como se recoge en el apartado 2.2.2 sobre seguridad perimetral, las soluciones de cortafuegos pueden ser implementadas y configuradas en los dispositivos finales de una red. De hecho, es altamente recomendable que, aun teniendo un firewall perimetral en la red, los dispositivos finales de esta misma red cuenten también con un firewall de *host* debidamente configurado. Ello añade una capa adicional de filtrado que protege aún más a la Raspberry Pi.

Dado que ya se ha expuesto con anterioridad la configuración de un firewall perimetral, no volveremos a entrar en detalle con ello. Recordemos que los pasos mostrados en el apartado de seguridad perimetral pueden ser reproducidos en nuestra Raspberry Pi 4.

De igual forma, se recomienda el uso de UFW como cortafuegos de dispositivo, ya que se trata de una sencilla y efectiva solución, especialmente diseñada para *endpoints*.

Si deseamos añadir una protección extra a nuestra Raspberry Pi frente a ataques de fuerza bruta desde dentro de la red, también podemos configurar la herramienta fail2ban que recordemos, utiliza reglas de iptables (a través de UFW) para poder bloquear ataques de fuerza bruta contra determinados protocolos. De nuevo, la guía de configuración de fail2ban puede encontrarse en el apartado 2.2.2 de este mismo capítulo.

#### 2.2.5. Seguridad de la capa de aplicación

El presente capítulo tiene por objetivo ofrecer distintas medidas de seguridad que pueden ser implementadas en nuestro sistema operativo Raspberry Pi OS y que permitirán proteger la capa de aplicación. Éstas comprenden por un lado medidas de gestión (de permisos y de identidades), y por otro lado, herramientas concretas para la protección de las aplicaciones.

De nuevo, debemos recordar que la variedad de medidas y herramientas presentadas en este capítulo atiende a la necesidad de cubrir los requerimientos que pueden darse en distintos entornos profesionales. En algunos casos, bastará con la gestión de permisos e identidades. En otros casos, necesitaremos de herramientas que protejan aún más nuestras aplicaciones más críticas.

##### 2.2.5.1. Gestión de permisos y atributos

En un sistema Linux, como es el caso de Raspberry Pi OS, la gestión de permisos es una de las partes más críticas para la seguridad del sistema. Un permiso no es más que la capacidad que un usuario tiene para poder realizar una acción en archivos y directorios del sistema al que pertenece.

Por defecto en sistemas Linux los archivos y directorios tienen propietarios y pertenecen a un grupo. De igual forma, también cuentan con permisos para

usuarios, grupos y otros (cualquiera que no pertenezca ni al grupo ni sea el usuario propietario del fichero o directorio). Para conocer la información anterior de un fichero o un directorio, se utiliza “ls” junto con alguna bandera. Por ejemplo:

```
$ ls -l
total 4
drwxr-xr-x 2 sergi sergi 4096 Apr 13 20:09 sergi
```

En el ejemplo anterior, podemos ver como el directorio */home* contiene otro directorio */home/sergi*, y podemos ver información relativa a: el tipo de fichero (la primera letra que aparece, en este caso “d” de directorio); los permisos de usuario, grupo y otros (las siguientes nueve letras); que pertenece al usuario “sergi” y al grupo “sergi”; y otra información como el peso y fecha de la última modificación.

Respecto al sistema de permisos de Linux, su funcionamiento es relativamente sencillo. Éstos se dividen en permisos de lectura (representado por “r”), escritura (“w”), y ejecución (“x”). Los permisos se asignan para tres grupos de usuarios: el propietario del fichero o directorio (el primer trio de letras después de la letra que define el tipo de fichero); el grupo al que pertenece el propietario (segundo trio de letras); y el grupo de otros usuarios (tercer trio de letras). En caso de no existir un permiso, en su lugar aparece un guion (“-“), que representa la ausencia del permiso.

Para la gestión de permisos, es fundamental que el administrador del sistema modifique y adecúe los permisos de los distintos ficheros y directorios de mayor importancia para los usuarios indispensables y reduciendo al mínimo el número de usuarios con permisos elevados. La gestión de permisos en Linux puede realizarse mediante la herramienta “*chmod*”. A la hora de asignar o eliminar permisos con *chmod*, podemos hacerlo de distintas formas. La más intuitiva con lo que se ha explicado hasta ahora es el modo simbólico. Éste utiliza los siguientes caracteres:

- Caracteres para permisos: “r” de lectura, “w” de escritura y “x” de ejecución.
- Caracteres para clases de usuarios: “u” de usuario propietario, “g” de grupo, “o” de otros, y “a” de todas las clases (propietario, grupo y otros).

A la hora de añadir o quitar permisos con el modo simbólico, se utilizan los siguientes operadores lógicos:

- Operador “+” para asignar o añadir derechos sobre un fichero o directorio.
- Operador “-“ para quitar o eliminar derechos existentes sobre un fichero o directorio.
- Operador “=” para renovar permisos de un fichero o directorio.

Un ejemplo de gestión de permisos con el modo simbólico puede ser:

```
$ chmod ug+rwx fichero.txt
```

Con lo anterior, estaríamos asignando al usuario (“u”) y al grupo (“g”) los derechos de lectura (“r”), escritura (“w”) y ejecución (“x”) de *fichero.txt*.

Sin embargo, existe otro modo de gestión de permisos que, si bien conlleva un aprendizaje previo, con el tiempo es mucho más eficiente: el modo octal. En el modo octal, en lugar de caracteres y operadores se utilizan números. Para la identificación de la clase de usuario se utiliza las siguientes posiciones de los valores de permisos:

- 1ª posición: identifica al propietario del fichero o directorio.
- 2ª posición: identifica al grupo del fichero o directorio.
- 3ª posición: identifica a otros.

Respecto a los permisos, se asignan los siguientes valores:

- “4”: para la lectura (“r”).
- “2”: para la escritura (“w”).
- “1”: para la ejecución (“x”).
- “0”: para la inexistencia permisos.

De lo anterior, podemos extraer las distintas combinaciones de permisos, sumando en cada caso el valor asignado para cada permiso. Todas las combinaciones de permisos y valores son:

- “0”: sin permisos (“---“)
- “1”: sólo ejecución (“-x”)
- “2”: sólo escritura (“-w-“)
- “3”: escritura y ejecución (“-wx”)

- “4”: sólo lectura (“r--”)
- “5”: lectura y ejecución (“r-x”)
- “6”: lectura y escritura (“rw-“)
- “7”: todos los permisos (“rwx”)

A continuación, se muestra un ejemplo que utiliza el modo octal:

```
$ chmod 640 fichero.txt
```

El anterior ejemplo da permisos de lectura y escritura para el propietario (mediante el valor “6”), permisos de lectura para el grupo (mediante el valor “4”), y ningún permiso para otros (mediante el valor “0”) a *fichero.txt*.

Lo anterior hace referencia a la gestión de los tres permisos comunes. Sin embargo, existen otros permisos especiales que deben ser igualmente gestionados. Éstos aparecen únicamente en determinados ficheros, como */bin/su*. Los permisos especiales son los siguientes [6, p. 161]:

- “*setuid*”: permiso de usuario representado por el carácter “s” si es ejecutable o “S” si no lo es.
- “*setgid*”: permiso de grupo representado por el carácter “s” si es ejecutable o “S” en caso de no serlo.
- “*STI*”: permiso de otros representado por el carácter “t” si es ejecutable o “T” si no lo es.

Por ejemplo:

```
$ ls -l /bin/su
-rwsr-xr-x 1 root root 67776 Jan 20 2022 /bin/su
```

En el ejemplo, podemos ver como el carácter “s” se ubica en el lugar del carácter de ejecución. Para la modificación de los permisos especiales, basta con añadir un cuarto valor en el modo octal, siendo ahora el primer valor de los cuatro el que hace referencia al permiso especial.

Sumado a los permisos comunes y especiales, los ficheros también cuentan con atributos, que pueden ser utilizados para controlar el acceso a los archivos de forma más específica y personalizada. Existen varios atributos, pero en lo que a seguridad de los ficheros se refiere, los más interesantes son [6, p. 162]:

- *“immutable”*: se representa con el carácter *“i”*, y se utiliza para evitar la sobrescritura o borrado de un fichero concreto. De hecho, lo evita incluso para *root*.
- *“append”*: se representa con la *“a”*, y su objetivo es limitar la escritura del fichero únicamente a la agregación de contenido, evitando la sobrescritura. Es especialmente útil para la protección de los ficheros de *logs* de nuestra Raspberry Pi.

Para la modificación de atributos no se utiliza *chmod*. En su lugar, debemos utilizar la herramienta *“chattr”* en modo simbólico. Por ejemplo:

```
$ sudo chattr +i /etc/passwd
```

O bien:

```
$ sudo chattr -i /etc/passwd
```

Para conocer los atributos de un fichero en concreto, podemos utilizar *“lsattr”*. Por ejemplo:

```
$ lsattr /etc/passwd
----i-----e----- /etc/passwd
```

#### 2.2.5.2. Gestión de identidades y accesos

La gestión de identidades y accesos, o IAM por sus siglas en inglés (*Identity and Access Management*) es la combinación de políticas y aplicación de tecnologías para la correcta administración del acceso a un sistema y a su información. El concepto se construye a partir de dos procesos: la gestión de identidades y la gestión de accesos.

La gestión de identidades es un término que hace referencia a la creación, modificación y eliminación de cuentas de usuario (y de sus credenciales) dentro de un sistema.

En cambio, la gestión de accesos hace referencia al control de los permisos de acceso que cada usuario tiene en un sistema o entorno concreto. Es decir, el rol con el que cuenta y los permisos del rol concreto.



Volviendo a nuestro entorno, la gestión de identidades en la Raspberry Pi puede realizarse de forma sencilla mediante la administración de los usuarios del sistema y de sus credenciales. En este sentido, es importante tener en consideración varias cosas:

1. Los usuarios en activo en un mismo sistema deben ser los mínimos indispensables, y su existencia debe estar justificada por la propia necesidad del acceso del usuario al sistema.
2. Las políticas de credenciales de usuario deben garantizar la seguridad del inicio de sesión de cada usuario y, por ende, de la seguridad del sistema. El uso de sistemas de doble o múltiple factor de autenticación y el establecimiento de cambios periódicos de credenciales contribuyen a ello.

Con lo anterior, la gestión de accesos en Linux es sencilla. Por ejemplo, para la creación de un usuario basta con ejecutar:

```
$ sudo adduser <nombreusuario>
```

Y para eliminar un usuario en desuso, podemos:

```
$ sudo deluser <nombreusuario>
```

A la hora de eliminar un usuario, también podemos eliminar su directorio */home* o todos los archivos del usuario en el sistema con:

```
$ sudo deluser -remove-home <nombreusuario>
```

```
$ sudo deluser -remove-all-files <nombreusuario>
```

Finalmente, si lo que deseamos es modificar la configuración de un usuario que ya existe en el sistema, basta con utilizar *"usermod"*.

Por otro lado, la gestión de accesos ya ha sido tratada en parte en el apartado anterior, en el cual se ha visto cómo gestionar permisos y atributos en nuestra Raspberry Pi. Sin embargo, a ello se debe añadir la administración de roles de cada usuario. En un sistema Linux, podríamos entender los roles de cada usuario como los grupos a los cuales pertenece el usuario, y a través de los cuales éste gana permisos. Los grupos pueden ser creados y eliminados por el administrador del sistema, y éste puede asignar a cada grupo determinados permisos sobre ficheros y archivos. Los grupos en nuestra Raspberry Pi se guardan en */etc/group*, y pueden consultarse mediante:

```
$ cat /etc/group
root:x:0:sergi
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
```

Lo anterior nos devuelve un listado en el que cada fila es un grupo, y cada columna contiene información relativa al grupo:

- La primera columna es el nombre del grupo (*root*, *daemon*, *bin*...).
- La segunda columna representa mediante “x” una contraseña cifrada.
- La tercera columna es el número de identificación GID.
- La cuarta columna es un listado de los usuarios que pertenecen al grupo, separados por una coma “,” y sin espacios.

Para crear o eliminar grupos en la Raspberry Pi, basta con lanzar los siguientes comandos:

```
$ sudo groupadd <nombregrupo>
$ sudo groupdel <nombregrupo>
```

Y para añadir usuarios ya existentes a un grupo concreto, basta con ejecutar:

```
$ sudo usermod -aG <grupo1,grupo2> <usuario>
```

Como se puede ver, podemos añadir un mismo usuario a un único grupo (indicando solo un grupo después de “-aG”), o bien añadir el mismo usuario a varios grupos (indicando varios grupos separados por una coma y sin espacios).

Finalmente, podemos eliminar un usuario de un grupo mediante:

```
$ sudo deluser <usuario> <grupo>
```

Combinando lo visto anteriormente en lo que respecta a la gestión de identidades y accesos con la gestión de permisos y atributos lograremos un entorno seguro, en el que las identidades se encuentren debidamente actualizadas, y ningún usuario tenga más permisos de los que debería tener. Ello reduce en gran medida el alcance de un posible ataque o intrusión en el sistema.

### 2.2.5.3. Cortafuegos a nivel de aplicación (WAF)

Podría darse el caso en el que nuestra Raspberry Pi tuviera por objetivo la ejecución de un servidor de aplicaciones web. Esto hace que el sistema se encuentre mucho más expuesto a amenazas. No sólo por el incremento de servicios que, por probabilidad, pueden tener vulnerabilidades (y que deberán ser tratadas mediante actualizaciones periódicas e incluso parcheado). Si no también por la apertura de nuestro sistema al tráfico de Internet.

Para reducir la exposición a posibles amenazas y ataques hacia nuestra aplicación web y la Raspberry Pi existe una herramienta que ya ha sido tratada anteriormente, pero que ahora se presenta a nivel de aplicación: el firewall de aplicaciones web (WAF en adelante). El cometido del WAF no varía respecto a sus homónimos de perímetro y red: garantiza la seguridad a través del análisis del tráfico y su filtrado a partir de reglas preestablecidas. La diferencia es que un WAF lo logra mediante el análisis de los paquetes de petición HTTP/HTTPS y estableciendo modelos de tráfico.

Respecto a qué WAF utilizar existen muchos tipos: de código abierto, por licencia, e incluso como servicio desde la nube. Algunos de ellos dan cobertura únicamente a servidores web concretos, como es el caso de NAXSI con Nginx, y otros son soluciones multiplataforma. Un ejemplo de WAF multiplataforma (que es justo el que se ha escogido para este apartado) es “ModSecurity”, de TrustWave (<https://github.com/SpiderLabs/ModSecurity>). ModSecurity es una de las soluciones WAF de código abierto más utilizadas, ya que soporta servidores web de Apache, Microsoft IIS y Nginx. La herramienta viene ya preconfigurada con reglas que ayudan a proteger el servidor web de las siguientes amenazas:

- Tráfico o actividad maliciosa;
- Inyección SQL;
- XSS;
- Filtración de información;
- Troyanos.

Para obtener detalles sobre la instalación de ModSecurity en un servidor Nginx, ésta se desarrolla más adelante, en el apartado 3.3.3. En dicho apartado también se entra en detalle en lo que a configuraciones avanzadas se refiere.

La implementación de un WAF como ModSecurity es suficiente en lo que a análisis y filtrado del tráfico a nivel de aplicación web. No obstante, es bastante poco intuitivo en lo que a monitorización se refiere. Por ello, puede resultar interesante combinar la herramienta con una consola como WAF-FLE (<https://waf-fle.org>), diseñada específicamente para ModSecurity y que nos permite administrar y controlar mediante un panel gráfico la solución WAF.

#### 2.2.5.4. *Application Armor (AppArmor)*

Para terminar el apartado de seguridad a nivel de aplicación, me gustaría hacer una mención a AppArmor. Application Armor (<https://apparmor.net/>) es un módulo del kernel de Linux que permite limitar o restringir el acceso de las aplicaciones que controla a determinados recursos del sistema. De esta forma, AppArmor permite asociar, mediante el modelo de mecanismo de control de acceso (conocido como MAC por sus siglas en inglés), un perfil de seguridad para cada aplicación. Y cada uno de estos perfiles son los encargados de limitar o restringir el acceso a los recursos por medio de políticas de seguridad.

Si bien AppArmor viene integrada por defecto en la mayoría de las distribuciones con kernel de Linux, lamentablemente no es el caso de nuestra Raspberry Pi y su sistema operativo. Esto se debe a que la Raspberry Pi Foundation consideró innecesaria su integración por defecto, ya que añade cargas de trabajo extra al kernel y se consideró que muy pocos usuarios utilizarían el módulo [19].

No obstante, para aquellas situaciones en las que la necesidad de uso de AppArmor en una Raspberry Pi esté justificado, sí que es posible integrar y permitir su uso en Raspberry Pi OS. El proceso no es sencillo y requiere de cierto conocimiento sobre sistemas operativos, ya que conlleva recompilar el kernel de Raspberry Pi OS con tal de permitir AppArmor. Por ello, lo anterior escapa al alcance de este trabajo final de máster. Sin embargo, se puede encontrar más información sobre el proceso en el siguiente recurso: <https://we.riseup.net/wikis/300095#enabling-apparmor-on-a-raspberry-pi>.

### 2.2.6. Seguridad de los datos

Llegamos finalmente al último nivel de la defensa en profundidad, en el que el objetivo es velar por la protección y la seguridad de los datos y la información. Cuando el resto de las medidas de seguridad han fallado, la seguridad de los datos se ocupa de proteger la confidencialidad, disponibilidad e integridad de la información del negocio.

Los datos aportan multitud de información sobre el negocio, los trabajadores y usuarios, y sus clientes. En muchos casos, se trata de información de carácter sensible e incluso información crítica para la continuidad del negocio. Por ello, la seguridad de los datos no es únicamente la última medida de defensa, sino que es una de las más importantes para el negocio.

El concepto de dato en seguridad de la información hace referencia a cualquier información concreta recogida en soportes tanto analógicos (hojas de papel, por ejemplo) como digitales. No obstante, para el caso concreto de este apartado, las medidas y herramientas descritas se enfocan únicamente en aquellos datos e información que se encuentren en la Raspberry Pi, tanto en la microSD como en hardware de almacenamiento añadido y extraíble.

#### 2.2.6.1. Clasificación de la información

Previo a la protección de los propios datos, lo primero de todo es conocer qué tipo de datos existen dentro de nuestra Raspberry Pi, para posteriormente clasificar la información en distintos niveles. El objetivo tras la clasificación de la información es aplicar la protección más adecuada con respecto a los requerimientos de seguridad de cada tipo de dato.

Respecto a la clasificación de los datos, existen muchas propuestas de clasificación que se adaptan a distintas organizaciones y empresas. A continuación, se aportan ejemplos de clasificación general de la información en cinco niveles tanto para empresas privadas, como para administraciones públicas que traten con información clasificada. Las clasificaciones expuestas no son exclusivas para cada tipo de organismo, y de hecho, lo más importante es que la clasificación se adapte a las realidades de cada organización o empresa.

Lo correcto es, por tanto, escoger aquella que refleje con mayor fidelidad la situación de la información, modificarla para que se adapte, o incluso diseñar una nueva clasificación desde cero. Para esta última opción, es importante tener en cuenta que debe reflejar los distintos niveles de información en términos de confidencialidad.

<b>Clasificación de la información en cinco niveles</b>	
<b>Nivel</b>	<b>Descripción</b>
Información de carácter público	Se trata de datos que pueden ser accesibles a todo el mundo, por lo que no requieren de protección en lo que a confidencialidad se refiere. Sin embargo, se pueden tomar medidas en lo que a la integridad de los datos se refiere.
Información interna	Son aquellos datos que deberían ser accesibles únicamente por personal de la compañía. El objetivo de las medidas es que tal información no se conozca más allá de los trabajadores y usuarios.
Información restringida	Se trata de información interna de la empresa, pero que únicamente debe ser conocida por un departamento o área concreta de la organización.
Información confidencial	Hace refiere a los datos que únicamente deben ser accesibles y conocidos por un número muy reducido de personas dado su carácter especialmente sensible.
Información secreta	Compuesto por datos accesibles prácticamente en exclusiva a la dirección de la organización, dado que se trata de datos de carácter estratégico y especialmente crítico para la continuidad del negocio.

Clasificación de la información según el CCN-STIC-001	
Nivel	Descripción [20]
Secreto	Información cuya revelación no autorizada supone una amenaza o perjuicio extremadamente grave para los intereses del país.
Reservado	Su revelación no autorizada supone una amenaza o perjuicio grave para los intereses del país.
Confidencial	Se aplica a información cuya revelación no autorizada puede causar una amenaza o perjuicio para los intereses del país.
Difusión limitada	Información cuya revelación no autorizada puede ser contraria a los intereses del país.

Una vez establecida la clasificación de la información, podremos tomar medidas efectivas para cada nivel de la clasificación, atendiendo en cada caso a las necesidades de seguridad y protección de la información.

#### 2.2.6.2. Cifrado de los datos

El cifrado de datos es la principal medida técnica en lo que a seguridad de los datos se refiere. Ésta nos permite proteger la confidencialidad e integridad de los datos almacenados y transmitidos por nuestra Raspberry Pi, incluso cuando el resto de las defensas han fallado.

En el capítulo 2.2.1 sobre seguridad física ya hemos visto una forma de cifrar los datos contenidos en la microSD mediante la herramienta LUKS, con el objetivo de reducir el impacto de su robo o pérdida. Para ello, encriptamos la totalidad de los datos de la memoria de la microSD. Sin embargo, ahora lo que buscamos es mayor versatilidad, cifrando tanto datos en comunicación como datos almacenados.

De cara al cifrado de datos en comunicación, sin duda el protocolo más utilizado es *Transport Layer Security* (TLS en adelante), la versión actualizada del conocido *Secure Sockets Layer* (o SSL). TLS se encuentra en su versión 1.3, y su objetivo principal es ofrecer confidencialidad y privacidad de las comunicaciones en Internet. Esto lo consigue mediante certificados TLS, que permiten verificar la identidad y establecer conexiones de red cifradas entre

sistemas [21]. Seguramente la herramienta de código abierto más usada para generar certificados SSL/TLS en Linux sea OpenSSL (<https://www.openssl.org/>), la cual viene por defecto instalada en las últimas versiones de Raspberry Pi OS (64-bit), y se ubica en `/usr/lib/ssl`. Como se ha dicho antes, para establecer una comunicación segura mediante TLS (por ejemplo, para proteger el tráfico web con el protocolo HTTPS), necesitamos un certificado TLS debidamente certificado por una Autoridad de Certificación, conocida también como CA.

Hoy en día, la manera más fácil (y gratuita) de obtener certificados TLS para servidores web es Let's Encrypt (<https://letsencrypt.org/es/>), autoridad de certificación sin ánimo de lucro impulsada por el *Internet Security Research Group* o ISRG (<https://www.abetterinternet.org/>), que permite generar certificados de forma automatizada y transparente al público. Pueden encontrarse más detalles sobre cómo generar un certificado TLS con Let's Encrypt en: <https://letsencrypt.org/es/how-it-works/>.

Respecto al cifrado de datos almacenados, debemos tener presente que no sólo estamos limitados al cifrado de volúmenes enteros, tal y como vimos en el capítulo 2.2.1, sino que además podemos cifrar particiones del sistema e incluso ficheros de forma individualizada. Esto nos aporta una gran versatilidad para la protección de todas aquellas categorías de datos que podamos almacenar en la Raspberry Pi.

Referente a las herramientas a utilizar para el cifrado de volúmenes y ficheros, de nuevo existen varias soluciones de código abierto profesionales. Sin embargo, se recomienda encarecidamente utilizar LUKS o VeraCrypt (<https://www.veracrypt.fr/en/Home.html>). Ambas herramientas tienen el mismo enfoque, y el uso de una o la otra es una cuestión de gustos personales.

Dado que ya hemos explicado el uso de LUKS, aprovecharemos en esta ocasión para explicar la instalación y uso de VeraCrypt.

La instalación de Veracrypt en la Raspberry Pi no puede hacerse desde los repositorios de Raspberry Pi OS. Pero sí que se puede realizar mediante la descarga de un repositorio de Debian. Para descargar la última versión de la herramienta en un Raspberry Pi OS Lite (64-bit), ejecutamos el siguiente comando:



```
$ wget
https://udomain.dl.sourceforge.net/project/veracrypt/VeraCrypt%201.25.9/Linux/veracrypt-1.25.9-Debian-10-arm64.deb
```

Tras unos minutos, el paquete se habrá descargado en `/home/user`. Estando en el mismo directorio, podemos instalar el paquete con:

```
$ sudo apt install ./veracrypt-1.25.9-Debian-10-arm64.deb
```

Cuando termine el proceso ya tendremos VeraCrypt instalado. Para verificarlo, podemos ejecutar:

```
$ veracrypt --version
VeraCrypt 1.25.9
```

VeraCrypt puede ser usado desde su interfaz gráfica, ya que cuenta con ella. Sin embargo, si usamos la Raspberry Pi como servidor deberemos hacer uso de la herramienta a través de Bash.

El proceso de cifrado con VeraCrypt se hace de forma muy parecida para directorios y ficheros individuales. Por ejemplo, para cifrar un archivo debemos seguir los pasos siguientes.

El primer paso es crear un volumen vacío equivalente al archivo que queremos proteger. Basta con ejecutar el siguiente comando:

```
$ mkdir -p <Ruta del directorio o archivo>
```

Ahora, entramos en el asistente de creación de volúmenes de VeraCrypt, mediante el comando:

```
$ veracrypt -t -c
```

El asistente nos hará un total de diez preguntas de configuración:

- 1) Volumen visible u oculto
- 2) Ruta completa del archivo cifrado (el volumen vacío)
- 3) Tamaño del archivo a cifrar
- 4) Algoritmo de cifrado (AES es el más usado)
- 5) Algoritmo para el *hash* (SHA-512 es el más robusto)
- 6) Sistema de archivos (en nuestro caso Linux Ext4)
- 7) Contraseña de cifrado
- 8) Multiplicador de iteraciones Personales (PIM)

- 9) Ruta de archivo de claves (si deseamos utilizar claves en lugar de contraseña)
- 10) Crear base para la clave de cifrado mediante la escritura de 320 caracteres

Una vez terminada la configuración y tras una espera de varios minutos, el volumen estará listo para montar. El montaje se hace con la siguiente línea de comando:

```
$ veracrypt <carpeta a cifrar> <carpeta de montaje>
```

Si deseamos realizar el proceso contrario (desmontaje del volumen), basta con ejecutar:

```
$ veracrypt -d <volumen montado>
```

### 2.2.6.3. Medidas de recuperación de la información

La recuperación de los datos es otro de los elementos clave para la seguridad de los datos. En una organización, las medidas de recuperación de la información se abordan desde un punto de vista muy extenso, ya que combinan las políticas de seguridad y la planificación de proyectos. En nuestro caso concreto, nos enfocaremos únicamente en el diseño de un proceso de copias de seguridad y redundancia de los datos de una Raspberry Pi.

Para el diseño de la política de copias de seguridad, el primer paso es delimitar el alcance que le queremos dar a los backups:

- 1) Hacer un inventario de la información y de los datos que requieren redundancia.
- 2) Hacer un inventario de los soportes de información existentes o usados en el sistema.
- 3) Definir con qué frecuencia se realizarán las copias de seguridad.
- 4) Dónde almacenaremos las copias de seguridad: *on premises* o en la nube.
- 5) Cada cuanto validaremos las copias de seguridad, es decir, cada cuanto haremos pruebas de restauración a partir de copias de seguridad.

Habiendo definido los aspectos anteriores, podemos pasar ahora a explorar las herramientas existentes para Raspberry Pi OS, que nos permitirán realizar

copias de seguridad. En este punto, es importante saber que existen muchas herramientas que permiten realizar copias de seguridad puntuales. Sin embargo, lo que nos interesa es la automatización del proceso y, más importante todavía, que pueda ser realizado sin la necesidad de apagar el sistema ni extraer la microSD. Dentro de la automatización de las copias de seguridad, tenemos dos principales opciones: guardar backups automáticos en un soporte de información conectado físicamente a la Raspberry Pi (por ejemplo, un SSD); o bien almacenar los backups directamente en un servicio cloud (Google Drive o similares). A continuación, se muestran un ejemplo de cómo automatizar el proceso con la herramienta `raspiBackup` (<https://github.com/framps/raspiBackup>).

`raspiBackup` es herramienta de código abierto que nos permite crear copias de seguridad de una Raspberry Pi mientras está en ejecución de una forma muy simple. Para su instalación, cuenta con un asistente que facilita mucho la configuración de la herramienta. La instalación de `raspiBackup` puede hacerse con el comando:

```
$ curl -s https://raw.githubusercontent.com/framps/raspiBackup/master/installation/install.sh | sudo bash
```

Al ejecutar el comando anterior, se nos abrirá el asistente de instalación. Basta con elegir el idioma (por defecto, inglés), e iniciar su instalación. Cuando tengamos la herramienta instalada, el asistente nos dará acceso a los menús de configuración de opciones y de gestión de componentes. Dentro de la configuración de opciones, `raspiBackup` permite seleccionar:

- 1) *Backup path*: ruta de la partición en la que se guardarán las copias de seguridad.
- 2) *Backup versions*: número de copias de seguridad a almacenar.
- 3) *Backup type*: tipo de copia de seguridad (se recomienda `rsync` o `tar/tgz`).
- 4) *Backup mode*: permite modificar de que particiones queremos generar la copia de seguridad.
- 5) *Services to stop and start*: orden y los servicios que deseamos parar y reiniciar durante el proceso.
- 6) *Message verbosity*: cantidad de información mostrada por `raspiBackup`.

- 7) *Email notificación*: dirección de correo electrónico en la cual queremos recibir las notificaciones de cada backup realizado.
- 8) *Regular backup*: permite establecer una periodicidad diaria o semanal y el día y hora en el que se realizan. Si se desactiva, el proceso de backup deberá ser realizado manualmente. Para configuraciones *custom*, podemos modificar manualmente el fichero `/etc/cron.d/raspiBackup`.
- 9) *Compression*: permite comprimir las copias de seguridad para que ocupen menos espacio. Únicamente está disponible para los tipos `dd` o `tar`.

Con lo anterior, `raspiBackup` irá realizando copias de seguridad del sistema según la configuración indicada. Sin embargo, éstas únicamente pueden ser guardadas bien en particiones de la microSD, o bien en soportes externos como un SSD o USB. Si lo que queremos es lograr la máxima flexibilidad y seguridad posible, necesitaremos almacenar las copias de seguridad en otras ubicaciones externas a la Raspberry Pi, como la nube. Es cuando entran en juego las herramientas `Rclone` (<https://rclone.org/>) y `crontab` (incluido en Raspberry Pi OS). Por un lado, `Rclone` nos permitirá gestionar el almacenamiento en la nube desde `Bash`. Por el otro, con `crontab` podremos programar y automatizar las acciones de gestión de archivos en la nube realizadas con `Rclone`.

La instalación de `Rclone` se hace a través del siguiente comando:

```
$ sudo -v ; curl https://rclone.org/install.sh | sudo bash
```

Una vez instalado podemos añadir una conexión a cualquier almacenamiento en la nube (Google Drive p.e.). `Rclone` cuenta con un asistente accesible con el comando:

```
$ rclone config
```

Entre otras opciones, al crear un *“remote”* deberemos aportar datos referentes a la API del servicio concreto, como el *client\_id* o el *client\_secret*. Esto es porque `Rclone` realiza la gestión de archivos (como subir una copia de seguridad `tar/tgz` a la nube) mediante peticiones a la API del servicio escogido.

Para transferir las copias de seguridad del directorio de backups escogido con `raspiBackup`, debemos usar el comando *“rclone copyto”*. Por ejemplo:

```
$ rclone copyto <directorio backup raspiBackup> <directorio remoto nube>:
```

Si deseamos mantener una buena gestión del almacenamiento de las versiones en nuestra nube, usando únicamente el espacio justo y necesario para las copias de seguridad más actualizadas, podemos usar también Rclone. Por ejemplo, si realizamos copias de seguridad cada semana, y únicamente queremos mantener las dos últimas versiones, podemos ejecutar:

```
$ sudo rclone delete <directorío backup nube> --min-age 15d
```

Lo anterior borrará todos los backups que tengan más de 15 días de antigüedad.

Para automatizar los comandos de Rclone con crontab tenemos dos opciones:

- 1) Crear un script en Bash que contenga todas las órdenes de copiado y borrado de versiones antiguas, y llamar al script desde crontab;
- 2) O bien programar directamente en `/tmp/crontab.cmdUPE/crontab` los comandos de copiado y borrado, separados por dos espacios y un punto y coma (;).

Para la primera opción, teniendo en cuenta que realizamos una copia de seguridad de la Raspberry Pi con raspiBackup los lunes de cada semana a las 00:00 A.M, un ejemplo de script podría ser:

```
#!/bin/bash -e

# Copia del backup a la nube:
rclone copyto <directorío backup raspiBackup> <directorío backup nube>

# Borrado de versiones anteriores:
sudo rclone delete <directorío backup nube> --min-age 15d
```

Teniendo el anterior script guardado en `/home/user/scripts/backup.sh`, podemos programar su ejecución con crontab de la siguiente forma, ejecutando:

```
$ sudo crontab -e
```

E incluyendo al final del documento la siguiente línea:

```
0 0 * * 1 bash /home/user/scripts/backup.sh
```

Con la segunda opción, podemos programar directamente los comandos en crontab mediante:

```
$ sudo crontab -e
```

E incluyendo al final del documento la siguiente línea:

```
0 0 * * 1 rclone copyto <directorio backup raspiBackup> <directorio backup nube> ; sudo rclone delete <directorio backup nube> --min-age 15d
```

Finalmente, respecto a las copias de seguridad subidas a la nube, si deseamos añadir un nivel de seguridad extra, podríamos cifrar los backups con cualquier herramienta como LUKS o VeraCrypt antes de subir los archivos.

Con este último apartado, se da por finalizado el bloque de guía general de bastionado de sistemas basados en Raspberry Pi. Con ello, damos paso al siguiente bloque cuyo objetivo principal es enfocar el uso práctico de la guía en un entorno empresarial en el que la Raspberry Pi se utiliza como servidor web.

### 3. Bastionado de un entorno profesional

En el capítulo anterior se ha mostrado un enfoque o estrategia a partir de la cual bastionar una Raspberry Pi y, en algunas ocasiones, su entorno más próximo. De igual forma, se han expuesto distintas herramientas compatibles con la Raspberry Pi y su sistema operativo, cuyo objetivo principal es su protección.

Las herramientas y configuraciones mostradas en el apartado anterior pretenden cubrir la seguridad en general de la Raspberry Pi. El presente apartado, por el contrario, aspira a mostrar configuraciones y herramientas concretas para la securización de la Raspberry Pi usada como servidor web.

#### 3.1. Entorno profesional: servidor LEMP

Como se ha indicado, la idea es trabajar el bastionado de la Raspberry Pi sobre un entorno profesional, en el que la Raspberry Pi se encuentre corriendo un servidor web. En concreto, se ha decidido configurar el sistema para que corra los servicios y servidores necesarios para albergar una aplicación web: el propio sistema operativo, el servidor web, el sistema de gestión de bases de datos, y el lenguaje de programación del lado del servidor. La combinación de estos cuatro componentes es lo que se conoce como un servidor LAMP (Linux, Apache, MySQL y PHP), si utiliza como servidor web Apache2, o un servidor LEMP (Linux, Nginx, MySQL y PHP), en caso de utilizar Nginx como servidor web.

En nuestro caso, dado el creciente éxito de Nginx, que cada vez es más utilizado por las empresas, hemos configurado la Raspberry Pi para que haga las veces de servidor LEMP. En concreto, el software utilizado para el entorno es el siguiente:

Linux	<i>Raspberry Pi OS Lite (64-bit)</i>
Servidor web	<i>Nginx versión 1.18.0</i>
Sistema de base de datos	<i>MariaDB 10.5.19</i>
Lenguaje de programación	<i>PHP versión 8.2</i>

Con respecto a la elección del sistema de base de datos o servidor de base de datos, los más comunes en entornos LAMP/LEMP son MariaDB y MySQL. El hecho de utilizar MariaDB en lugar de MySQL atiende simplemente al hecho de

que MariaDB es de licencia GPL (*General Public License*), mientras que MySQL se encuentra bajo licencia dual (GPL y licencia comercial propiedad de Oracle Corporation). No obstante, ambas son muy parecidas, dado que MariaDB deriva de MySQL, y las configuraciones mostradas en los apartados siguientes son compatibles con ambos sistemas de gestión.

Por otro lado, referente al lenguaje de programación del lado del servidor, los más utilizados en entornos LAMP y LEMP son PHP, Perl y Python. En este caso, se ha optado por PHP por ser el lenguaje más utilizado en este tipo de entornos, aunque Python está ganando fuerza en estos últimos años.

Finalmente, se ha incluido una guía de configuración del servidor LEMP en la Raspberry Pi 4 (el mismo entorno utilizado para este capítulo) en el Anexo I de este mismo trabajo final.

### 3.2. Configuración segura de la Raspberry Pi

Teniendo configurado el entorno LEMP, es el momento de empezar a aplicar configuraciones de seguridad sobre éste. Más adelante entraremos en configuraciones específicas del servidor web, el sistema de bases de datos y el lenguaje de programación. Pero antes, debemos tomar algunas de las medidas que se han visto en el capítulo segundo sobre Seguridad de sistemas Raspberry Pi.

Respecto a qué apartados aplicar de la guía de bastionado de Raspberry Pi y su sistema operativo basado en Linux, es una decisión que recae en el administrador de sistemas (o el equipo de *Blue Team* si existe), y deberá adaptarse en todo momento a las necesidades y entorno del sistema. Para el caso concreto del servidor LAMP corriendo en una Raspberry Pi 4, a continuación se comentan aquellas configuraciones más importantes, mencionando en cada caso qué aspectos de la guía aplican.

#### 3.2.1. Seguridad física del servidor

En cuanto a la seguridad física del servidor, son de aplicación todos los apartados: configuración básica inicial; configuración del arranque para evitar el arranque desde USB; el bloqueo de los puertos USB y de los módulos Bluetooth y Wi-Fi; y el cierre automático de sesión.



Lo anterior es necesario para incrementar la seguridad, pero sin duda alguna, lo más importante en lo que a seguridad física del servidor se refiere es la protección de la microSD. Para ello, es necesario el uso de cajas que eviten el fácil acceso y extracción de la microSD, así como su cifrado para proteger los datos en caso de extracción, pérdida o robo.

### 3.2.2. Seguridad perimetral y de red

La configuración de seguridad perimetral más importante es la securización de protocolos utilizados para conectarnos con el servidor LEMP. En principio, los dos protocolos necesarios son SSH (para la administración del servidor) y FTP (para la transferencia de archivos). Por ello, es crucial realizar una configuración segura de ambos.

Respecto a la configuración segura de SSH, ya ha sido tratada en la guía, y es más que suficiente para poder administrar de forma segura un servidor LEMP. De todas formas, el administrador del servidor deberá analizar todas las opciones de configuración de *sshd\_config* y adaptarlas a los requerimientos de seguridad del servicio.

En lo que a FTP respecta, se desaconseja en absoluto su uso, dado que tanto la autenticación del usuario como la transferencia de archivos se realiza sin cifrado alguno. Dado que se presupone que utilizamos SSH configurado de forma segura (a ser posible, utilizando autenticación con clave pública), lo más cómodo es utilizar el protocolo SFTP en lugar de FTP, mediante el cual podremos transferir datos de forma segura utilizando la misma autenticación que configuramos en SSH [22]. Respecto a su utilización, funciona de una forma muy parecida a FTP.

Por otro lado, si lo que queremos es exponer nuestro servidor LEMP a Internet, es importante considerar varias cosas. Lo primero es que no sólo debemos proteger nuestro servidor. También debemos proteger la red interna y el resto de los *hosts*. Es obligatorio por tanto asilar nuestro servidor LEMP a una DMZ mediante un Firewall (el del propio router u otro dedicado). En la actualidad, la mayoría de routers cuentan con un Firewall (por ejemplo, iptables) mediante el cual podemos configurar reglas para que únicamente entre tráfico del exterior hacia los puertos HTTP/HTTPS o cualquier otro puerto que necesitemos exponer

a Internet. El aislamiento de nuestro servidor LEMP a la DMZ nos lleva a considerar otra cosa, y es que no suele ser una buena idea ubicar el sistema de base de datos en la DMZ, dado que, si alguien consiguiese tener acceso a nuestro servidor web, también tendría acceso a la base de datos. En los apartados siguientes se muestran configuraciones que ayudan a la protección de la base de datos en el caso anterior. No obstante, la configuración ideal sería ubicar el sistema de base de datos en una red distinta a la DMZ.

Finalmente, es muy recomendable aplicar las configuraciones de protección contra la suplantación de identidad, como tablas ARP estáticas, listas negras y otras mencionadas en el apartado 2.2.3.1 de la guía.

### 3.2.3. Seguridad del dispositivo

Enfocándonos en la protección a nivel de servidor, las dos medidas más necesarias son por un lado el uso de un ERP, HIDS, EDR o XDR (según convenga a las necesidades de seguridad deseadas) y, por otro lado, la correcta configuración del Firewall del dispositivo.

En este caso particular, se ha optado por el uso del antivirus ClamAV ya presentado en la guía de bastionado. Recordemos que ClamAV es un antivirus que realiza escaneos desde terminal, por lo que lo más efectivo es automatizar los escaneos de forma diaria o semanal. Un ejemplo de ello es realizar escaneos diarios de la raíz del servidor y automatizarlo en una tarea de crontab, de la siguiente forma:

```
# Escaneo de raíz y registro con ClamAV cada noche
1 1 * * * sudo clamscan -r --remove / -l /home/sergi/scan.txt --exclude-dir=/sys/
```

Figura 24. Ejemplo de escaneo automatizado con ClamAV y crontab

Dado que estamos trabajando sobre un servidor web, el uso en exclusiva de ClamAV como antivirus podría no ser suficiente. Para entornos de aplicaciones web, podemos utilizar el detector de malware de código abierto *Linux Malware Detect* o LMD (<https://www.rfxn.com/projects/linux-malware-detect/>) de forma integrada a ClamAV. La instalación de LMD puede realizarse desde el sitio web de proyectos de R-FX Networks:

```
$ sudo wget http://www.rfxn.com/downloads/maldetect-current.tar.gz
```

Una vez descargado en nuestra Raspberry Pi, descomprimos el archivo tar.gz:

```
$ tar -xvzf maldetect-current.tar.gz
```

Con lo anterior, nos habrá creado un directorio en el que se han movido los archivos descomprimidos. Nos movemos al directorio creado:

```
$ cd maldetect-1.6.5/
```

Y dentro del directorio ejecutamos el instalador como en otras ocasiones:

```
$ sudo ./install.sh
```

Con lo anterior ya tendríamos la herramienta instalada, por lo que podemos entrar en su configuración, accesible en el fichero */usr/local/maldetect/conf.maldet*. Existen muchas opciones que deberán ser investigadas por el administrador, pero las más importantes a configurar son:

- 1) *scan\_clam*: si su valor es “1”, LMD utilizará el binario de clamscan como motor de escaneo por defecto, lo que agilizará los escaneos del sistema.
- 2) *quarantine\_hits*: permite enviar a cuarentena los ficheros detectados como malware si su valor es “1”, para que puedan ser analizados por el administrador (en lugar de eliminarlos directamente). Con valor “0” LMD eliminará automáticamente el malware detectado.
- 3) *quarantine\_clean*: requiere que *quarantine\_hits* esté activada, y borra automáticamente los archivos que se encuentren en cuarentena si su valor es “1”, por lo que si deseamos analizar el malware en cuarentena, deberemos aportarle un valor de “0”.
- 4) *scan\_user\_access*: si su valor es “0”, sólo se podrán realizar acciones de escaneo, actualización o revisión de reportes con permisos de superusuario.

Un factor importante respecto a los antivirus o antimalware es que, para que su efectividad sea la máxima posible, debemos mantenerlos actualizados en lo que a la base de datos respecta.

Para actualizar la base de datos de LMD, basta con ejecutar el siguiente comando:

```
$ sudo maldet -u
```

Dado que es una tarea que debe realizarse de forma asidua, podría ser interesante automatizarla cada cierto tiempo mediante crontab, tal y como se ha visto en otros ejemplos.

Los escaneos con LMD funcionan de forma muy parecida a como lo hace ClamAV, dado que se realizan a través de comandos. Un ejemplo de escaneo con LMD es el siguiente:

```
$ sudo maldet -a /home/sergi
Linux Malware Detect v1.6.5
      (C) 2002-2023, R-fx Networks <proj@rfxn.com>
      (C) 2023, Ryan MacDonald <ryan@rfxn.com>
This program may be freely redistributed under the terms of the GNU
GPL v2

maldet(2133): {scan} signatures loaded: 17637 (14801 MD5 | 2053 HEX |
783 YARA | 0 USER)
maldet(2133): {scan} building file list for /home/sergi, this might
take awhile...
maldet(2133): {scan} setting nice scheduler priorities for all
operations: cpunice 19 , ionice 6
maldet(2133): {scan} file list completed in 0s, found 52 files...
maldet(2133): {scan} found clamav binary at /usr/bin/clamscan, using
clamav scanner engine...
maldet(2133): {scan} scan of /home/sergi (52 files) in progress...
maldet(2133): {scan} processing scan results for hits: 2 hits 0
cleaned
maldet(2133): {scan} scan completed on /home/sergi: files 52, malware
hits 2, cleaned hits 0, time 3s
maldet(2133): {scan} scan report saved, to view run: maldet --report
230519-1523.2133
```

Como podemos ver en la última línea, el resultado del escaneo puede consultarse con el comando:

```
$ sudo maldet --report 230519-1523.2133
```

```
GNU nano 5.4 /usr/local/maldet/scan/session.230519-1523.2133 *
HOST: raspberrypi
SCAN ID: 230519-1523.2133
STARTED: May 19 2023 15:23:02 +0200
COMPLETED: May 19 2023 15:23:05 +0200
ELAPSED: 3s [find: 0s]

PATH: /home/sergi
TOTAL FILES: 52
TOTAL HITS: 2
TOTAL CLEANED: 0

FILE HIT LIST:
{HEX}php.gzbase64.inject.456 : /tmp/maldet-1.6.5/files/clean/gzbase64.inject.unclassified => /usr/local/maldet/quarantined/gzbase64.inject.456
{HEX}php.gzbase64.inject.456 : /tmp/maldet-1.6.5/files/signs/rfxn.yara => /usr/local/maldet/quarantined/rfxn.yara
=====
Linux Malware Detect v1.6.5 < proj@rfxn.com >
```

Figura 25. Resultado de un escaneo con LMD

Finalmente, para automatizar el escaneo de malware con LMD, deberemos hacerlo con crontab, tal y como se ha indicado anteriormente en la automatización de ClamAV:

```
$ sudo crontab -e
```

Y añadimos la tarea:

```
# Escaneo de raíz y registro con LMD cada día a las 01:00
0 1 * * * sudo maldet -a /
```

Figura 26. Ejemplo de escaneo automatizado con LMD y crontab

Pasando ahora a la configuración del Firewall de la Raspberry Pi, lo más cómodo y efectivo es el uso de UFW en combinación con Fail2ban, tal y como se muestra en los apartados 2.2.2.1 y 2.2.2.2 de la guía. La política por defecto de UFW es bloquear todo el tráfico entrante que no haya sido requerido, y permitir todo el tráfico saliente. Deberemos aplicar por tanto únicamente las reglas que nos permitan administrar el servidor y que permitan a éste ofrecer los servicios a Internet. Respecto a las reglas de UFW para el servidor LEMP, un ejemplo de configuración podría ser el siguiente:

- 1) Limitar el uso del puerto utilizado para SSH y SFTP, a poder ser distinto al puerto 22 (por ejemplo, el puerto 4564):

```
$ sudo ufw limit 4564
```

Aunque quizás podría ser interesante permitir el uso de tales puertos o protocolos únicamente a direcciones IP públicas o privadas conocidas:

```
$ sudo ufw allow from 192.168.1.55 to any port 4564
$ sudo ufw allow from 79.156.177.150 to any port 4564
```

- 2) Permitir el uso del puerto 443, utilizado por defecto para conexiones HTTPS:

```
$ sudo ufw allow 443
```

- 3) Permitir el uso del puerto 80 o el uso del protocolo HTTP:

```
$ sudo ufw allow 80
$ sudo ufw allow HTTP
```

Respecto a Fail2ban, se debe tener especial atención a la correcta configuración de las *jails* de SSH, SFTP, HTTP y HTTPS para servidores Nginx, que puede realizarse mediante la edición del archivo `/etc/fail2ban/jail.conf`, tal y como recoge el apartado 2.2.2.2 de la guía.

#### 3.2.4. Seguridad de la capa de aplicación

Los aspectos más importantes de seguridad de la capa de aplicación para un servidor LEMP son: la gestión de los permisos y atributos, adaptando las capacidades de los usuarios a las mínimas indispensables; la administración de identidades y accesos, que deberán estar actualizadas; y la configuración de un cortafuegos de aplicación web o WAF compatible con el servidor web, como por ejemplo ModSecurity.

En este apartado no entraremos en detalle con la configuración del WAF ModSecurity para el servidor Nginx, ya que ello será tratado en el apartado 3.3.4 sobre configuración segura de Nginx con ModSecurity.

#### 3.2.5. Seguridad de los datos

Tanto si nuestro servidor LEMP se utiliza para mostrar aplicaciones o sitios web únicamente para la organización o si el cometido es prestar servicios a clientes externos, debemos asegurar los datos y la información del sistema mediante copias de seguridad.

En la guía se han expuesto dos herramientas (raspiBackup y Rclone) que, utilizadas de forma combinada, nos permiten aplicar una política de backups de forma sencilla, automatizada, y con redundancia en la nube. La instalación de raspiBackup y Rclone ya han sido abordadas en la guía, por lo que en este

apartado se mostrará un ejemplo de implementación de una política de backups en el servidor LEMP.

Para poder realizar backups del sistema con raspiBackup, primero necesitaremos una unidad de memoria igual o mayor a la que alberga el sistema. En nuestro caso, haremos uso de una unidad de estado sólido conectada mediante USB 3.0, y que cuenta con 500 GB de almacenamiento. En nuestro caso, se ha creado una partición `/dev/sda1` en EXT4 y montado en el directorio `/mnt/backup`, tal y como se muestra a continuación:

```
$ sudo lsblk
[sudo] password for sergi:
NAME            MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda              8:0    0 465.8G  0 disk
└─sda1           8:1    0 465.7G  0 part /mnt/backup
mmcblk0         179:0    0  59.7G  0 disk
├─mmcblk0p1     179:1    0   256M  0 part /boot
└─mmcblk0p2     179:2    0   59.4G  0 part
   └─sdcard      254:0    0   59.4G  0 crypt /
```

Con la unidad de memoria lista y montada de forma persistente en el sistema, pasamos ahora a diseñar la política de backups en raspiBackup. Podremos hacerlo mediante su instalación o, una vez instalada, con el siguiente comando:

```
$ sudo raspiBackupInstallUI
```

Una vez nos encontremos en la herramienta de configuración de raspiBackup, entramos en *“Configure major options”*. Una vez dentro, deberemos ir accediendo a cada sección para configurar la política de backup. Si se desea conocer que hace cada sección, éstas han sido expuestas en el apartado 2.2.6.3. Un ejemplo de política (utilizada para este trabajo) podría ser:

Sección	Opción
<i>Backup path</i>	<i>/mnt/backup</i>
<i>Backup versions</i>	<i>Keep a maximum number of backups (5)</i>
<i>Backup type</i>	<i>Backup with tar</i>
<i>Backup mode</i>	<i>Backup the two standard partitions</i>

<i>Services to stop and start</i>	<i>clamav-daemon ; clamav-freshclam; cron; fail2ban; mariadb; ModemManager; nginx; php8.2-fpm; polkit</i>
<i>Message verbosity</i>	<i>Display important messages only</i>
<i>Regular backup</i>	<i>Enable regular backup</i> <i>Weekday of regular backup: Daily</i> <i>Time of regular backup: 03:00</i>
<i>Compression</i>	<i>Compress tar backup</i>

Con lo anterior, y tras haber guardado y actualizado los cambios al salir de la herramienta, raspiBackup realizará las copias de seguridad de forma automática siguiendo la política establecida.

El siguiente paso será preparar Rclone para que nos copie los backups periódicos a un almacenamiento en la nube. Lo primero que debemos hacer es crear un nuevo “*remote*”, que es el nombre que Rclone utiliza para referirse a los almacenamientos en la nube del usuario (o el perfil para la interacción de Rclone con la API del servicio cloud). Accedemos al asistente de configuración de Rclone:

```
$ sudo rclone config
```

Nos aparecerá un menú con distintas opciones. Para crear un nuevo *remote*, basta con pulsar la tecla “n”. A continuación, la herramienta nos guiará por distintos pasos: nombre para el *remote*; tipo de almacenamiento en la nube; *client\_id*; *client\_secret*; nivel de acceso a la nube; y configuraciones avanzadas.

Aquí, la idea es que aportemos al *remote* la información necesaria para que pueda interactuar con la API del proveedor de nube, y pueda así subir los backups y gestionarlos.

Una vez configurado el *remote*, nos aparecerá en el menú de configuración de Rclone:



```
Current remotes:

Name                Type
====                ====
gdrive              drive

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q>
```

Como ejemplo, se ha creado un *remote* llamado “*gdrive*” y cuyo propietario es *root*, configurado para tener acceso completo a la nube de Google Drive de un usuario concreto, que será utilizado para guardar los backups.

Ahora que ya tenemos el *remote* creado, podemos usar Rclone para subir archivos mediante comandos por consola. Sin embargo, tal y como se ha expuesto anteriormente, lo ideal es automatizar la tarea con crontab. Para el ejemplo anterior, se ha incluido la siguiente línea en el crontab de *root*:

```
# Subir backup de raspiBackup a Google Drive cada día a las 05:00
0 5 * * * rclone copyto /mnt/backup/raspberrypi gdrive:
```

Figura 27. Subida de backup a Google Drive con Rclone

Como podemos ver en la siguiente figura, efectivamente, raspiBackup ha realizado una copia de seguridad del sistema, guardada en */mnt/backup/raspberrypi*, y Rclone la ha subido a Google Drive a las 05:00 AM:

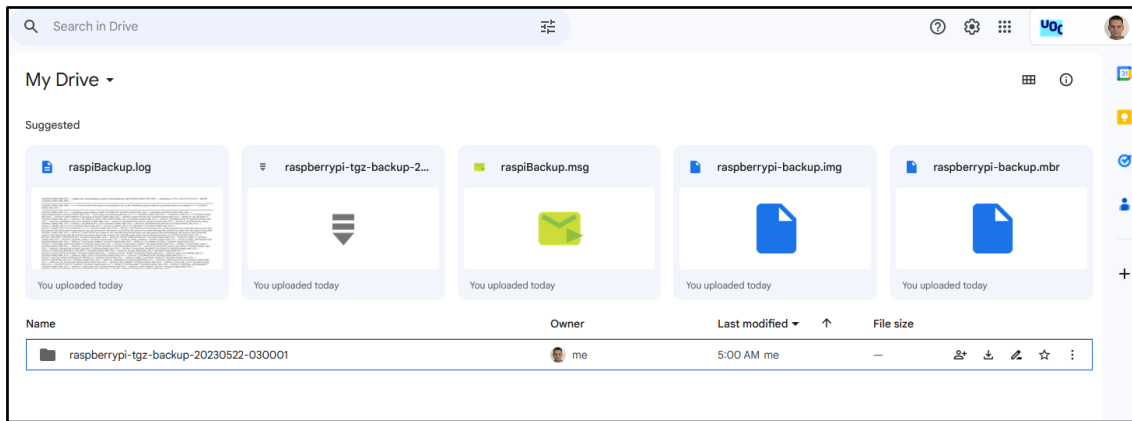


Figura 28. Redundancia de backups en Google Drive

### 3.3. Bastionado de Nginx

Es momento de hablar de la configuración segura de nuestro servidor web Nginx. Para ello, trataremos varios aspectos importantes, siendo el primero de ellos la securización general del servidor. Complementarios a éste, se trabajarán también aspectos más concretos, como el uso del protocolo HTTPS o la implementación de un WAF.

#### 3.3.1. Aspectos generales del servidor

Uno de los aspectos a configurar al momento de desplegar un nuevo servidor de Nginx son todas aquellas opciones que afectan de forma general a todos los *Server Blocks*.

En los siguientes apartados se recogen los aspectos generales más importantes en lo que a configuración de seguridad de Nginx se refiere.

##### 3.3.1.1. Desactivar módulos de Nginx no utilizados

De cara a tener un servidor Nginx en producción, es importante que éste únicamente tenga activos los mínimos módulos indispensables para su función. De esta forma, estaremos reduciendo la superficie susceptible de recibir ataques del exterior.

En servidores Apache, la desactivación de módulos puede hacerse una vez instalado el servidor, mediante el uso de *a2dismod* [6, p. 223]. Sin embargo, en servidores Nginx la única posibilidad de hacerlo es antes de la instalación y despliegue del servidor. Es mediante la utilización de la opción “*configure*”, antes de lanzar el comando de instalación, con la cual podremos desactivar los

módulos que no utilizemos. Necesitaremos descargar el código fuente de Nginx a fin de recompilarlo:

```
$ wget https://nginx.org/download/nginx-1.23.4.tar.gz
$ tar zxf nginx-1.23.4.tar.gz
$ cd nginx-1.23.4
```

Una vez dentro del directorio, hacemos uso de la opción *configure* para poder deshabilitar los módulos que le indiquemos. Por ejemplo:

```
$ ./configure --without-http_proxy_module
```

Y realizamos la instalación con la nueva configuración de Nginx:

```
$ make
$ make install
```

Si deseamos ver qué módulos podemos desactivar, basta con ejecutar el siguiente comando:

```
$ ./configure --help | grep without
```

### 3.3.1.2. Configuración de *server blocks*

En servidores Nginx instalados en Raspberry Pi OS, por defecto, el directorio de configuración del servidor es */etc/nginx/*, siendo el archivo principal */etc/nginx/nginx.conf*. Es en este archivo en el que deberemos realizar la configuración de, entre otras cosas, los *server blocks*.

Nginx trae configurado por defecto *nginx.conf* para que sea utilizable desde el momento en que instalamos el servidor. Sin embargo, lo más recomendable es modificar el archivo para realizar nuestra propia configuración.

El contexto principal del fichero *nginx.conf* cuenta con la directiva *user*, que indica el usuario que tendrá acceso y capacidad de modificar directorios y archivos relacionados con la aplicación o los sitios web. Se recomienda que dicho usuario sea único y exclusivo para Nginx, que por defecto es *www-data*.

Respecto a la configuración de los *server blocks*, se recomienda incluir los *virtual hosts* que deseemos tener en producción tal y como lo hace por defecto Nginx. Es decir, configurando cada uno de los *server blocks (virtual hosts)* en el directorio */etc/nginx/sites-available* (cada uno con su fichero), añadiendo un enlace simbólico de los *virtual host* que deseemos pasar a producción en

`/etc/nginx/sites-enabled`, y añadirlos a `nginx.conf` mediante la opción `"include /etc/nginx/sites-enabled/*"`.

### 3.3.1.3. Eliminar información del servidor

La configuración por defecto de Nginx hace que, en caso de devolver un error a una petición, la respuesta HTTP vaya acompañada de información sobre el tipo de servidor y su versión en el encabezado del servidor. Por ejemplo, si escribimos en el navegador una URL del servidor que apunta a una ruta que no existe, la respuesta del servidor será algo como la siguiente:

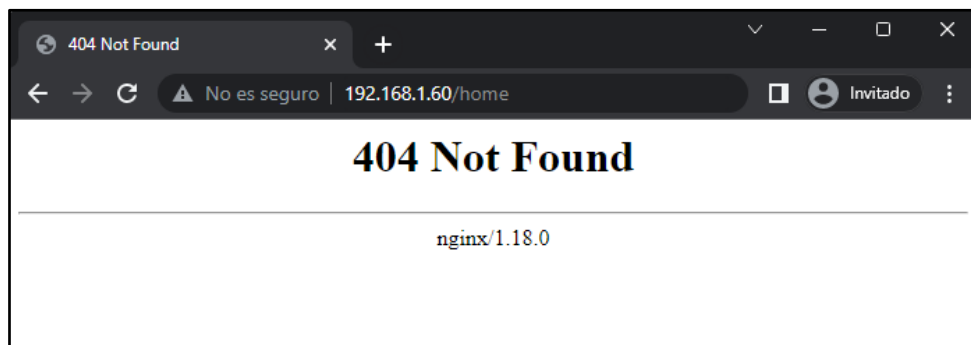


Figura 29. Versión del servidor Nginx expuesta

Lo anterior es producido por una directiva de Nginx llamada `server_tokens` que, si bien en entornos de prueba puede ser útil, nunca debe estar activa en entornos en producción. Para deshabilitar la directiva, basta con añadir la directiva desactivada en las opciones básicas de `/etc/nginx/nginx.conf`, tal y como se muestra a continuación:

```
http {  
  
    ##  
    # Basic Settings  
    ##  
  
    sendfile on;  
    tcp_nopush on;  
    types_hash_max_size 2048;  
    server_tokens off;  
  
    # server_names_hash_bucket_size 64;  
    # server_name_in_redirect off;  
  
    include /etc/nginx/mime.types;  
    default_type application/octet-stream;
```

Figura 30. Directiva `server_tokens` deshabilitada

Al parecer, en Raspberry Pi OS el servidor Nginx (por defecto) no muestra el sistema operativo en las respuestas de error (como si lo hace, por ejemplo, en Ubuntu). Esto se llama *Server Signature*, y para deshabilitarlo basta con incluir la directiva `proxy_hide_header` en `/etc/nginx/nginx.conf`:

```
proxy_hide_header X-Powered-By;
```

Figura 31. Directiva para deshabilitar Nginx Server Signature

#### 3.3.1.4. Cabeceras de seguridad

El uso de cabeceras de seguridad nos permite fortificar nuestro servidor Nginx para evitar ataques más concretos y forzar la utilización de protocolos más seguros [23].

Mediante la configuración básica de `nginx.conf` podemos evitar los ataques de tipo *Clickjacking*. Éstos se basan en engañar al usuario víctima para que haga click en enlaces maliciosos (puestos por el atacante) sin saberlo. El *clickjacking* se puede evitar utilizando la cabecera *X-Frame-Options*.

Si nuestro servidor cuenta con un certificado SSL o TLS, y por lo tanto es capaz de usar el protocolo HTTPS, se puede forzar la utilización del protocolo seguro añadiendo la cabecera *Strict-Transport-Security*.

Por último, podemos combatir los ataques de *Cross-site Scripting* (XSS) y otros ataques de inyección de código con la cabecera *Content-Security-Policy* y *X-XSS-Protection*.

Las cabeceras de seguridad expuestas en este apartado se deben añadir todas a `/etc/nginx/nginx.conf`. A continuación se muestra un ejemplo de implementación de todas ellas:

```
##  
# Security Headers  
##  
  
add_header X-Frame-Options SAMEORIGIN;  
add_header Strict-Transport-Security "max-age=31536000; includeSubdomains; preload";  
add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline'" always;  
add_header X-XSS-Protection "1; mode=block";
```

Figura 32. Cabeceras de seguridad

### 3.3.1.5. Limitación de los métodos HTTP

El protocolo HTTP cuenta con varios métodos de petición utilizados para pedir una acción concreta al servidor. Muchos de ellos, como PUT, DELETE o TRACE son muy útiles durante el desarrollo de la aplicación web, pero nunca deberían estar disponibles para su uso en producción. Para aplicaciones web en producción, lo ideal es limitar el uso de métodos HTTP a los métodos GET, HEAD y POST, y denegar el uso del resto de métodos.

En servidores Nginx, podemos limitar el uso de métodos HTTP en el bloque *location* de los archivos de configuración de los *virtual hosts* (en los archivos de configuración de los *server blocks*, ubicados en `/etc/nginx/sites-available`) utilizando *limit\_except*. Por ejemplo, para el *server block default*:

```
$ sudo nano /etc/nginx/sites-available/default
```

```
location / {  
    limit_except GET HEAD POST { deny all; }  
}
```

Figura 33. Limitación de métodos HTTP

### 3.3.1.6. Protección contra DoS y Buffer Overflow

Los ataques de denegación de servicio (DoS) y los de denegación de servicio distribuidos (DDoS) suelen ser ataques muy comunes en el ámbito de los servidores web, y un gran dolor de cabeza cuando se producen.

En la configuración de *nginx.conf*, contamos con varias opciones que nos ayudan a poner límites en el tamaño del búfer de los usuarios, a fin de evitar el colapso del servidor por posibles ataques de denegación de servicio e incluso de *Buffer Overflow*. A continuación, se muestran las directivas que deberemos añadir, así como un ejemplo de su configuración [23]:

```
##  
# Client Size Limitations  
##  
  
client_body_buffer_size 1k;  
client_header_buffer_size 1k;  
client_max_body_size 1k;  
large_client_header_buffers 2 1k;
```

Figura 34. Directivas para limitar el tamaño del búfer

Relacionado con lo anterior, y sobre todo con la mejora del rendimiento del servidor, podemos añadir también a *nginx.conf* directivas para controlar los *Timeouts* de los usuarios [24]:

```
##
# Client Timeouts
##

client_body_timeout    10;
client_header_timeout  10;
keepalive_timeout      5 5;
send_timeout           10;
```

Figura 35. Directivas para el control de *Timeouts*

### 3.3.1.7. Registros de acceso y errores del servidor

El registro de accesos y *logs* de nuestro servidor Nginx ya viene preconfigurado en *nginx.conf*, concretamente en las directivas *access\_log* y *error\_log*. Por defecto, los *logs* de acceso se guardarán en */var/log/nginx/access.log*, y los *logs* de errores se guardarán en */var/log/nginx/error.log*. No obstante, pueden cambiarse según las preferencias del administrador.

En referencia a los registros de errores, podemos indicar en la directiva *error\_log* que se registren todos los mensajes de error, o registrar únicamente aquellos que deseemos. En Nginx existen los siguientes niveles de severidad de errores de *logging*, en orden ascendente: *debug*; *info*; *notice*; *warn*; *error*; *crit*; *alert*; y *emerg*.

Si deseamos que únicamente se registren los errores de *logging* de severidad superior al nivel *warn*, basta con indicarlo en la directiva:

```
##
# Logging Settings
##

access_log /var/log/nginx/access.log;
error_log /var/log/nginx/error.log warn;
```

Figura 36. Registro de errores iguales o superiores a *warn*

Por otro lado, también podemos modificar el formato en el que el servidor registra los accesos. Para ello, podemos utilizar la directiva *log\_format* para indicar valores como el tiempo para establecer conexión con otro servidor que nos

preste servicios (*\$upstream\_connect\_time*), el tiempo desde el establecimiento de la conexión hasta que se recibe el primer byte de respuesta (*\$upstream\_header\_time*), el tiempo entre el establecimiento de la conexión y el último byte (*\$upstream\_response\_time*), o el tiempo del proceso de petición completo (*\$request\_time*). Para más información, puede consultarse la documentación oficial de Nginx en el enlace: <https://docs.nginx.com/nginx/admin-guide/monitoring/logging/>.

Finalmente, podría ser interesante monitorizar el registro de accesos y errores del servidor implementando cualquier herramienta de gestión de *logs*. Una opción muy utilizada es Elastic (ELK) Stack, mencionada en el apartado 2.2.3.3 de la guía.

#### 3.3.1.8. Otras recomendaciones

Terminaremos con los aspectos generales que podemos atender para fortificar nuestro servidor con algunas recomendaciones.

La primera recomendación es que, tal y como ocurre con cualquier otro sistema, la actualización regular del servidor es una de las tareas básicas y que siempre deben existir. Nginx hace un gran trabajo publicando en su sitio web [https://nginx.org/en/security\\_advisories.html](https://nginx.org/en/security_advisories.html) todos avisos de todas las vulnerabilidades conocidas, en los que indican la severidad de la vulnerabilidad, las versiones vulnerables y los parches que la solucionan.

La segunda recomendación es el uso de herramientas de análisis estático de la configuración de nuestro servidor, con el objetivo de ayudarnos a encontrar posibles fallos o configuraciones erróneas y que pueden suponer un problema para la seguridad. Una herramienta de análisis estático para Nginx enfocada en encontrar configuraciones erróneas de seguridad es Gixy (<https://github.com/yandex/gixy>), aunque por desgracia no se incluye en los repositorios de Raspberry Pi OS. Por lo que si deseamos utilizar la herramienta, deberemos incluir PyPI (repositorio en el que se distribuye Gixy) al listado de repositorios de APT.



### 3.3.2. Protocolo HTTPS

Una de las tareas más esenciales en lo que a seguridad del servidor (y de los usuarios) se refiere es la implementación de HTTPS. Para que nuestro servidor pueda utilizar HTTPS en lugar de HTTP, debemos obtener un certificado SSL/TLS. En este caso práctico, haremos uso de Let's Encrypt (para obtener el certificado digital) y del servicio de DNS Dinámico No-IP (mediante el cual hemos creado un DDNS vinculado a la IP pública de la red en la que se encuentra el servidor LEMP) para la configuración de HTTPS en nuestro servidor Nginx.

Previo a conseguir el certificado SSL/TLS, debemos asegurarnos de que la configuración del *server block* para el que vamos a pedir el certificado está debidamente configurado. En este sentido, es importante que el archivo de configuración del *server block* (ubicado en */etc/nginx/sites-available*) contenga en la directiva *server\_name* el dominio en cuestión. En nuestro caso concreto, el dominio de No-IP es *tfmpi.ddns.net*, por lo que la configuración del *server block* es la siguiente:

```
server_name tfmpi.ddns.net;
```

*Figura 37. Dominio configurado en server\_name*

Una vez lo tengamos listo, verificamos que no hay ningún problema con la configuración de Nginx y reiniciamos el servicio:

```
$ sudo nginx -t  
$ sudo systemctl reload nginx
```

Por otro lado, es importante confirmar que la configuración del Firewall del servidor permite el tráfico HTTPS o el tráfico del puerto 443.

Ya podemos pasar a obtener el certificado SSL/TLS con Let's Encrypt. Para ello, utilizaremos una herramienta que automatiza y facilita la obtención de certificados Let's Encrypt, llamada "certbot" (<https://certbot.eff.org/>). De hecho, certbot no sólo nos ayuda a obtener de forma muy sencilla un certificado para dominios concretos, sino que además se ocupará de renovarlos cuando sea necesario (los certificados de Let's Encrypt son válidos por tres meses) de forma automática.

Certbot se puede instalar directamente mediante APT en Raspberry Pi OS, por lo que ejecutamos el siguiente comando:

```
$ sudo apt install certbot python3-certbot-nginx
```

Una vez instalado, basta con ejecutar el comando “*certbot --nginx*” para comenzar el proceso de petición de certificado digital. Si es la primera vez que ejecutamos certbot, éste nos pedirá un correo electrónico al cual nos enviará alertas sobre las renovaciones. También nos pedirá aceptar los términos y condiciones.

Durante el proceso, certbot detectará automáticamente los dominios asociados a los *server block* que tengamos, y nos preguntará sobre cual deseamos obtener un certificado. Una vez indicado, la herramienta se comunicará con los servidores de Let’s Encrypt con tal de realizar un desafío mediante el cual verificará que, efectivamente, contamos con el dominio indicado.

A continuación se muestra un ejemplo del proceso para el dominio *tfmpi.ddns.net*:

```
$ sudo certbot --nginx
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator nginx, Installer nginx

Which names would you like to activate HTTPS for?
- - - - -
- - - - -
1: tfmpi.ddns.net
- - - - -
- - - - -

Select the appropriate numbers separated by commas and/or spaces, or
leave input

blank to select all options shown (Enter 'c' to cancel): 1
Requesting a certificate for tfmpi.ddns.net
Deploying Certificate to VirtualHost /etc/nginx/sites-
enabled/tfmpi.ddns.net
```

```
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/tfmpi.ddns.net

-----

Congratulations! You have successfully enabled https://tfmpi.ddns.net

-----
```

Certbot guardará el certificado, su cadena y la clave privada en `/etc/letsencrypt/live/`. En el ejemplo mostrado, la cadena del certificado la ha guardado en `/etc/letsencrypt/live/tfmpi.ddns.net/fullchain.pem` y la clave privada en `/etc/letsencrypt/live/tfmpi.ddns.net/privkey.pem`.

Si entramos con el navegador web a nuestro sitio web, veremos como ahora cuenta con un certificado válido y utiliza HTTPS:

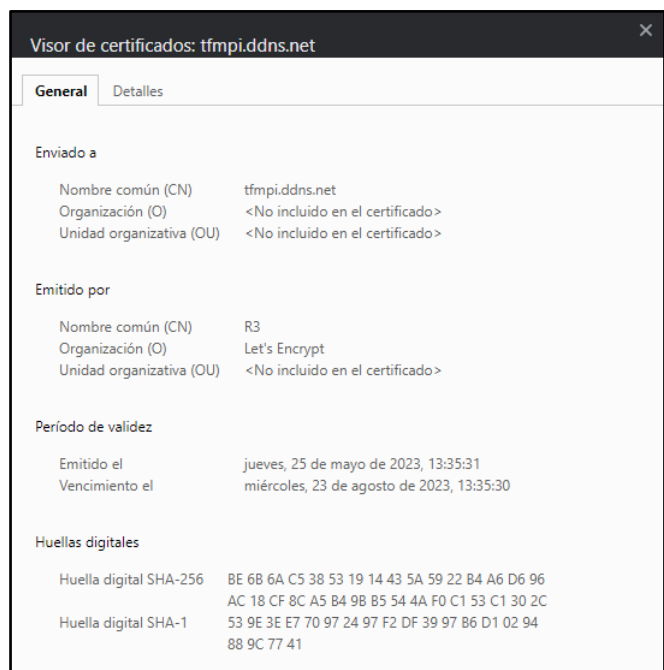


Figura 38. Certificado SSL/TLS válido

Lo anterior, junto con la configuración realizada en el apartado 3.3.1.4 para forzar el uso de HTTPS (añadiendo la cabecera `Strict-Transport-Security` en la configuración de Nginx), hará que en todo momento el usuario navegue en nuestro servidor con el protocolo HTTPS en lugar de HTTP.

Finalmente, para una configuración más segura de HTTPS, podemos modificar dos directivas: `ssl_protocols` para forzar el uso de las últimas versiones de TLS;

y `ssl_ciphers` y `ssl_prefer_server_ciphers` para que Nginx utilice conjuntos de cifrado modernos. Todo ello puede hacerse modificando el apartado “SSL Settings” de `nginx.conf`. Por ejemplo:

```
##
# SSL Settings
##
ssl_protocols TLSv1.2 TLSv1.3;
ssl_ciphers 'ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:
ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384';
ssl_prefer_server_ciphers on;
```

Figura 39. SSL Settings en `nginx.conf`

### 3.3.3. Web Application Firewall (WAF)

Tal y como se ha tratado en el apartado 2.2.5.3 de esta misma guía, añadir un nivel extra de protección a nivel de aplicación con un *Webb Application Firewall* es algo extremadamente recomendable en sistemas Raspberry Pi que corren servidores web.

En este caso utilizaremos el WAF ModSecurity v3. Si bien nació como un módulo de Apache, el proyecto de ModSecurity ha ido creciendo a lo largo del tiempo, hasta el punto en el que se ha desvinculado totalmente de Apache, dando soporte a otros servidores web como Nginx.

Teniendo instalado Nginx, la implementación de ModSecurity se debe realizar añadiendo como módulo el WAF a Nginx, por lo que deberemos montar el módulo de ModSecurity a partir de una compilación de la misma versión de Nginx que tenemos instalado, para posteriormente añadirlo al Nginx instalado e incluirle el conjunto de reglas del cortafuegos. A continuación se desarrollan los pasos a seguir [25].

#### Descarga y compilación de ModSecurity:

Primero de todo, necesitaremos instalar las dependencias necesarias para poder realizar los distintos procesos de compilación:

```
$ sudo apt-get install bison build-essential ca-certificates curl dh-
autoreconf doxygen flex gawk git iputils-ping libcurl4-gnutls-dev
libexpat1-dev libgeopip-dev liblmbd-dev libpcre3-dev libpcre++-dev
libssl-dev libtool libxml2 libxml2-dev libyajl-dev locales lua5.3-dev
pkg-config wget zlib1g-dev libgd-dev
```

Entre otras cosas, habremos instalado git, mediante el cual clonaremos el repositorio de ModSecurity de Github (se recomienda hacer en el directorio */opt*):

```
$ cd /opt
$ sudo git clone https://github.com/SpiderLabs/ModSecurity
```

Una vez descargado, nos movemos al directorio del repositorio clonado y inicializamos y actualizamos el submódulo de Git:

```
$ cd ModSecurity
$ sudo git submodule init
$ sudo git submodule update
```

Lanzamos el script *build.sh* y también el ejecutamos el fichero *configure*, a fin de obtener todas las dependencias necesarias para el proceso de montaje:

```
$ sudo ./build.sh
$ sudo ./configure
```

Una vez hecho, montamos ModSecurity:

```
$ sudo make
$ sudo make install
```

### Descarga del conector de ModSecurity para Nginx:

Antes de proceder a compilar el módulo de ModSecurity para Nginx, necesitamos clonar el repositorio del conector de ModSecurity para Nginx (<https://github.com/SpiderLabs/ModSecurity-nginx>). Éste es el punto de conexión entre Nginx y ModSecurity v3, requerido para usar ModSecurity con Nginx.

Basta con clonar con git el repositorio en cuestión (de nuevo, se recomienda hacerlo en */opt*):

```
$ cd /opt
$ sudo git clone --depth 1 https://github.com/SpiderLabs/ModSecurity-nginx.git
```

### Montaje del módulo de ModSecurity:

Con el conector de Nginx y ModSecurity instalado, ya podemos montar el módulo de ModSecurity a partir de una copia de la misma versión de Nginx que tenemos instalado.

Primero, descargamos la copia de Nginx en `/opt`, que necesariamente debe ser la misma versión que el Nginx instalado. Para conocer la versión, basta con ejecutar:

```
$ nginx -v
nginx version: nginx/1.18.0
```

En nuestro caso, la versión instalada es la 1.18.0, por lo que descargamos una copia de esta misma versión mediante:

```
$ cd /opt
$ sudo wget http://nginx.org/download/nginx-1.18.0.tar.gz
$ sudo tar -xvzmf nginx-1.18.0.tar.gz
$ cd nginx-1.18.0
```

Debemos conocer los argumentos de configuración para nuestra versión de Nginx con tal de compilar el módulo de ModSecurity. Para que se muestren por Terminal, ejecutamos (nótese que en este caso, la uve es mayúscula):

```
$ nginx -V
nginx version: nginx/1.18.0
built with OpenSSL 1.1.1n 15 Mar 2022
TLS SNI support enabled

configure arguments: --with-cc-opt='-g -O2 -ffile-prefix-
map=/build/nginx-I6LWFq/nginx-1.18.0=. -fstack-protector-strong -
Wformat -Werror=format-security -fPIC -Wdate-time -D_FORTIFY_SOURCE=2'
--with-ld-opt='-Wl,-z,relro -Wl,-z,now -fPIC' --
prefix=/usr/share/nginx --conf-path=/etc/nginx/nginx.conf --http-log-
path=/var/log/nginx/access.log --error-log-
path=/var/log/nginx/error.log --lock-path=/var/lock/nginx.lock --pid-
path=/run/nginx.pid --modules-path=/usr/lib/nginx/modules --http-
client-body-temp-path=/var/lib/nginx/body --http-fastcgi-temp-
path=/var/lib/nginx/fastcgi --http-proxy-temp-
path=/var/lib/nginx/proxy --http-scgi-temp-path=/var/lib/nginx/scgi --
http-uwsgi-temp-path=/var/lib/nginx/uwsgi --with-compat --with-debug -
-with-pcre-jit --with-http_ssl_module --with-http_stub_status_module -
-with-http_realip_module --with-http_auth_request_module --with-
http_v2_module --with-http_dav_module --with-http_slice_module --with-
threads --with-http_addition_module --with-http_gunzip_module --with-
http_gzip_static_module --with-http_sub_module
```

Copiamos todos los *configure arguments* y los pegamos en el comando de compilación, cambiándolos por `<configure_arguments>` en el siguiente comando:

```
$ sudo ./configure <configure_arguments> --ad-dynamic-  
module=../ModSecurity-nginx
```

Habiendo compilado el módulo de ModSecurity en Nginx, montamos los módulos con el siguiente comando:

```
$ sudo make modules
```

Ya tendremos listo el módulo de ModSecurity para incluirlo en nuestro Nginx instalado. Para ello, creamos un nuevo directorio de módulos en */etc/nginx*, y copiamos el módulo recién compilado a este directorio:

```
$ sudo mkdir /etc/nginx/modules
```

```
$ sudo cp objs/nginx_http_modsecurity_module.so /etc/nginx/modules
```

Ahora, sólo nos queda cargar el módulo desde la configuración de Nginx en */etc/nginx/nginx.conf* añadiendo la directiva *load\_module* apuntando al módulo copiado en el paso anterior:

```
$ sudo nano /etc/nginx/nginx.conf
```

```
user www-data;  
worker_processes auto;  
pid /run/nginx.pid;  
include /etc/nginx/modules-enabled/*.conf;  
load_module /etc/nginx/modules/nginx_http_modsecurity_module.so;
```

Figura 40. Carga del módulo ModSecurity en *nginx.conf*

Ya tenemos listo y cargado el módulo de ModSecurity en nuestro Nginx.

### Configuración de ModSecurity y Nginx:

ModSecurity se incluye con un conjunto de reglas básicas para la detección y bloqueo de ataques llamado *ModSecurity-CRS*. Sin embargo, puede llegar a ser demasiado básico. Existe un conjunto de reglas para ModSecurity, llamado *OWASP ModSecurity Core Rule Set* (<https://github.com/coreruleset/coreruleset>), cuyo objetivo es detectar y bloquear un amplio abanico de ataques a servidores web, incluyendo los *OWASP Top Ten* (<https://owasp.org/www-project-top-ten/>).

Para añadir el *OWASP ModSecurity CRS*, debemos eliminar primero el *CRS* que incluye ModSecurity por defecto:

```
$ sudo rm -rf /usr/share/modsecurity-crs
```

A continuación, clonamos el repositorio de GitHub que contiene OWASP *ModSecurity CRS* directamente en */usr/share/modsecurity-crs*:

```
$ sudo git clone https://github.com/coreruleset/coreruleset
/usr/local/modsecurity-crs
```

Renombramos el archivo de configuración de ejemplo *crs-setup.conf.example* a *crs-setup.conf*:

```
$ sudo mv /usr/local/modsecurity-crs/crs-setup.conf.example
/usr/local/modsecurity-crs/crs-setup.conf
```

Y por último, renombramos también el archivo *REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf.example* a *REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf*:

```
$ sudo mv /usr/local/modsecurity-crs/rules/REQUEST-900-EXCLUSION-
RULES-BEFORE-CRS.conf.example /usr/local/modsecurity-
crs/rules/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf
```

Con lo anterior ya tendremos listo el nuevo conjunto de reglas de Firewall, pero necesita ser implementado en ModSecurity. Para ello, creamos un directorio de ModSecurity dentro de */etc/nginx*:

```
$ sudo mkdir -p /etc/nginx/modsec
```

Y copiamos desde el repositorio de GitHub de ModSecurity que hemos copiado en */opt* los ficheros *unicode.mapping* y *modsecurity.conf-recommended*:

```
$ sudo cp /opt/ModSecurity/unicode.mapping /etc/nginx/modsec
$ sudo cp /opt/ModSecurity/modsecurity.conf-recommended
/etc/nginx/modsec/modsecurity.conf
```

Ahora, editamos el archivo de configuración recién copiado */etc/nginx/modsec/modsecurity.conf* con tal de cambiar el valor de *SecRuleEngine* de “*DetectionOnly*” a “*On*”:

```
$ sudo nano /etc/nginx/modsec/modsecurity.conf
```



```
# -- Rule engine initialization -----
# Enable ModSecurity, attaching it to every transaction. Use detection
# only to start with, because that minimises the chances of post-installation
# disruption.
#
SecRuleEngine On
```

Figura 41. Inicialización del motor de reglas

Para terminar con la configuración de ModSecurity con el nuevo conjunto de reglas, creamos un nuevo fichero de configuración `/etc/nginx/modsec/main.conf`, en el que deberemos incluir y especificar las reglas y la configuración de ModSecurity a Nginx:

```
$ sudo nano /etc/nginx/modsec/main.conf
```

```
Include /etc/nginx/modsec/modsecurity.conf
Include /usr/local/modsecurity-crs/crs-setup.conf
Include /usr/local/modsecurity-crs/rules/*.conf
```

Figura 42. Contenido de `main.conf`

Y editamos el archivo de configuración de los `server block` que tengamos en nuestro servidor Nginx para activar ModSecurity mediante las directivas `modsecurity` (que activa el WAF) y `modsecurity_rules_file` (que indica la ruta del fichero de reglas). Nótese que debemos realizar el siguiente proceso para cada `server block` en `/etc/nginx/sites-available/`.

En nuestro caso contamos con un `server block` llamado `"tfmpi.ddns.net"`, para el cual activamos ModSecurity de la siguiente forma:

```
$ sudo nano /etc/nginx/sites-available/tfmpi.ddns.net
```

```
server {

    # SSL configuration
    #
    # listen 443 ssl default_server;
    # listen [::]:443 ssl default_server;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/tfmpi.ddns;

    modsecurity on;
    modsecurity_rules_file /etc/nginx/modsec/main.conf;
```

Figura 43. Configuración de `server block` con ModSecurity activo

### 3.4. Configuración segura de MariaDB (MySQL)

Cuando realizamos una nueva instalación de MySQL o, en nuestro caso, MariaDB, es importante realizar varias tareas que afectan a la seguridad de la base de datos. Éstas son: incluir una nueva contraseña para el usuario *root*; no permitir el inicio de sesión remoto de *root*; eliminar los usuarios anónimos; y el borrado de las bases de datos de test.

Las tareas nombradas pueden ser realizadas de forma manual a través de los archivos de configuración, en */etc/mysql/my.cnf* para MySQL y en */etc/mysql/mariadb.conf.d/50-server.cnf* para el servidor MariaDB. Sin embargo, existe un script llamado *mysql\_secure\_installation* disponible para MySQL y MariaDB, y cuyo objetivo es automatizar la configuración de las tareas nombradas anteriormente.

Para utilizar *mysql\_secure\_installation* basta con ejecutar el siguiente comando y seguir las indicaciones del asistente:

```
$ sudo mysql_secure_installation
```

El asistente nos pedirá que introduzcamos la contraseña actual de *root*. Si acabamos de instalar MariaDB, tal contraseña no existe. Simplemente pulsamos la tecla ENTER:

```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!
```

```
In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.
```

```
Enter current password for root (enter for none):
```

```
OK, successfully used password, moving on...
```

A continuación, nos preguntará si deseamos utilizar la autenticación mediante el *unix\_socket*. Junto con la contraseña, esta opción aporta una capa más de seguridad para el usuario *root*. En este caso la activaremos:

```
Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.
```

```
You already have your root account protected, so you can safely answer 'n'.
```

```
Switch to unix_socket authentication [Y/n] y
```

```
Enabled successfully!
```

```
Reloading privilege tables..
```

```
... Success!
```

Siguiendo con las opciones de autenticación de *root*, el asistente nos preguntará si queremos cambiar la contraseña de *root*. Si es una nueva instalación de MariaDB, es el momento en el que podremos indicar la contraseña para *root*:

```
You already have your root account protected, so you can safely answer 'n'.
```

```
Change the root password? [Y/n] y
```

```
New password:
```

```
Re-enter new password:
```

```
Password updated successfully!
```

```
Reloading privilege tables..
```

```
... Success!
```

Una vez configurada la contraseña para *root*, es el momento de eliminar los usuarios anónimos. Por defecto, la instalación de MariaDB cuenta con un usuario anónimo, cuyo único cometido debe ser facilitar la configuración inicial de MariaDB.

Una vez contemos con los usuarios de la base de datos y el usuario *root*, deberemos eliminar los usuarios anónimos:

```
By default, a MariaDB installation has an anonymous user, allowing anyone to log into MariaDB without having to have a user account created for them. This is intended only for testing, and to make the installation go a bit smoother. You should remove them before moving into a production environment.
```

```
Remove anonymous users? [Y/n] y
```

```
... Success!
```

También deberemos permitir que el usuario *root* únicamente pueda ser utilizado desde *localhost* y en ningún caso en remoto:

```
Normally, root should only be allowed to connect from 'localhost'.
This ensures that someone cannot guess at the root password from the
network.
```

```
Disallow root login remotely? [Y/n] y
... Success!
```

Al igual que los usuarios anónimos, la instalación de MariaDB también incluye una base de datos llamada “*test*”, cuyo cometido es el testeo, por lo que deberá ser eliminada:

```
By default, MariaDB comes with a database named 'test' that anyone can
access. This is also intended only for testing, and should be removed
before moving into a production environment.
```

```
Remove test database and access to it? [Y/n] y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!
```

Una vez hemos configurado todas las opciones anteriores, el script nos dará la opción de aplicar todos los cambios, que se consigue recargando las tablas de privilegios. Indicando que queremos recargar las tablas de privilegios, el script aplicará los cambios y habremos terminado con la configuración:

```
Reloading the privilege tables will ensure that all changes made so
far will take effect immediately.
```

```
Reload privilege tables now? [Y/n] y
... Success!
```

```
Cleaning up...
```

```
All done! If you've completed all of the above steps, your MariaDB
installation should now be secure.
Thanks for using MariaDB!
```

Lo anterior nos habrá dejado una configuración de MariaDB (o MySQL) bastante segura. Sin embargo, podemos hacer algún que otro cambio para poder incrementar la seguridad de la base de datos.

Uno de los riesgos más evidentes que pone en peligro nuestra base de datos son los ataques de tipo inyección SQL, mediante el cual un atacante podría manipular nuestras bases de datos, accediendo a información confidencial.

Para evitar los ataques SQLi en MariaDB podemos añadir dos directivas (*local-infile* y *secure-file-priv*) en el fichero de configuración del servidor, ubicado en */etc/mysql/mariadb.conf.d/50-server.cnf*:

```
local-infile      = 0
secure-file-priv  = /dev/null
```

Figura 44. Configuración de MariaDB para evitar SQLi

Por último, como ocurre en cualquier otro servidor o sistema, la gestión de permisos y privilegios de los distintos usuarios es crucial para la integridad del servidor de base de datos. El control de los privilegios de usuario en MySQL y MariaDB se realiza a través de dos acciones principales [6, p. 218]. La primera de ellas, para comprobar los privilegios de todos los usuarios de MariaDB:

```
MariaDB [(none)]> select distinct(grantee) from
information_schema.user_privileges;
```

Y la segunda, para comprobar los privilegios con los que cuenta un usuario concreto del servidor:

```
MariaDB [(none)]> SHOW GRANTS FOR 'sergi'@'localhost';
```

### 3.5. Configuración segura de PHP

El último de los componentes de nuestro entorno LEMP que nos queda por atender es el lenguaje de programación del lado del servidor: PHP. Hay dos aspectos principales en lo que a fortificación de PHP se refiere: la correcta configuración del fichero *php.ini* por un lado, y la actualización de PHP a las últimas versiones disponibles con soporte. Empezaremos por este último.

#### 3.5.1. Usar versiones actualizadas de PHP en Raspberry Pi OS

La correcta actualización del software es algo que aplica a todos los sistemas, y no podía ser menos para el caso de PHP. En el momento de redactar este

trabajo, las versiones de PHP que reciben un soporte activo son únicamente las versiones 8.1 y 8.2. Respecto a las versiones que reciben únicamente soporte de seguridad, encontramos la versión 8.0. Respecto a las versiones 7.4, 7.3 y anteriores, ya no reciben soporte de seguridad, por lo que cualquier vulnerabilidad descubierta no será parcheada. Por lo anterior, es vital conocer el ciclo de vida de las versiones de PHP, que puede consultarse en <https://www.php.net/supported-versions.php>, a fin de mantener siempre una versión que cuente con, al menos, soporte de parcheado de seguridad.

El uso de una versión actualizada de PHP en Raspberry Pi OS se complica, ya que la última versión disponible en los repositorios oficiales del sistema operativo es PHP 7.4. La opción más rápida con tal de utilizar una versión con soporte de PHP en la Raspberry Pi es descargar e instalar de forma manual la última versión desde <https://www.php.net/downloads>. Sin embargo, también podemos añadir el repositorio de *Sury.org* el cual contiene las versiones actualizadas de PHP.

Para ello, descargamos la clave GPG:

```
$ sudo wget -qO /etc/apt/trusted.gpg.d/php.gpg
https://packages.sury.org/php/apt.gpg
```

Añadimos el repositorio de PHP y actualizamos la lista de paquetes:

```
$ echo "deb https://packages.sury.org/php/ $(lsb_release -sc) main" |
sudo tee /etc/apt/sources.list.d/php.list
$ sudo apt update
```

Una vez actualizada la lista de paquetes, ya podremos instalar las nuevas versiones de PHP mediante APT como, por ejemplo:

```
$ sudo apt install php php-pfm php-mysql
```

Y de forma automática se descargará la última versión disponible en el repositorio de PHP de *Sury.org*.

### 3.5.2. Configuración segura de PHP

Como se ha indicado en la introducción, el archivo de configuración principal de PHP, leído durante el arranque, es *php.ini*. Debemos recordar que Nginx utiliza PHP-FPM. Por lo que, en este caso, el archivo de configuración que deberemos modificar se encuentra en */etc/php/8.2/fpm/php.ini*.

Dentro del fichero de configuración, podemos modificar varias directivas que afectan a la seguridad de PHP. A continuación, se muestra una tabla explicativa con todas ellas [26]. No obstante, es tarea del administrador del servidor comprender la función de cada una de ellas, a fin de ajustar la configuración a la realidad del servidor.

DIRECTIVA	DESCRIPCIÓN
Opciones generales	
<i>open_basedir</i>	Delimita la ubicación de ficheros con la que puede trabajar PHP (el directorio base). Indicando una ubicación como <i>/var/www</i> se estaría limitando el impacto de ataques tipo <i>Path Traversal</i> [6, p. 219].
<i>allow_url_fopen</i>	Habilita o deshabilita las envolturas fopen de tipo URL que permiten el acceso a objetos URL. Con el valor <i>“Off”</i> podremos evitar ataques de tipo <i>Remote File Inclusion</i> .
<i>allow_url_include</i>	Permite o deniega el uso de envolturas fopen de tipo URL con las funciones <i>include</i> , <i>include_once</i> , <i>require</i> , y <i>require_once</i> .
Gestión de errores	
<i>expose_php</i>	Expone en la cabecera HTTP la versión de PHP instalada en el servidor web. En producción su valor debe ser siempre <i>“Off”</i> .
<i>display_errors</i>	Define si los errores se muestran impresos por pantalla o si deben ocultarse al usuario. Con el valor <i>“Off”</i> se ocultarán al usuario.

Ejecutables	
<i>disable_functions</i>	<p>Permite desactivar las funciones internas de PHP que se incluyan en la directiva. Las funciones que más afectan a la seguridad son: <i>system</i>, <i>exec</i>, <i>shell_exec</i>, <i>passthru</i>, <i>phpinfo</i>, <i>show_source</i>, <i>highlight_file</i>, <i>popen</i>, <i>proc_open</i>, <i>fopen_with_path</i>, <i>dbmopen</i>, <i>dbase_open</i>, <i>putenv</i>, <i>move_uploaded_file</i>, <i>chdir</i>, <i>mkdir</i>, <i>rmdir</i>, <i>chmod</i>, <i>rename</i>, <i>filepro</i>, <i>filepro_rowcount</i>, <i>filepro_retrieve</i>, y <i>posix_mkfifo</i>.</p> <p>En la medida de lo posible, se deberán desactivar todas las funciones anteriores que no se utilicen.</p>
Subida de ficheros	
<i>file_uploads</i>	<p>Permite o deniega la subida de ficheros mediante HTTP. Si no hacemos uso de la subida de ficheros, su valor debería ser "Off".</p>



## 4. Reflexiones finales

### 4.1. Conclusiones

El presente apartado constituye la culminación del trabajo final de máster. A lo largo del proyecto, se han explorado e investigado multitud de soluciones de seguridad informática y de la información, a fin de poder aportar un documento de referencia para el uso seguro de este tipo de sistemas en entornos profesionales.

Con el trabajo finalizado, y desde una perspectiva integral de todos sus capítulos y apartados, se presentan a continuación las principales conclusiones obtenidas. Éstas se derivan del producto final obtenido, de los contratiempos y desafíos enfrentados durante su realización, así como de las limitaciones encontradas a lo largo del proyecto.

- I. Se ha investigado, de forma exhaustiva y en distintos recursos, las herramientas y soluciones existentes para la securización de sistemas informáticos. De esta forma, se ha obtenido una visión global a fin de poder seleccionar las medidas de seguridad más oportunas en cada caso.
- II. En muchos casos, ha sido necesaria la adaptación de las distintas herramientas al sistema operativo Raspberry Pi OS. Gracias a que éste está basado en Debian, en la mayoría de los casos ha sido posible realizar la implementación. En otras ocasiones, principalmente por la dificultad de adaptación o la imposibilidad de realizarlo sin afectar a otros componentes, se ha decidido descartar su uso.
- III. A fin de poder verificar el correcto funcionamiento de las distintas soluciones de seguridad, se ha logrado obtener un laboratorio de pruebas completamente funcional y en el que es posible probar las medidas de seguridad en distintos escenarios y entornos.
- IV. El capítulo sobre el bastionado de un entorno profesional se constituye como un ejemplo de puesta en práctica de la guía de bastionado. Como resultado final, se ha obtenido un servidor LEMP funcional y seguro.
- V. En la medida de lo posible, la guía ha sido redactada con la premisa de poder ser entendida y aplicada con relativa facilidad. Como resultado, la extensión del trabajo se ha visto notablemente incrementada a

consecuencia de la inclusión de ejemplos prácticos y recomendaciones específicas.

- VI. Las limitaciones de tiempo, extensión y forma del trabajo han hecho que no se pudieran añadir ciertos apartados surgidos como idea durante su realización. Por ende, se incluyen como futuras líneas de investigación.
- VII. La reducción de costes y del consumo han sido una prioridad durante todo el proyecto. El reducido coste del laboratorio con el que se han realizado las pruebas, así como el uso de software de código abierto son muestras de ello. Por ende, se ha cumplido el objetivo de hacer accesible el conocimiento y la práctica de la ciberseguridad a todas las personas.

Por lo anterior, se consideran cumplidos los objetivos planteados en la fase inicial del proyecto, dando como resultado una guía que cubre las necesidades de seguridad más importantes, y que a su vez, ayuda a cubrir la carencia de literatura sobre seguridad en sistemas Raspberry Pi.

#### 4.2. Futuras líneas de trabajo e investigación

Una de las mayores problemáticas con las que se ha hecho frente en las distintas fases del trabajo final de máster ha sido el limitado tiempo para realizarlo. Durante los tres meses que ha durado el proyecto, han surgido ideas y líneas de trabajo que, por la carga de trabajo que suponían, no han podido formar parte del proyecto.

Por ello, a continuación se recogen todas ellas con el objetivo de dotar al trabajo de continuidad y longevidad.

- El desarrollo de un sitio web que sirva como recopilatorio de las herramientas trabajadas en el proyecto. La idea final de producto se inspira claramente en sitios web como “*OSINT Framework*” (<https://osintframework.com>), un recurso que, mediante un esquema de árbol interactivo, lista y enlaza las mejores herramientas para realizar investigaciones de inteligencia de fuentes abiertas. El objetivo, por tanto, es la creación de un sitio web que de forma interactiva enumere y enlace las mejores herramientas de bastionado para Raspberry Pi.

- La investigación sobre el despliegue seguro de aplicaciones y servidores en contenedores de software (con soluciones como Docker, LXD o Podman) en Raspberry Pi.
- La inclusión de nuevas herramientas y soluciones de seguridad que cubran un mayor espectro de necesidades profesionales, haciendo la guía más completa y actualizada a los avances en materia de ciberseguridad.
- En muchos casos, se ha priorizado por el uso de herramientas de facilitan la gestión de la seguridad de los sistemas en favor de la accesibilidad. Por ello, otra línea de trabajo es la investigación de soluciones más complejas, pero que tengan un mayor grado de personalización.
- Finalmente, la monitorización y el análisis de la seguridad y de los sistemas ha sido un aspecto que, si bien ha sido mencionado en algunos apartados, no se ha podido trabajar con la profundidad que merece.

## 5. Glosario

**RISC:** del inglés *Reduced Instruction Set Computer*, es una arquitectura de diseño de procesadores diseñados para ejecutar un número muy reducido de instrucciones. De igual forma, requieren menos cantidad de hardware, y suele ser empleada en sistemas montados en una única placa.

**Debian:** sistema operativo libre basado en Linux, con un enfoque de colaboración, de voluntariado y de naturaleza no comercial, que puede ser usado en multitud de dispositivos.

**Hardware:** se refiere a los elementos físicos o materiales que conforman un sistema informático como, por ejemplo, un microprocesador o los módulos de memoria.

**Software:** se refiere al sistema formal o lógico que conforma un sistema informático. Ejemplos de ello son tanto los sistemas operativos como los programas informáticos.

**Hardening:** conocido en español como “bastionado”, es el proceso por el cual se diseñan e implementan medidas y controles con el objetivo de proteger y mantener seguro un sistema informático.

**NAS:** del inglés *Network Attached Storage*, es lo que se conoce como un sistema de almacenamiento conectado a una red, mediante el cual los usuarios de dicha red pueden almacenar y gestionar datos e información.

**PYME:** acrónimo del concepto “pequeña y mediana empresa”. Se refiere al tamaño de una empresa en lo que a volumen de ingresos, valor de patrimonio y cantidad de trabajadores se refiere.

**SBC:** o en inglés, *system basis chip*, es un circuito integrado que puede albergar varias funciones lógicas u otro tipo de interfaz en una misma placa.

**Firewall:** sistema de seguridad perimetral dedicado a la restricción del tráfico entrante y saliente de una red (o de parte de ella) mediante reglas preestablecidas. Puede encontrarse como solución de *software* o como *hardware* (y *software*) dedicado para ello.

**EDR:** un *endpoint detection and response* es una herramienta de monitorización y análisis dedicado a la seguridad de los dispositivos finales de una red, también conocidos como endpoints.

**HIDS/NIDS:** los *host intrusion detection system* (HIDS) y *network intrusion detection system* (NIDS) son herramientas de monitorización y análisis de amenazas dentro de un sistema informático mediante distintas técnicas. Según la localización en la que trabajan (los hosts de la red o la propia red) estaremos hablando de HIDS (IDS de host) o de NIDS (IDS de red).

**GNU/Linux:** mientras que GNU hace referencia a un proyecto de librerías y programas de software libre para su uso en sistemas operativos, Linux hace referencia a un núcleo de sistema operativo (lo que se conoce como *kernel*) que, junto con GNU, conforma el sistema operativo completo GNU/Linux. De este último nacen variaciones como Debian, Ubuntu y otros.

**SSH:** se refiere a *Secure Shell*, un protocolo de comunicaciones seguras entre dos sistemas bajo la arquitectura de cliente/servidor. SSH permite a los usuarios de distintos sistemas conectarse a un host determinado de forma remota y segura.

**Zero Trust:** estrategia de seguridad en la red basada en la idea de que la confianza en los sistemas y la propia red debe ser eliminada, priorizando antes por la correcta identificación y autenticación de los usuarios.

**CentOS:** es una de las distribuciones del sistema operativo GNU/Linux más conocidas. Ha día de hoy se encuentra en desuso, y ha sido remplazado por otro proyecto conocido como Rocky Linux.

**OSINT:** acrónimo en inglés de *Open Source Intelligence*, hace referencia a los procesos de inteligencia y obtención de información a partir de fuentes abiertas, cuyo acceso está al alcance de todos.

**TI:** son las siglas de “tecnologías de la información”, término que hace referencia a todos aquellos sistemas cuyo objetivo final es el almacenamiento, recuperación, transmisión y manipulación de datos e información.

**INCIBE:** acrónimo de Instituto Nacional de Ciberseguridad, empresa pública española que depende del Ministerio de Asuntos Económicos y Transformación

Digital, y cuyo cometido es dar soporte a nivel nacional en materia de seguridad informática a ciudadanos, empresas públicas y privadas, y a las administraciones públicas.

**Framework:** concepto que hace referencia a un entorno de trabajo, constituido de prácticas, criterios e incluso herramientas.

**OS:** siglas de “Operating System”, o sistema operativo.

**One liner:** término en inglés que hace referencia a un conjunto de comandos presentados en una única línea, mediante el uso de operadores lógicos, con el objetivo de evitar la ejecución de cada uno de los comandos de forma individual.

**Bash:** *Bourne-again shell* es una interfaz de usuario e intérprete de comandos de Unix. También hace referencia al lenguaje de comandos (*scripting*).

**RSA:** sistema criptográfico de clave pública que utiliza factorización de números enteros para la creación de claves. RSA proviene de las iniciales de sus creadores: Rivest, Shamir y Adleman.

**Endpoint:** hace referencia al término en castellano “dispositivo final”, que son todos aquellos sistemas conectados al final de una red, como ordenadores de escritorio, portátiles, impresoras, etc.

**Gateway:** término que hace referencia al elemento de red que hace de pasarela o puerta de enlace entre los dispositivos de una red interna y el exterior. De esta forma, permite compartir información de distintas redes que utilizan protocolos diferentes. Normalmente es una función realizada por el *Router* que da salida a Internet.

**Script:** un script es una secuencia de comandos u ordenes de código, escritos en un lenguaje de programación, y cuyo objetivo es la ejecución de funciones concretas.

**XDR:** acrónimo de *Extended Detection and Response*, es una de las soluciones más completas de seguridad de dispositivos finales, siendo una versión aún más avanzada de un EDR.

**Bash:** término que hace referencia tanto al lenguaje de programación de *scripting* (lenguaje de comandos) como a la interfaz de usuario de línea de comandos (la *Shell* de Unix).

**Blue Team:** en ciberseguridad, el Blue Team es el equipo de expertos especializados y encargados de la defensa de los activos de información y los datos.

**DMZ:** acrónimo del concepto “*demilitarized zone*”, que se traduce como “zona desmilitarizada”, es una red local aislada de la red interna de una organización cuyo objetivo es albergar los servidores o activos que deberán ser accesibles desde el exterior. El objetivo final es no exponer toda la red interna de la organización, si no únicamente la DMZ.

## 6. Referencias

- [1] P. Santamaría, «Raspberry Pi: la historia del miniPC más famoso del mundo,» EIOOutput, 5 Marzo 2021. [En línea]. Available: <https://eloutput.com/productos/gadgets/raspberry-pi/>. [Último acceso: 11 Marzo 2023].
- [2] A. Scheller, A. Alasdair y M. Talbot, «Securing your Raspberry Pi,» Raspberry Pi Foundation, 24 Abril 2022. [En línea]. Available: <https://www.raspberrypi.com/documentation/computers/configuration.html#securing-your-raspberry-pi>. [Último acceso: 9 Marzo 2023].
- [3] O. Alfandi, M. Hasan y Z. Balbahaith, «Assessment and Hardening of IoT Development Boards,» de *17th IFIP WG 6.2 International Conference, WWIC*, Bolonia, 2019.
- [4] CCN, Implementación de seguridad sobre CentOS 8 (CCN-STIC-619B), Madrid: Ministerio de Defensa, 2020.
- [5] CCN, Securización de sistemas basados en Debian (CCN-STIC-612), Madrid: Ministerio de la Presidencia, 2014.
- [6] C. Álvarez y P. González, Hardening de servidores GNU/Linux, Madrid: 0xWord, 2020.
- [7] Ż. Michał, «The Practical Linux Hardening Guide,» GitHub, 5 Abril 2020. [En línea]. Available: <https://github.com/trimstray/the-practical-linux-hardening-guide>. [Último acceso: 10 Marzo 2023].
- [8] J. Cannon, Linux Security and Hardening. The Practical Security Guide, Birmingham: Packt Publishing, 2018.
- [9] J. M. Rubio, Medidas de ciberdefensa, Barcelona: Fundació Universitat Oberta de Catalunya, 2022.
- [10] M. Palao y J.-R. Rodríguez, El gobierno de las TIC: introducción al marco de referencia COBIT, Barcelona: Fundació Universitat Oberta de Catalunya, 2022.
- [11] Raspberry Pi Foundation, «The config.txt file,» Raspberry Pi Foundation, 13 Septiembre 2021. [En línea]. Available:



[https://www.raspberrypi.com/documentation/computers/config\\_txt.html](https://www.raspberrypi.com/documentation/computers/config_txt.html). [Último acceso: 02 Mayo 2023].

- [12] Raspberry Pi Foundation, «USB Mass Storage Boot,» Raspberry Pi Foundation, 10 Septiembre 2021. [En línea]. Available: <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#usb-mass-storage-boot>. [Último acceso: 02 Mayo 2023].
- [13] P. Fromaget, «7 Ways to Disable Wi-Fi on Raspberry Pi (Lite/Desktop),» RaspberryTips, 14 Marzo 2020. [En línea]. Available: <https://raspberrytips.com/disable-wifi-raspberry-pi/>. [Último acceso: 25 Marzo 2023].
- [14] "rr-developer", «LUKS on Raspberry Pi,» GitHub, 23 Febrero 2021. [En línea]. Available: <https://github.com/rr-developer/LUKS-on-Raspberry-Pi/blob/gh-pages/index.md>. [Último acceso: 26 Marzo 2023].
- [15] P. Fromaget, «17 Consejos De Seguridad Para Proteger Tu Raspberry Pi,» RaspberryTips, 21 Agosto 2022. [En línea]. Available: <https://raspberrytips.es/17-consejos-seguridad-para-la-raspberry-pi/>. [Último acceso: 28 Marzo 2023].
- [16] WireGuard.How, «WireGuard Client: Raspberry Pi OS,» WireGuard.how, 16 Agosto 2022. [En línea]. Available: <https://wireguard.how/client/raspberry-pi-os/>. [Último acceso: 04 Mayo 2023].
- [17] Elastic, «Elastic Stack,» Elastic, 5 Noviembre 2019. [En línea]. Available: <https://www.elastic.co/es/elastic-stack/>. [Último acceso: 6 Abril 2023].
- [18] Atomicorp, «Pick the OSSEC version you want to install,» Atomicorp, 17 Febrero 2020. [En línea]. Available: <https://www.ossec.net/ossec-downloads/>. [Último acceso: 7 Abril 2023].
- [19] popcornmix, «AppArmor support,» Raspberry Pi, 26 Agosto 2015. [En línea]. Available: <https://github.com/raspberrypi/firmware/issues/468#issuecomment-135125924>. [Último acceso: 17 Abril 2023].
- [20] Centro Criptológico Nacional, «CCN-STIC-822,» Ministerio de la Presidencia, Madrid, 2012.

- [21] Amazon Web Services, «¿Qué es un certificado SSL/TLS?,» AWS, 12 Septiembre 2022. [En línea]. Available: <https://aws.amazon.com/es/what-is/ssl-certificate/>. [Último acceso: 19 Abril 2023].
- [22] S. d. Luz, «Para qué sirven FTPS, FTPES y SFTP, conoce sus diferencias,» Redes Zone, 2023 Enero 27. [En línea]. Available: <https://www.redeszone.net/tutoriales/servidores/ftps-ftpes-sftp-caracteristicas-diferencias/>. [Último acceso: 15 Mayo 2023].
- [23] T. A. Nidecki, «Nginx Security: How to Harden Your Server Configuration,» Acunetix, 02 Julio 2020. [En línea]. Available: <https://www.acunetix.com/blog/web-security-zone/hardening-nginx/>. [Último acceso: 24 Mayo 2023].
- [24] V. Gite, «Top 25 Nginx Web Server Best Security Practices,» nixCraft, 12 Septiembre 2022. [En línea]. Available: <https://www.cyberciti.biz/tips/linux-unix-bsd-nginx-webserver-security.html>. [Último acceso: 24 Mayo 2023].
- [25] A. Ahmed, «Securing Nginx With ModSecurity,» Linode (Akamai), 09 Marzo 2023. [En línea]. Available: <https://www.linode.com/docs/guides/securing-nginx-with-modsecurity/>. [Último acceso: 26 Mayo 2023].
- [26] OWASP Foundation, «PHP Configuration Cheat Sheet,» OWASP Foundation, 11 Octubre 2011. [En línea]. Available: [https://cheatsheetseries.owasp.org/cheatsheets/PHP\\_Configuration\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/PHP_Configuration_Cheat_Sheet.html). [Último acceso: 27 Mayo 2023].
- [27] Amazon Web Services, «¿Qué es IPsec?,» AWS, 17 Octubre 2022. [En línea]. Available: <https://aws.amazon.com/es/what-is/ipsec/>. [Último acceso: 5 Abril 2023].

# Anexos

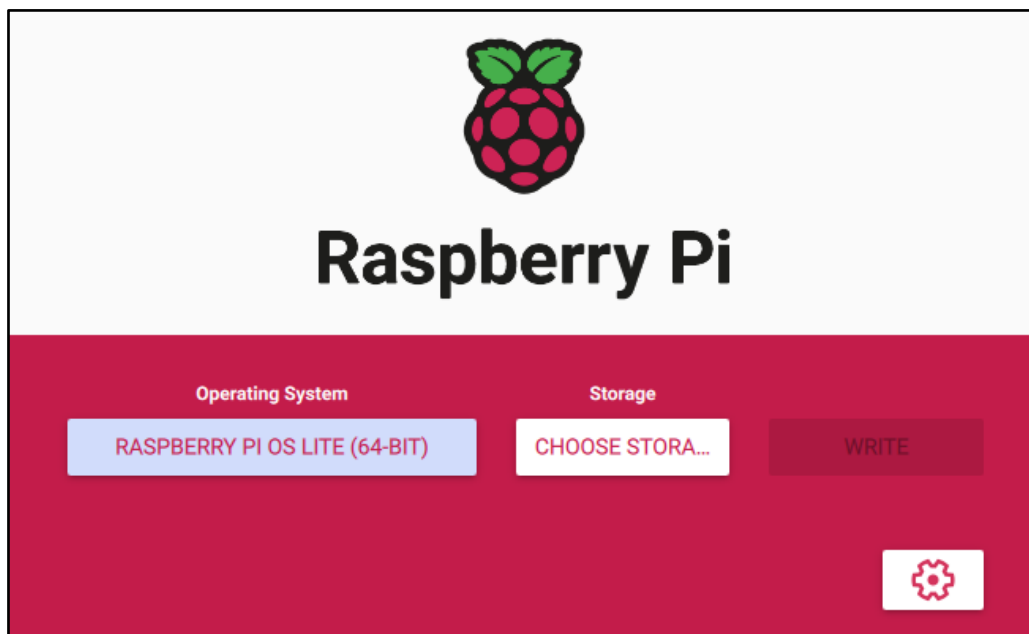
## Anexo I: Instalación de entorno LEMP

A continuación, se muestran los pasos para la instalación de un entorno o servidor LEMP en una Raspberry Pi 4 Model B.

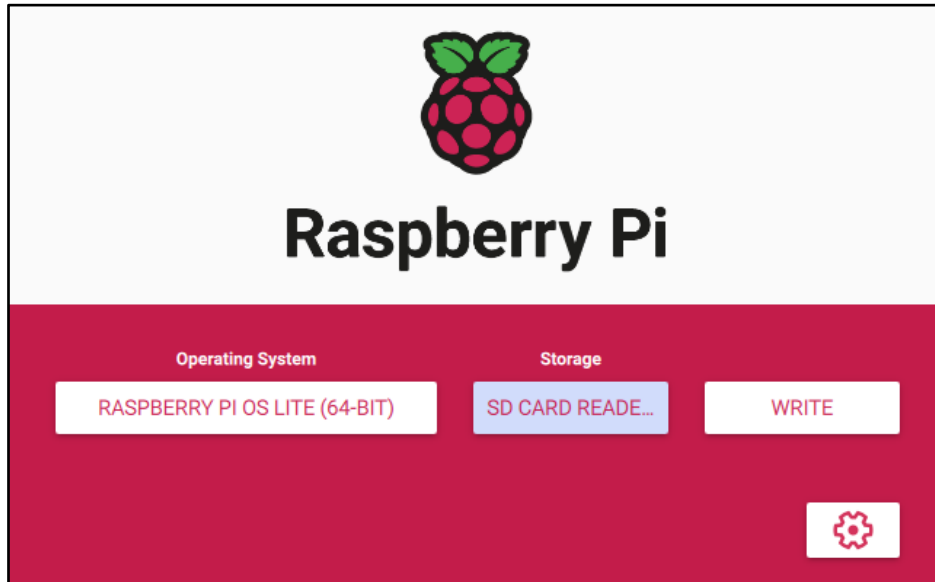
### Instalación de Linux:

En primer lugar, debemos instalar el sistema operativo Linux en nuestra microSD. Para ello, utilizaremos el *software* “Raspberry Pi Imager” (obtenible de: <https://www.raspberrypi.com/software/>), creado por la Raspberry Pi Foundation con el objetivo de facilitar la instalación de los diferentes OS. Una vez instalado en la máquina que utilizaremos para flashear la microSD, abriremos Raspberry Pi Imager y elegimos el sistema operativo a instalar en el apartado “*Operating System*”. En nuestro caso, el camino a seguir es:

*Choose OS >> Raspberry Pi OS (other) >> Raspberry Pi OS Lite (64-bit)*



Ahora, seleccionamos la unidad microSD (que deberá estar conectada a la máquina) en “Choose Storage”, ubicado en el apartado “Storage”:



Si lo deseamos, podemos preconfigurar algunas opciones en el botón que se encuentra en la parte inferior derecha (identificable por un icono de engranaje), como el *hostname*, la autenticación de SSH, o la configuración de un usuario y contraseña, entre otras opciones.

Set hostname: raspberrypi.local

Enable SSH

Use password authentication

Allow public-key authentication only

Set authorized\_keys for 'sergi': \_\_\_\_\_

Set username and password

Username: sergi

Password: .....

Una vez tengamos todo preparado, basta con pulsar el botón “Write” para que empiece el proceso de instalación del OS en la microSD.



#### Instalación del resto de componentes:

Una vez tenemos el sistema operativo instalado, podemos realizar la instalación del resto de componentes del entorno. Para ello, primero de todo deberemos actualizar la caché de repositorios y los paquetes mediante:

```
$ sudo apt update && sudo apt upgrade
```

Ahora sí, realizamos la instalación de Nginx, MariaDB y PHP de una vez en un mismo comando. Nótese que para que PHP funcione con MariaDB y Nginx, necesitaremos PHP, PHP-FPM y PHP-MySQL:

```
$ sudo apt install -y nginx mariadb-server php php-fpm php-mysql
```

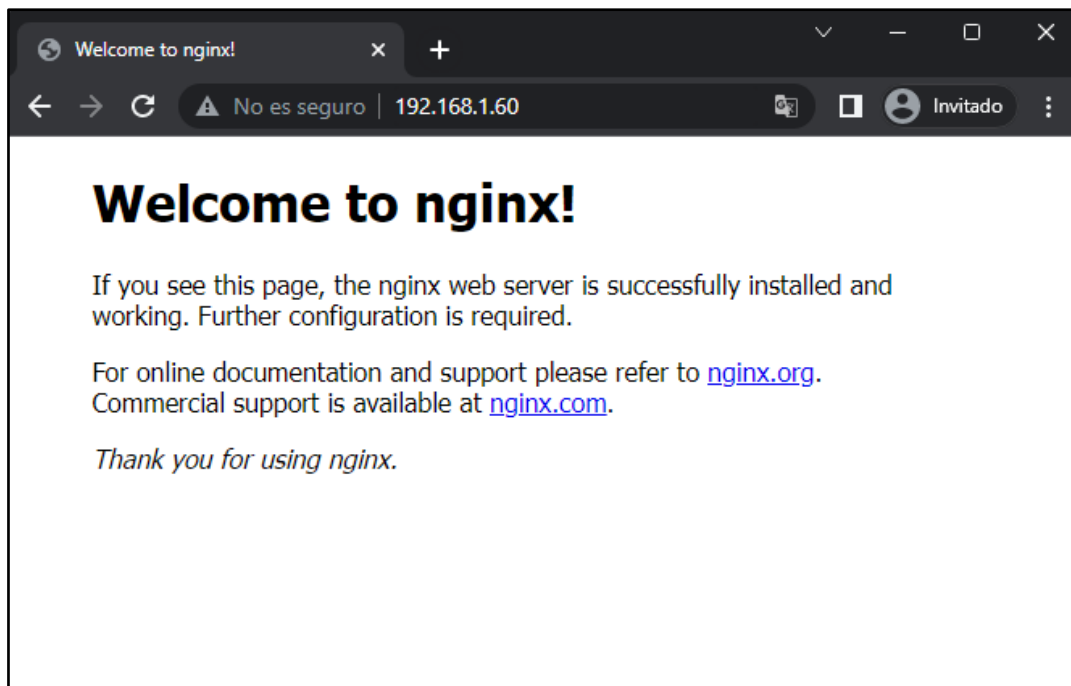
Tan pronto como termine la instalación, comprobamos el funcionamiento del servidor Nginx:

```
$ sudo systemctl status nginx
```

```
sergi@raspberrypi:/run/php $ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2023-05-11 15:50:25 CEST; 1h 21min ago
     Docs: man:nginx(8)
  Process: 18562 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -s reload (code=exited, status=0/SUCCESS)
 Main PID: 1153 (nginx)
    Tasks: 5 (limit: 3933)
       CPU: 335ms
   CGroup: /system.slice/nginx.service
           └─ 1153 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
              └─ 18563 nginx: worker process
                 └─ 18564 nginx: worker process
                    └─ 18565 nginx: worker process
                       └─ 18566 nginx: worker process

May 11 15:50:25 raspberrypi systemd[1]: Starting A high performance web server and a reverse proxy server...
May 11 15:50:25 raspberrypi systemd[1]: Started A high performance web server and a reverse proxy server.
May 11 16:58:28 raspberrypi systemd[1]: Reloading A high performance web server and a reverse proxy server.
May 11 16:58:28 raspberrypi systemd[1]: Reloaded A high performance web server and a reverse proxy server.
May 11 17:09:35 raspberrypi systemd[1]: Reloading A high performance web server and a reverse proxy server.
May 11 17:09:35 raspberrypi systemd[1]: Reloaded A high performance web server and a reverse proxy server.
```

Vemos que se encuentra activo y corriendo, por lo que si entramos en un navegador e introducimos la dirección IP de la Raspberry Pi, deberíamos ver el sitio web por defecto de Nginx:



El siguiente paso es configurar el servidor Nginx para que utilice el procesador PHP. Para ello, debemos editar el fichero `/etc/nginx/sites-available/default`:

```
$ sudo nano /etc/nginx/sites-available/default
```

Y descomentamos las siguientes líneas:

```
# pass PHP scripts to FastCGI server
#
location ~ /\.php$ {
    include snippets/fastcgi-php.conf;
#
#     # With php-fpm (or other unix sockets):
    fastcgi_pass unix:/run/php/php8.2-fpm.sock;
#     # With php-cgi (or other tcp sockets):
    fastcgi_pass 127.0.0.1:9000;
}
```

Nótese que se ha modificado la versión de PHP a la actual del servidor. También es importante no olvidar descomentar la llave cerrada “}”. De lo contrario, la verificación nos devolverá error.

Una vez guardado y modificado el archivo, verificamos que la sintaxis del documento es la correcta mediante el comando:

```
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Si la verificación ha sido positiva, podemos recargar el servicio de Nginx para que utilice la nueva configuración:

```
$ sudo systemctl reload nginx
```

Para comprobar que PHP funciona correctamente en nuestro servidor Nginx, creamos el fichero *info.php* en el directorio */var/www/html*, con el objetivo de que nos muestre la información sobre PHP a través del servidor web:

```
$ sudo nano /var/www/html/info.php
```

El contenido de *info.php* debe ser el siguiente:

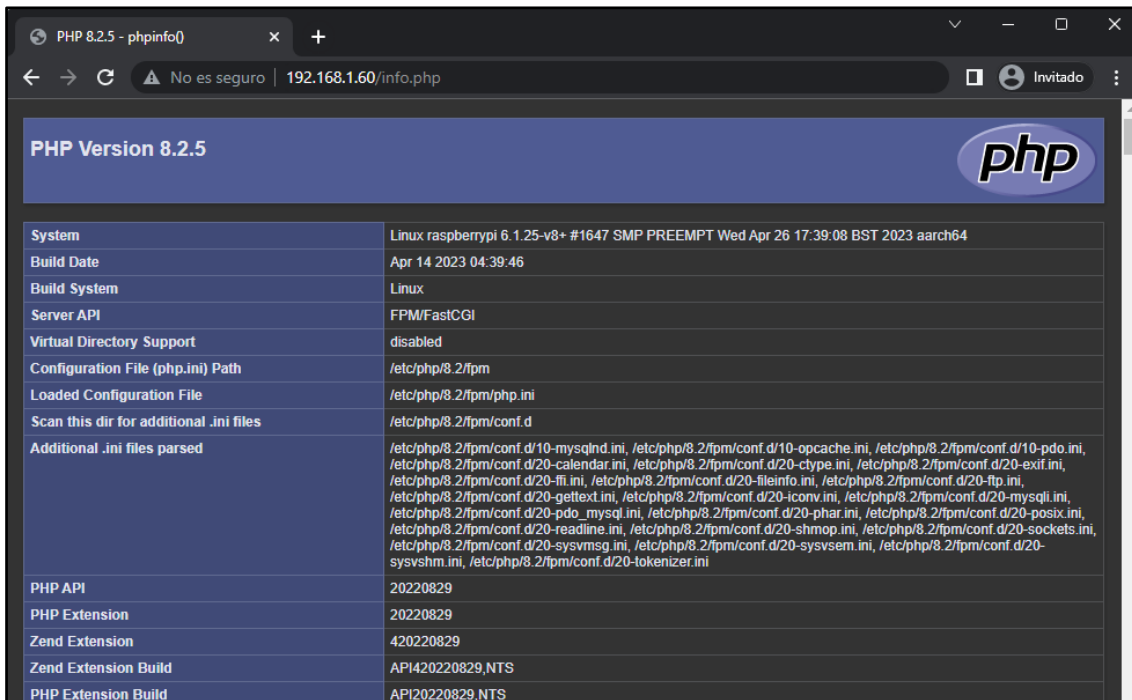
```
<?php

phpinfo();

?>
```



Ahora, si accedemos mediante el navegador a la dirección <http://192.168.1.60/info.php> (la dirección IP de nuestra Raspberry Pi), nos debería mostrar la información de PHP:



PHP Version 8.2.5	
System	Linux raspberrypi 6.1.25-v8+ #1647 SMP PREEMPT Wed Apr 26 17:39:08 BST 2023 aarch64
Build Date	Apr 14 2023 04:39:46
Build System	Linux
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.2/fpm
Loaded Configuration File	/etc/php/8.2/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/8.2/fpm/conf.d
Additional .ini files parsed	/etc/php/8.2/fpm/conf.d/10-mysqld.ini, /etc/php/8.2/fpm/conf.d/10-opcache.ini, /etc/php/8.2/fpm/conf.d/10-pdo.ini, /etc/php/8.2/fpm/conf.d/20-calendar.ini, /etc/php/8.2/fpm/conf.d/20-ctype.ini, /etc/php/8.2/fpm/conf.d/20-exif.ini, /etc/php/8.2/fpm/conf.d/20-fini.ini, /etc/php/8.2/fpm/conf.d/20-fileinfo.ini, /etc/php/8.2/fpm/conf.d/20-ftp.ini, /etc/php/8.2/fpm/conf.d/20-gettext.ini, /etc/php/8.2/fpm/conf.d/20-iconv.ini, /etc/php/8.2/fpm/conf.d/20-mysqli.ini, /etc/php/8.2/fpm/conf.d/20-pdo_mysql.ini, /etc/php/8.2/fpm/conf.d/20-phar.ini, /etc/php/8.2/fpm/conf.d/20-posix.ini, /etc/php/8.2/fpm/conf.d/20-readline.ini, /etc/php/8.2/fpm/conf.d/20-shmop.ini, /etc/php/8.2/fpm/conf.d/20-sockets.ini, /etc/php/8.2/fpm/conf.d/20-sysvmsg.ini, /etc/php/8.2/fpm/conf.d/20-sysvsem.ini, /etc/php/8.2/fpm/conf.d/20-sysvshm.ini, /etc/php/8.2/fpm/conf.d/20-tokenizer.ini
PHP API	20220829
PHP Extension	20220829
Zend Extension	420220829
Zend Extension Build	API420220829,NTS
PHP Extension Build	API20220829,NTS